UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

SAMSARA INC.,
Petitioner

v.

MOTIVE TECHNOLOGIES, INC.,
Patent Owner

---

Case IPR2026-00108
U.S. Patent No. 12,062,243

---

**DECLARATION OF TREVOR J. DARRELL, PH.D.,
IN SUPPORT OF PETITION FOR INTER PARTES REVIEW OF
U.S. PATENT NO. 12,062,243**

***Mail Stop "PATENT BOARD"***
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Samsara EX1003
Samsara v. Motive
IPR2026-00108

# TABLE OF CONTENTS

I. INTRODUCTION ..... 1
II. MY BACKGROUND AND QUALIFICATIONS ..... 3
III. UNDERSTANDING OF APPLICABLE PATENT LAW ..... 6
A. Claim Construction ..... 6
B. Obviousness ..... 6
IV. TECHNOLOGY OVERVIEW ..... 8
A. Multi-Task Training and Modeling Techniques ..... 9
B. Detecting Distracted Driving ..... 12
C. Object Detection ..... 13
D. Pose Estimation ..... 14
E. Minimizing Loss in a Neural Network ..... 15
F. Storing a Neural Network After Updating Parameters ..... 16
G. Prior Art Relied on for the Grounds of Unpatentability ..... 18
1. Chun discloses a neural network for detecting a state of a driver using a backbone and prediction heads ..... 19
2. Chao discloses systems and methods for monitoring occupants of a vehicle to determine the state of the occupant using a base model coupled to a task-specific classifier. ..... 23
3. Shanmugamani discloses methods for designing and implementing deep learning techniques for computer vision .. 26
4. Zhengyang discloses a multi-task network model for identifying an attribute of a driver. ..... 31
5. He discloses a multi-task network architecture for object detection, instance segmentation, and human pose estimation. 33
V. PROSECUTION HISTORY ..... 36
A. Examination ..... 36
B. Examiner's search versus the documents discussed in my declaration ..... 36
VI. PERSON OF ORDINARY SKILL IN THE ART ..... 39

VII. GROUNDS OF UNPATENTABILITY ..... 40

VIII. GROUND 1: CHUN IN VIEW OF CHAO RENDERS OBVIOUS CLAIMS 1, 2, 6-14, 18-20, and 24-33 ..... 40

A. Motivation to Combine ..... 41

B. Independent Claim 1 ..... 46

1. [1Pre] "A method comprising" ..... 46

2. [1A] "inputting labeled examples into a neural network, the neural network comprising:" ..... 47

3. [1B] "a backbone network, and" ..... 49

4. [1C] "one or more prediction heads coupled to the backbone network, the one or more prediction heads comprising one or more of" ..... 50

5. [1D] "a distraction classification prediction head," ..... 53

(a) Chun ..... 53

(b) Chao ..... 56

(c) MTC ..... 60

6. [1E] "object detection prediction head" ..... 66

7. [1F] "and a pose estimation prediction head;" ..... 69

8. [1G] "minimizing a loss based on outputs of the one or more prediction heads causing a change in parameters of the backbone network; and" ..... 69

(a) Chun ..... 70

(b) Chao ..... 74

(c) MTC ..... 77

9. [1H] "storing a distraction classification model after minimizing the loss, the distraction classification model comprising the parameters of the backbone network and parameters of at least one of the one or more prediction heads." ..... 81

(a) Chun ..... 81

(b) Chao ..... 83

(c) MTC ..... 83

C. Claim 2: The method of claim 1, wherein the distraction classification prediction head includes a convolutional layer, pooling layer, and fully connected layer. ..... 88

1. Chun ..... 88

2. Chao ..... 90

3. MTC ..... 93

D. Claim 6: The method of claim 1, further comprising inputting image data into the distraction classification model and receiving a distraction classification tag from the distraction classification model based on the image data. ..... 95

1. Chun ..... 96

2. Chao ..... 97

3. MTC ..... 101

E. Claim 7: The method of claim 6, further comprising performing at least one action based on the distraction classification tag. ..... 107

1. Chun ..... 108

2. Chao ..... 109

3. MTC ..... 110

F. Claim 8: The method of claim 7, the at least one action comprising one or more of updating a display of a computing device, playing an audible alarm, transmitting or a prediction to a remote server. ..... 114

1. Chun ..... 115

2. Chao ..... 116

3. MTC ..... 118

G. Claim 9: The method of claim 1, wherein the distraction classification prediction head classifies an input vector with a label, the label comprising one or more of an undistracted, mobile usage, or other distracted label. ..... 121

1. Chun ..... 121

2. Chao ..... 123

3. MTC ..... 125

H. Claim 10: The method of claim 1, wherein the object detection prediction head detects a location of an object within an input vector. .....130

I. Claim 11: The method of claim 1, wherein the pose estimation prediction head detects an orientation of one or more body parts of a user depicted in an input vector.....131

J. Claim 12: The method of claim 1, wherein storing a distraction classification model comprises storing the distraction classification model in a dash-mounted camera device. .....133

1. Chun ..... 133

2. Chao ..... 134

3. MTC ..... 137

K. Claims 13, 14, 18-20, 24, and 25 .....140

L. Independent Claim 26 .....141

M. Claim 27: The method of claim 26, wherein the distraction classification prediction head classifies the image with a label, the label comprising one or more of an undistracted, mobile usage, or other distracted label.....144

N. Claim 28: The method of claim 26, wherein the object detection prediction head detects a location of an object within the image. ....145

O. Claim 29: The method of claim 26, wherein the pose estimation prediction head detects an orientation of one or more body parts of a user depicted in the image. .....146

P. Claims 30-33 .....147

IX. GROUND 2: CHUN, CHAO, AND SHANMUGAMANI RENDERS OBVIOUS CLAIMS 3-5, 15-17, and 21-23 ..... 148

A. Motivation to Combine .....148

1. Chun's CNN Architecture for Distracted Driving Detection. 149

2. Chao's CNN Architecture for Distracted Driving Classification. ..... 151

3. Shanmugamani's Explicit Disclosure of Conventional CNN Computer Vision Techniques. ..... 153

4. Motivation to Combine ..... 154

B. Claim 3: The method of claim 1, wherein the object detection prediction head comprises a bounding box regression network and an object class prediction network, wherein the bounding box regression network outputs coordinates of a bounding box enclosing adetected object and the object class prediction network outputs a class corresponding to the detected object. ..... 156

1. Chun ..... 157

2. Chao ..... 158

3. Shanmugamani ..... 159

4. MTC ..... 163

C. Claim 4: The method of claim 3, wherein each of the bounding box regression network and the object class prediction network comprise deep neural networks having a plurality of hidden layers, each hidden layer in the plurality of hidden layers comprising convolutional layer, a batch normalization layer, and a batch activation layer. ..... 166

1. Chun ..... 167

2. Chao ..... 167

3. Shanmugamani ..... 168

4. MTC ..... 172

D. Claim 5: The method of claim 1, wherein the pose estimation prediction head includes a plurality of hidden layers and an output layer, each of the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and an activationlayer, and the output layer comprising a convolutional layer. ..... 175

1. Chun-Chao ..... 176

2. Shanmugamani ..... 176

3. MTC ..... 182

E. Claims 15-17 and 21-23 ..... 185

X. GROUND 3: ZHENGYANG IN VIEW OF HE RENDERS OBVIOUS CLAIMS 1-6, 9-18, 20-24 ..... 186

A. Motivation to Combine ..... 187

B. Independent Claim 1 ..... 191

1. [1Pre] “A method comprising” ..... 191

2. [1A] "inputting labeled examples into a neural network,".... 192

3. [1B] "the neural network comprising: a backbone network," 195

4. [1C] "one or more prediction heads coupled to the backbone network,"..... 200

5. [1D] Distraction classification head ..... 201

6. [1E] Object Detection Prediction Head ..... 204

7. [1F] Pose Estimation Prediction Head..... 207

   (a) He..... 208

   (b) MTC..... 210

8. [1G] "minimizing a loss based on outputs of the one or more prediction heads causing a change in parameters of the backbone network"..... 216

   (a) Zhengyang ..... 217

   (b) He..... 220

   (c) MTC..... 221

9. [1H] "storing a distraction classification model after minimizing the loss, the distraction classification model comprising the parameters of the backbone network and parameters of at least one of the one or more prediction heads"..... 222

C. Claim 2: The method of claim 1, wherein the distraction classification prediction head includes a convolutional layer, pooling layer, and fully connected layer.....224

1. Convolutional-Layer Element..... 225

2. Zhengyang Defines the Main Network as SE-ResNet18, a Convolutional Architecture..... 226

3. The Attention Sub-Networks in the Distraction Heads Have the Same Structure. ..... 228

4. Functional Role of the Convolutional Layers within the Distraction Heads..... 229

5. Disclosure of the Pooling-Layer Element..... 231

6. Explicit Per-Task Pooling Prior to Head Input..... 232

7. Functional and Architectural Placement within the Head.... 232

8. Functional Role and Equivalence to Claim 2 Pooling Layer. 233

9. Fully Connected-Layer Element. ..... 234

10. MTC ..... 236

D. Claim 3: The method of claim 1, wherein the object detection prediction head comprises a bounding box regression network and an object class prediction network, wherein the bounding box regression network outputs coordinates of a bounding box enclosing adetected object and the object class prediction network outputs a class corresponding to the detected object. .....241

1. Object Detection Prediction Head. ..... 241

2. Bounding Box Regression Network Outputs Object Coordinates. ..... 242

3. Object Class Prediction Network Outputs Object Class Label. ..... 244

E. Claim 4: The method of claim 3, wherein each of the bounding box regression network and the object class prediction network comprise deep neural networks having a plurality of hidden layers, each hidden layer in the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and a batch activation layer. .....247

1. Zhengyang Defines the SE-ResNet18 Architecture and Its Layer Composition. ..... 248

2. Convolutional Layer. ..... 249

3. Batch Normalization Layer. ..... 249

4. Activation Layer (Prelu). ..... 250

5. Bounding-Box Regression Network (Attention Proposal Sub-Network). ..... 252

6. Object-Class Prediction Network (Attention Networks). ..... 253

7. POSITA Understanding of Shared Architecture. ..... 254

8. MTC ..... 255

F. Claim 5: The method of claim 1, wherein the pose estimation prediction head includes a plurality of hidden layers and an output layer, each of the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and an activation layer, and the output layer comprising a convolutional layer. .....258

1. Zhengyang's Structural Correspondence and Complementarity. ..... 259

2. He ..... 259

(a) Hidden Layers (Eight 3×3 Convolutional Layers). ..... 260

(b) Output Layer (Final Convolutional Layer)..... 261

(c) Supporting Operations (Deconvolution and Upsampling). ..... 262

3. MTC ..... 262

G. Claim 6: The method of claim 1, further comprising inputting image data into the distraction classification model and receiving a distraction classification tag from the distraction classification model based on the image data.....266

1. Zhengyang..... 266

(a) Inputting Image Data into the Distraction Classification Model. ..... 266

(b) Receiving a Distraction Classification Tag from the Model. ..... 268

2. He ..... 271

3. MTC ..... 272

H. Claim 9: The method of claim 1, wherein the distraction classification prediction head classifies an input vector with a label, the label comprising one or more of an undistracted, mobile usage, or other distracted label.....277

1. Zhengyang..... 277

(a) Express Disclosure of Classification Labels. ..... 277

2. He ..... 280

3. MTC ..... 281

I. Claim 10: The method of claim 1, wherein the object detection prediction head detects a location of an object within an input vector. .....284

J. Claim 11: The method of claim 1, wherein the pose estimation prediction head detects an orientation of one or more body parts of a user depicted in an input vector.....287

1. Zhengyang ..... 288
(a) Zhengyang's Multi-Task Architecture Provides the Structural Foundation. ..... 288
(b) Spatial Outputs in Zhengyang Capture Orientation Cues. ..... 289
2. He ..... 290
(a) He (Mask R-CNN) Provides an Explicit Pose-Estimation Prediction Head. ..... 290
3. MTC ..... 292
K. Claim 12: The method of claim 1, wherein storing a distraction classification model comprises storing the distraction classification model in a dash-mounted camera device. ..... 296
L. Claims 13-18 and 20-24 ..... 298
1. Claim 13 (CRM version of claim 1). ..... 299
2. Claim 14 (CRM version of claim 2). ..... 299
3. Claims 15-17 (CRM versions of claims 3-5). ..... 300
4. Claim 18 (CRM versions of claim 6) ..... 300
5. Claim 20 (Device version of claim 1) ..... 301
6. Claim 21-23 (Device version of claims 3-5). ..... 301
7. Claim 24 (Device versions of claim 6). ..... 301
XI. GROUND 4: ZHENGYANG IN VIEW OF HE AND FURTHER IN VIEW OF CHAO RENDERS OBVIOUS CLAIMS 7-8, 19, AND 25-33 ..... 302
A. Claims 7, 8, 19, and 25 ..... 302
1. Zhengyang/He ..... 303
2. Chao ..... 310
3. *Motivation to Combine* ..... 312
B. Claims 26-33 ..... 314
1. Independent Claim 26 ..... 314
2. Claims 27-29 ..... 316
3. Independent Claim 30 ..... 316
XII. OBJECTIVE INDICIA OF NONOBVIOUSNESS ..... 317

XIII. CONCLUSION..... 318

# I. INTRODUCTION

I, Trevor J. Darrell, declare as follows:

1. I have been retained on behalf of Samsara Inc. for the above-captioned inter partes review proceeding. I understand that this proceeding involves U.S. Patent No. 12,062,243 (EX1001, "the '243 patent"), which is titled "Distracted Driving Detection Using a Multi-Task Training Process."

2. I understand the '243 patent is assigned to Motive Technologies, Inc. I also understand that the '243 patent names five inventors: Ali Hasaan, Ijaz Akhter, Muhammad Faisal, Afsheen Rafaqat Ali, and Ahmed Ali. EX1001, front page, (72).

3. I have reviewed and am familiar with the file history of the '243 patent. The file history of the '243 patent has been provided as Exhibit 1002. A summary of my understanding of the prosecution history is in Section V.

4. My opinions are based on more than 30 years of working with computer vision and pattern recognition, which includes decades of experience teaching and researching predictive models for computer vision applications. My opinions are also based on investigation and study of the relevant materials, including the patent at issue and its file history, and the prior art.

5. For example, I have reviewed and am familiar with the following prior art used in the petition for inter partes review of the '243 patent:

- "NADS-Net: A Nimble Architecture for Driver and Seat Belt Detection via Convolutional Neural Networks" to Chun et al. ("Chun") (EX1005)
- U.S. Publication No. US20200242381 to Chao et al. ("Chao") (EX1006)
- Chinese Publication No. CN111860253 to Dongguan Zhengyang Electronic Mechanical Co ltd ("Zhengyang") (EX1007)
- "Mask R-CNN" to He et al. ("He") (EX1008)
- "Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras" to Shanmugamani ("Shanmugamani") (EX1010)

6. I have been retained as an expert witness on behalf of Petitioner Samsara Inc. ("Petitioner" or "Samsara") for the above-captioned *inter partes* review (IPR). I am being compensated for my time in connection with this IPR at my standard consulting rate, which is $810 per hour.

7. I understand that this Declaration accompanies a petition for IPR involving U.S. Patent No. 12,062,243 ("the '243 patent") (EX1001), which resulted from U.S. Patent Application No. 18/470,513 ("the '513 application"),

filed on September 20, 2023. I understand that the ’243 patent alleges a priority date of June 15, 2021. I refer to this date throughout this declaration.

8. In preparing this Declaration, I have reviewed the ’243 patent and each of the documents cited herein, in light of general knowledge in the art before June 15, 2021. In formulating my opinions, I have relied upon my experience, education, and knowledge in the relevant art. In formulating my opinions, I have also considered the viewpoint of a person of ordinary skill in the art (“POSITA”) (*i.e.*, a person of ordinary skill in the field of computer vision technology, as defined further below in Section VI) prior to June 15, 2021.

## II. MY BACKGROUND AND QUALIFICATIONS

9. I am currently a Professor in the Computer Science Division (EECS) at UC Berkeley. I am also currently a member of the steering committee for the Berkeley Artificial Intelligence Research Center (BAIR). I was previously a Professor in the Department of Electrical Engineering and Computer Science at MIT from 1999 to 2008 and a Group Leader at the International Computer Science Institute (ICSI) from 2008 to 2014.

10. I have been researching and/or teaching in the area of computer science, with a particular focus on computer vision, machine learning, computer graphics, and perception-based human-computer interfaces for over 30 years. I

currently lead a laboratory where our current research focuses on algorithms to enable visual recognition across a variety of platforms and applications.

11. I am a past or current member of a number of scientific societies that focus on computer vision and machine learning, including the DARPA Information Science and Technology Advisory Study Group (2003-2014) and various National Science Foundation (NSF) Panels (2002-present). I have been invited to chair numerous conferences related to computer vision and machine learning. For example, I have served as both the Area Chair and Program Chair for the Conference on Computer Vision and Pattern Recognition (CVPR), the Area Chair for the International Conference on Computer Vision (ICCV), the Area Chair and General Chair of the International Conference on Multimodal Interfaces (ICMI), the Area Chair for European Conference on Computer Vision (ECCV), the Area Chair for the International Conference on Machine Learning (ICML), the Area Chair for the International Conference on Learning Representations (ICLR), and the Program Chair for the Conference on Perceptual User Interfaces (PUI).

12. I have been an editor for a number of leading journals in the field of computer vision and artificial intelligence, such as the Artificial Intelligence Journal (2004-2011) and the IEEE Transactions Pattern Analysis and Machine Intelligence Journal (2005-2011, 2015-2017).

13. I have published over 300 articles and conference papers related to computer algorithms, computer vision, and image processing. My publications have won several awards and honors, including the International Conference on Machine Learning (ICML) Test of Time Award in 2024, the Computer Vision Foundation Longuet-Higgins Prize in 2024, the Association for Computer Machinery SIGMM Test of Time Paper Award in 2024, the North American Associate for Computation Linguistics Best Paper award in 2016, the Computer Vision and Pattern Recognition Best Paper Honorable Mention in 2015, the ACM Multimedia Best Open Source Software Award in 2014, and the Internet Vision Workshop Best Paper Award in 2008. In 2016, my team also won the Everingham Prize at the International Conference on Computer Vision, recognizing our work in providing a significant contribution to the computer vision community. To date, my publications have been cited by other scholars over 269,000 times.

14. I am a named inventor on several patents related to object recognition and image processing. I have also served as a consultant in the area of computer vision for a number of start-up and established companies, including Pinterest (IPO), IQ Engines (acquired by Yahoo), and BotSquare/Flutter (acquired by Google). I continue to advise a number of startup companies including Nexar, Voio, and SafelyYou. I also have previously served as a consulting and/or

testifying expert witness in a number of patent infringement cases, including two ITC investigations, as described more fully in my CV (EX1004).

## III. UNDERSTANDING OF APPLICABLE PATENT LAW

15. I am not a lawyer. However, I have been advised that technical experts should apply certain legal standards in forming their opinions.

### A. Claim Construction

16. I understand that claim terms should be interpreted under a claim construction standard called *Phillips*. I understand that, according to the *Phillips* standard, claim terms should be construed according to the terms' meaning as understood by a person having ordinary skill in the art (POSITA) at the time of the invention. I understand that the claim terms should be construed in light of the specification. I also understand that the prosecution history of the patent should be consulted in determining claim meaning. In this declaration, I have construed the terms according to their ordinary and customary meanings as understood by a POSITA in view of the specification and prosecution history.

### B. Obviousness

17. I understand a patent claim is unpatentable if the claimed invention would have been obvious to a person of ordinary skill in the art (POSITA) at the time of the alleged invention. This means that a patent claim can be unpatentable even if all of the claim elements cannot be found in a single prior-art reference.

18. I understand an invention is obvious when the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time of the invention.

19. I understand that to prove that prior art renders a patent claim obvious, it is necessary to: (1) identify the particular references that, singly or in combination, teach or suggest every element of the claim; (2) identify which claim elements appear in which reference; and (3) explain why a POSITA would have sought to combine the prior-art references.

20. I understand that when considering whether a patent claim is obvious, one should consider whether at the time of the alleged invention a teaching, suggestion, or motivation to combine the references existed so as to avoid impermissibly applying hindsight. I understand this test should not be rigidly applied, but that the test can be important to avoid such hindsight.

21. I understand the following rationales can support a motivation to combine prior-art references:

(A) Combining prior art elements according to known methods to yield predictable results;

(B) Simple substitution of one known element for another to obtain predictable results;

(C) Use of known technique to improve similar devices (methods, or products) in the same way;

(D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;

(E) Choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and

(F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art.

22. I understand objective indicia (also called secondary considerations) can be important evidence as to whether a patent claim is obvious. Such indicia include: (1) commercial success of products covered by the patent claim; (2) a long-felt need for the invention; (3) failed attempts by others to make the invention; (4) copying of the invention by others; (5) unexpected results achieved by the invention; (6) praise of the invention by the infringer or others in the field; (7) taking of licenses under the patent by others; (8) expressions of surprise by experts and those skilled in the art at the making of the invention; and (9) the patentee proceeded contrary to the accepted wisdom of the prior art.

## IV. TECHNOLOGY OVERVIEW

23. The ’243 patent relates to a “multi-task modeling technique for detecting distracted driving” by using a backbone network coupled to prediction

heads for distraction classification, object detection, and pose estimation. ’243 patent, 2:43-49 and 3:3-9. In particular, the patent describes inputting labeled examples into a neural network that comprises a backbone network coupled to prediction heads including distraction classification, object detection, and pose estimation prediction heads, minimizing loss based on the outputs of the prediction heads, and storing the trained neural network. *Id*. at 2:50-62. These claimed methods are directed to neural network training for distracted driving that were well known in the art prior to the filing of the ’243 patent.

### A. Multi-Task Training and Modeling Techniques

24. Training neural networks using multi-task modeling techniques for image classification was well established before the ’243 patent. For many years, multi-task learning using CNNs has been adopted for solving different problems in computer vision. As early as 1997, Caruana established the standard framework for multi-task learning including a shared representation (later known as a backbone), and multiple task-specific outputs (or heads). This foundational architecture utilized the training of a neural network to extract features from an image and simultaneously recognize different elements of an object to better learn complex objects in the real world. EX1009. This approach known as multi-task learning exploits commonalities and differences across tasks so that multiple learning tasks can be solved at the same time. These multi-task models are generated by “learning

tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better," such that learning efficiency and prediction accuracy are improved for task-specific models. EX1009, 41. For example, Caruana discusses that multi-task learning can be used in applications such as steering direction predictions, object recognition, and decision making. EX1009, 41.

25. In 2014, a transformative CNN architecture named R-CNN, for which I am an author of (which later received a Test of Time award), provided a multi-stage modular design for object detection, which was designed to generate region proposals of images, perform feature extraction from the region proposals, and perform localization and classification of objects in the images. EX1017, 580. To address the computational bottlenecks of R-CNN, both Fast R-CNN and Faster R-CNN were developed in 2015. These architectures formalized the backbone-and-head paradigm for object detection. They use a pre-trained CNN to extract features from an image and the features are then processed by a task-specific heads to propose bounding boxes and classify objects. EX1018, 1441-42; EX1019, 3-6. In one example, Gkioxari utilizes this modular architecture by training an R-CNN with labeled input data to take a single input and simultaneously compute outputs for person detection, pose estimation, and action classification. EX1024, 1.

26. This structure of AI models proves to be beneficial as task-specific heads allow the same high-performance backbone to be reused for different applications with minimal modification. Additionally, the heads allow a model to have separate loss functions for each task, including classification, localization, and segmentation. Thus, the benefits provided by a backbone and prediction head allow for more dynamic and efficient models in many aspects of computer vision technology.

27. As R-CNN standardized an effective and efficient architecture for multi-task modeling, the two-stage model became the conventional approach to image classification and the use of the terms "backbone network" and "prediction head" became commonplace in the field. For example, Li described a multi-task model for pose estimation using a backbone network configured to extract spatial and temporal information from images and two prediction heads configured for action recognition and pose estimation. EX1020, 3595-6. Chun described a multi-task model that includes a feature pyramid backbone and prediction heads that are trained using annotated images captured from cameras inside a vehicle. EX1005, 1.

28. These AI architectures and references confirm that multi-task training and modeling techniques using backbone networks and prediction heads trained using labeled training data were routine image classification tools long before the '243 patent.

### B. Detecting Distracted Driving

29. Determining whether a driver is distracted using image classification was well established before the ’243 patent. It is widely established that distracted driving includes mental distractions like inattention, sleepiness, and fatigue, and manual distractions like talking or texting using mobile phones, eating and drinking, talking to other passengers, adjusting the radio, doing makeup, and other actions. For decades, researchers have developed systems for detecting a distracted driver. As early as 2003, researchers developed a system that robustly tracks a person’s head and facial features to determine a driver’s visual attention using a single camera. EX1011, 205. As technology in machine learning evolved, detecting distracted driving using neural networks became common practice in the field. For example, Baheti describes the detection of distracted drivers using CNNs that were trained using a dataset consisting of images labeled based on ten different distracted driving classifications. EX1012, 1145. In another example, Majdi describes a driver distraction classification model that consists of two stages: a CNN that receives an image as the first stage, and a random decision forest to predict the distracted class label of the image. EX1013, 69. The distraction classification model of Majdi was trained using training images labeled based on ten different classes of distracted driving. Thus, training a two-stage neural network using labeled training images was a well-understood step in detecting

distracted drivers. By the time of the ’243 filing, driver distraction detection models were routinely trained CNNs for detecting distracted driving using labeled images. A POSITA would have understood that applying these distracted driver detection techniques to a multi-task model with multiple prediction heads was a straightforward extension of these existing teachings.

### C. Object Detection

30. Detecting objects using a multi-task model prediction head was well established before the ’243 patent. As early as 1997, researchers were discussing the benefits of training a neural network to simultaneously recognize different elements of an object to better learn complex objects in the real world. For example, in 1997, Caruana discusses that multi-task learning can be used in applications including object recognition. EX1009, 46-47.

31. As previously described, R-CNN provided a multi-stage modular design for object detection by generating region proposals of images, performing feature extraction from the region proposals, and performing localization and classification of objects in the images. EX1017, 1-4. To address the computational bottlenecks of R-CNN, both Fast R-CNN and Faster R-CNN were developed in 2015, which formalized the backbone-and-head paradigm for object detection by using a pre-trained CNN to extract features from an image and the features are then

processed by a task-specific head to propose bounding boxes and classify objects. EX1018, 1441-42; EX1019, 3-6.

32. In one example of object detection using multi-task learning, Yang discusses well-known deep learning detectors in vehicles that utilize a two-stage detection model including a backbone network and detection head for achieving object localization and classification. EX1014, 1:46-3:61. In fact, object detection prediction heads were so well-established by the time of the ’243 filing that a patent application filed in 2020 simply relied upon “previous solutions known” to describe an object detection prediction head. EX1015, 11:28-35. These disclosures confirm that a prediction head trained for object detection was a predictable, conventional practice before the ’243 filing.

**D. Pose Estimation**

33. Pose estimation using a multi-task model prediction head was also well established before the ’243 patent. For example, Gkioxari describes training a R-CNN with labelled input data to take a single input and simultaneously compute outputs for *pose estimation*, person detection, and action classification. EX1024, 3. This pose estimation “is the task of predicting the location of specific keypoints in the human body.” *Id*. Ranjan describes the training of a CNN comprising a backbone network configured to generate feature maps of an image of a face coupled to prediction heads that are trained to classify image regions of the face in

order to estimate head pose, locate face landmarks, and recognize gender. EX1016, 1. In another example, Chun describes a multi-task model that includes a feature pyramid backbone and a prediction head for pose estimation based on images captured in a vehicle by finding keypoints of the body. EX1005, 1. These disclosures confirm that prediction heads trained for pose estimation were a predictable, conventional practice before the ’243 filing.

**E. Minimizing Loss in a Neural Network**

34. Minimizing loss in a multi-task model is a conventional element of training a neural network. For example, loss may refer to a numerical measure that quantifies a difference or “error” between the model's predicted outputs and actual values for each task it is designed to perform. And, for example, minimizing this loss can mean adjusting the model’s parameters to reduce the error across all tasks it is learning simultaneously. This can be done by combining the individual losses from each task into a single, total loss, which is then minimized through an optimization process to make the model's predictions more accurate and generalize better to new data.

35. Multi-task loss functions allow for one model to be trained and optimized for multiple tasks with a shared CNN backbone, rather than training and optimizing a separate model for each task. In multi-task learning, this is performed by utilizing individual loss functions for each task, combining the losses for each

task into a single aggregated loss, and optimizing the backbone by backpropagating this single aggregated loss and updating the weights of the CNN such that the backbone is effective for each task.

36. As early as 1997, researchers were discussing the use of loss functions and backpropagation to optimize performance on tasks in a multi-task learning model. EX1009. In Fast R-CNN, the model is trained "end-to-end with a multi-task loss" that has "one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors, rather than training a softmax classifier, SVMs, and regressors in three separate stages." EX1018, 1441-42. In further examples of distracted driver detection, researchers used algorithms including stochastic gradient descent to minimize loss by optimizing the weights of the CNN. EX1012, 1149. In Yang, the inventors include the use of detection and segmentation loss functions for the described backbone network and prediction heads. EX1014, 12:54-13:1, 15:54-65. These disclosures confirm that minimizing a loss for a multi-task learning model was a predictable, conventional practice before the '243 filing.

**F. Storing a Neural Network After Updating Parameters**

37. Storing a multi-task model after updating parameters of the network during training is a conventional element of training a neural network, as storing the multi-task model allows for the trained neural network to be deployed to perform the tasks of the model. Without storing a model after training even

temporarily, the trained neural network would not be deployable or executable. Deep-learning frameworks such as PyTorch and TensorFlow were the dominant tools used by practitioners to develop and deploy neural networks. Both frameworks provided standardized, built-in mechanisms for storing trained models, such as checkpoints in TensorFlow and state dictionaries in PyTorch, that captured the complete set of learned parameters. These serialization features made it straightforward to save a trained network in its entirety and later reload it into memory for inference. These and other frameworks enabled trained networks, including backbone and prediction head components, to be stored as a single artifact and deployed without bespoke engineering. EX1021, 62-66.

38. The storage of multi-task models in vehicles after updating the parameters of the model was also a common method in in-vehicle monitoring systems. As early as 2016, developers were implementing computer platforms in vehicles that aimed at providing autonomous vehicle and driver assistance powered by deep learning (*e.g.*, NVIDIA DRIVE). EX1022, 79. In one example, Bojarski describes PilotNet, a convolutional neural network (CNN) that learns to steer a vehicle directly from camera images. The system was trained on human driving data and is an example of an early neural network optimized and stored to run on vehicle hardware. EX1023, 1-3. These disclosures confirm that storing a multi-task

model after updating parameters for a multi-task learning model was a predictable, conventional practice before the ’243 filing.

39. In short, by the time of the ’243 patent, it was already routine to: (i) train a multi-task model using labeled training data, (ii) predict distracted driving with a distraction classification, (iii) perform object detection with an object detection prediction head, (iv) estimate a pose using a pose estimation prediction head, (v) minimize loss of a multi-task model, and (vi) store a multi-task model after training. The ’243 patent’s claims therefore reflect the combination of well-known techniques in multi-task modeling, image classification, and driver distraction detection rather than any technical innovation.

### G. Prior Art Relied on for the Grounds of Unpatentability

40. The following prior art used in the Grounds of unpatentability show that training a neural network with labeled examples, the neural network having a backbone network coupled to driver distraction, object detection, and pose estimation prediction heads, minimizing loss based on outputs of the prediction heads, and storing a distraction classification model with the backbone network and prediction heads were all well known before the ’243 patent.

**1. Chun discloses a neural network for detecting a state of a driver using a backbone and prediction heads.**

41. Chun is directed towards a convolutional neural network (CNN) architecture for driver and passenger state detection tasks, named NADS-Net ("A Nimble Architecture for Driver and Seat Belt Detection via Convolutional Neural Network"). Chun explains that "a vast majority of vehicle accidents reported worldwide are caused by distracted driving behaviors," so they propose a new CNN architecture "that is more nimble and, thus, more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches." EX1005, 1. Chun describes that NADS-Net is used to detect and alert drivers that are exhibiting distracted behavior, including but not limited to "use of smartphones/mobile devices, smoking tobaccos, engaging in a conversation with other passengers, drinking beverages, eating foods, and such," and "drowsiness, fatigues, medication effects, and other medical/physiological issues," *Id*.

42. As such, NADS-Net may be used in an in-vehicle monitoring system (IVMS) for detecting a variety of states, activities, and aspects of drivers, such as body posture and seatbelt wearing. IVMS "play a critical role in preventing and mitigating traffic accidents by alerting the distracted driver and adaptively adjusting the safety mechanisms," and are "expected to be even more critical" in autonomous driving. *Id*. Due to the importance of functional IVMS, Chun provides

previous approaches to in-vehicle human pose estimation and seatbelt detection, but "expands these previous works towards more comprehensive and reliable detection performance" and "add[s] more granularity in [seatbelt] detection results." *Id*., 2-3. In order to achieve this, NADS-Net includes a feature pyramid network (FPN) backbone and multiple detection heads that each perform different tasks, including pose estimation, seat belt detection, and key point detection.

43. Chun first describes that many pose estimation models are trained on publicly available datasets that do not have specific in-vehicle training images. *Id*., 3. Therefore, the researchers collected videos of drivers and passengers performing in-cabin activities and annotated frames of the videos to generate a training dataset, as seen in Chun's Figure 1:

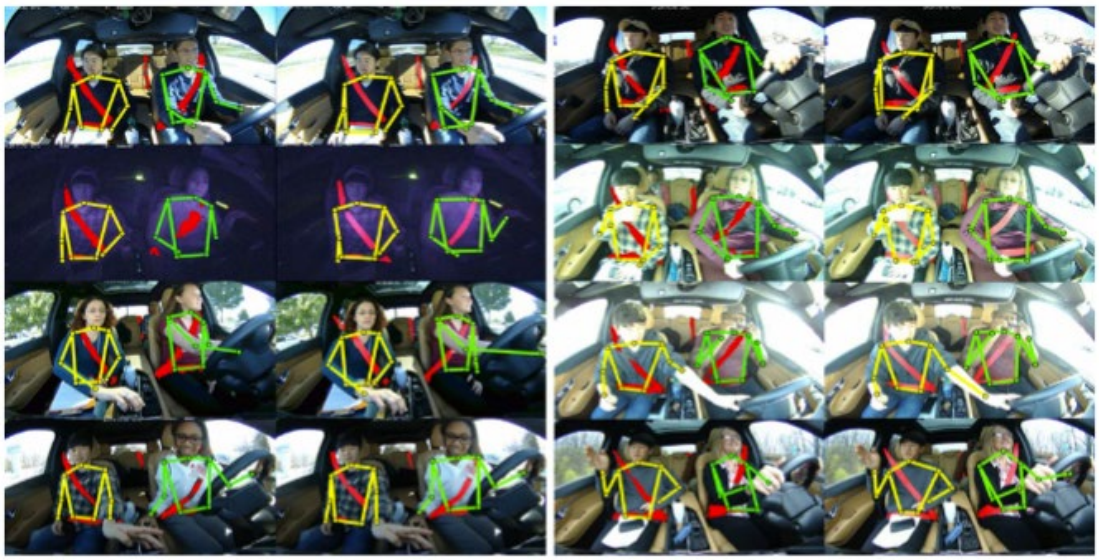


Figure 1: First and third columns in the figure are results produced by NADS-Net and second and fourth columns are the corresponding ground truth annotations.

**EX1005, FIG. 1.**

44. The videos used to obtain in-vehicle images were collected using two cameras and infrared lights installed in a research vehicle, as shown in Chun's Figure 2:

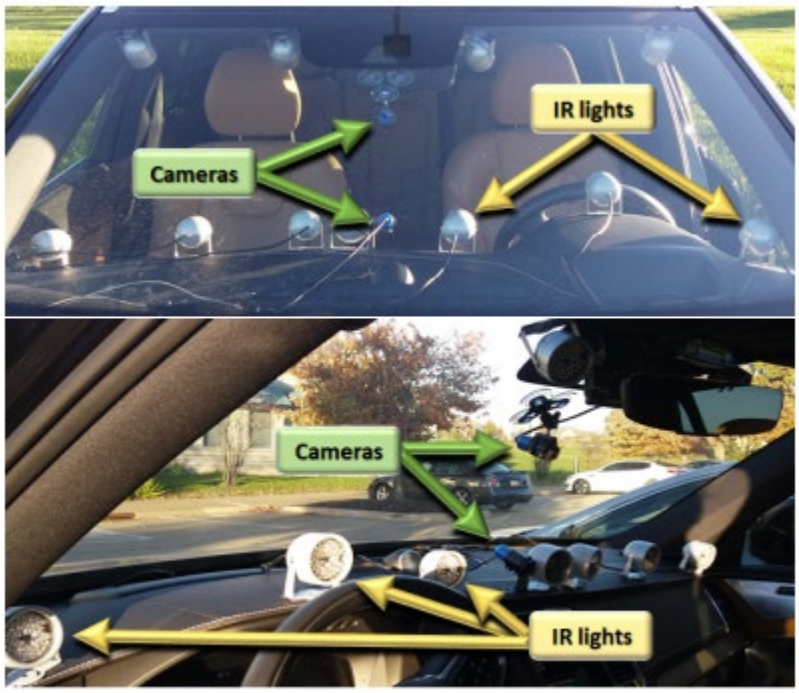



Figure 2: Data collection setup for this study.

**EX1005, FIG 2.**

45. The drivers and passengers were instructed to "perform certain gestures and motions such as drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id.*, 4. These video clips were then manually annotated "to segment all visible seat belts with a binary mask and to mark x and y pixel coordinates of body key points" to follow "the common

convention in other publicly available pose estimation data sets such as MS COCO." *Id*.

46. Using this annotated data set, NADS-Net is trained to perform pose estimation and seat belt detection using a backbone and a plurality of prediction heads. As described, "[t]he FPN backbone of NADS-Net is comprised of ResNet50 and produces a rudimentary feature pyramid for the later detection branches." *Id*., 5. This feature pyramid is used as input for the plurality of prediction heads, specifically a key point head, part-affinity field (PAF) head, and seat belt detection head. NADS-Net is illustrated in Figure 3:

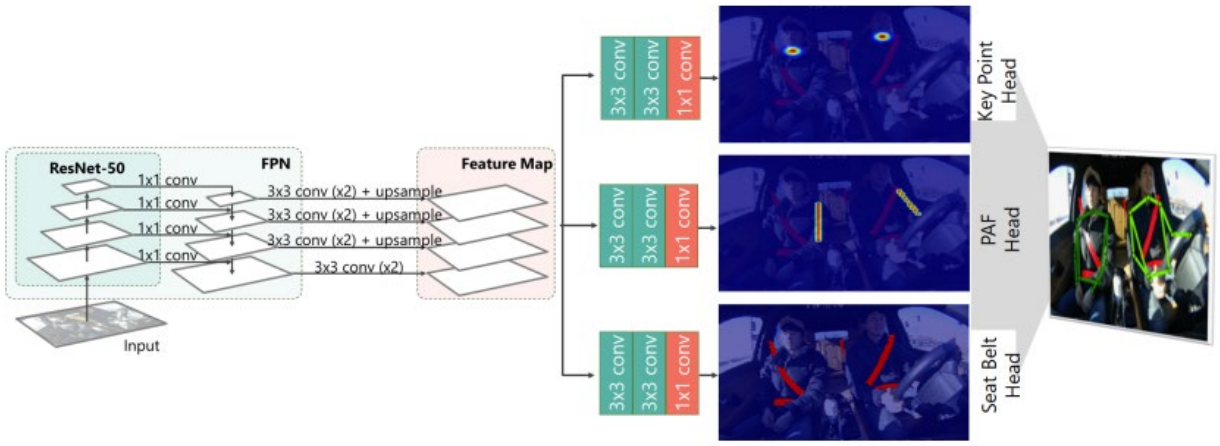



Figure 3: NADS-Net architecture.

**EX1005, FIG. 3**

47. Each prediction head "employs two 3×3 convolutions and a 1×1 convolution to predict a pixel-wise probability distribution." *Id*., 5. The key point head uses the pixel-wise probability to indicate "the probability of the

corresponding pixel being a certain joint." The PAF head uses the pixel-wise probability to "encode pairwise relationships between body joints." The seat belt detection head uses the pixel-wise distribution to indicate "the likelihood of each pixel being a seat belt. Each pixel-wise probability is then thresholded to generate a binary seat belt detection mask." *Id*., 5. These prediction heads provide both driver "pose estimation and seat belt detection in vehicles." *Id*., 8. Thus, NADS-Net is "capable of estimating body pose together with seat belt segmentation with the similar accuracy than the state of the art baseline, while requiring fewer parameters and demonstrating a faster inference time." *Id*., 8.

48. Taken together, Chun expressly teaches the key features of inputting labeled examples into a neural network and a backbone coupled to multiple prediction heads, and strongly suggests minimizing losses that update the backbone parameters and storing the resulting trained model for deployment and inference.

**2. Chao discloses systems and methods for monitoring occupants of a vehicle to determine the state of the occupant using a base model coupled to a task-specific classifier.**

49. Chao discloses systems, methods, and devices for triggering vehicular actions "based on the monitoring of one or more occupants of a vehicle." EX1006, Abstract. The primary goal of the system is to determine "states" of occupants to trigger these vehicular actions. These states often relate to driver attention and distraction, including "features/objects such as a hand, a bottle, a cellphone, a

person holding the steering wheel with both hands, a profile of a face, the back of a person's head, etc." *Id*., ¶[0098].

50. To classify these states, Chao explicitly discloses a multi-stage CNN architecture comprising a distracted classification prediction head. Specifically, Chao discloses a modular architecture comprising a CNN that acts as a backbone network and a spatio-temporal (ST) model or temporal model that act as prediction heads coupled to the CNN. The ST model acts as a distraction classification prediction head by utilizing features extracted from the base model to output a class that "may correspond to a status of the driver." *Id*., ¶[0101]. This status may include states directly corresponding to driver distraction, including "Drive both hands," "Switch gear," "Talk to passenger," "Adjust radio," "Drink," "Call," "Look right," "Makeup," and "Text". *Id*., ¶[0113].

51. Additionally, Chao explicitly discloses minimizing loss of the multi-stage CNN. Specifically, Chao describes supervised training of CNN-based models using objective functions (*i.e.*, optimization algorithms such as loss functions) and "any kind of training principle, such as backpropagation." *Id*., ¶¶[0041], [0046]. The prediction heads (*e.g.*, ST modeling block) produce classification outputs such as driver state or distraction. Those outputs are used as "positive or negative feedback" to the model during training, forming the basis of the loss function. *Id*., ¶[0043]. Chao describes that the CNN is "configured to be further trained based on

the feedback from the results produced by the ST system." *Id*., ¶[0098]. Thus, backpropagation propagates the feedback, or loss, through the prediction heads and into the backbone CNN, thereby causing changes in backbone parameters. After training, Chao explicitly states that the resulting trained CNN is stored in memory, which comprises both the backbone parameters and prediction head parameters. *Id*., ¶[0054].

52. A POSITA would understand from this disclosure that Chao teaches a conventional end-to-end learning workflow: train a backbone-plus-head network for driver distraction classification, optimize its parameters through loss minimization, and store the trained model (including all learned weights) in memory for subsequent loading and inference.

53. Accordingly, Chao provides both (1) the backbone-plus-head parameter structure that mirrors Chun's architecture for driver state detection and (2) the standard loss minimization that Chun strongly suggests. It therefore represents clear evidence of conventional training and deployment of a trained neural-network model for driver distraction classification, including a backbone and prediction-head parameters.

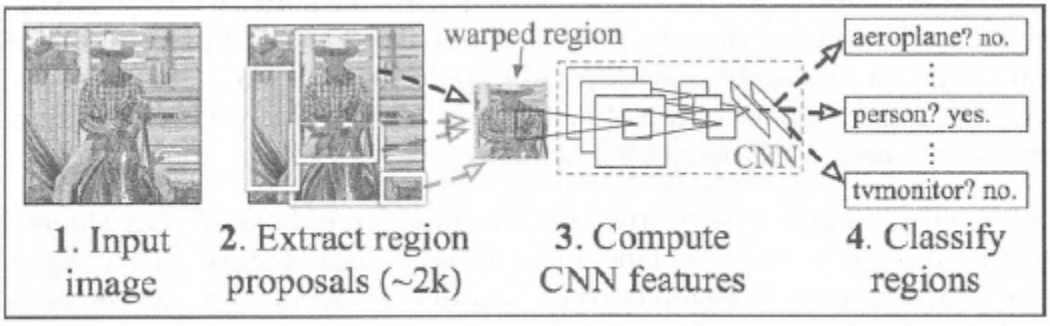
### 3. Shanmugamani discloses methods for designing and implementing deep learning techniques for computer vision.

54. Shanmugamani provides expert techniques to train advanced neural networks using TensorFlow and Keras for applications "including autonomous driving." EX1010, 7. As described, the book was written for readers "who want[] to know how to apply deep learning computer vision problems such as classification, detection, retrieval, segmentation, generation, captioning, and video classification." Using the information provided by Shanmugamani "the reader should be able to develop code for problems of person detection, face recognition, product search, medical image segmentation,… video classification, and so on." *Id*., Preface.

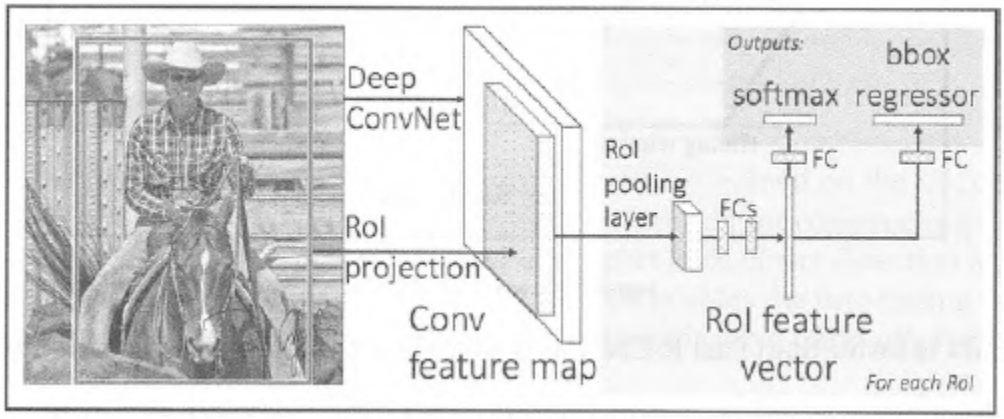
55. Shanmugamani first describes a neural network as "a collection of perceptrons and activation functions," wherein "inputs of *x* are passed through a hidden layer of perceptrons and summed to the output." *Id*., 12. Shanmugamani provides the standard architecture of CNNs and the conventional functions and layers that are utilized for training a neural network, including activation, batch normalization, backpropagation, pooling layers, hidden layers, fully connected (or dense) layers, and convolutional layers. As described "CNNs have weights, biases, and outputs through nonlinear activation" that are trained to "detect features or patterns in images." *Id*., 18.

56. Using these conventional techniques, Shanmugamani provides methods for designing and implementing object detection CNNs using bounding boxes and object class prediction. As described, "[o]bject detection is the act of finding the location of an object in an image" by "find[ing] all the objects within the image along with the labels." *Id*., 109-110. Shanmugamani provides "several variants of object detection algorithms" (*Id*., 120) including R-CNN, Fast R-CNN, Faster R-CNN, Single shot multi-box detector (SSD), and YOLO. Each of these algorithms provides a CNN architecture that determines bounding boxes and classifies the objects in the bounding boxes.

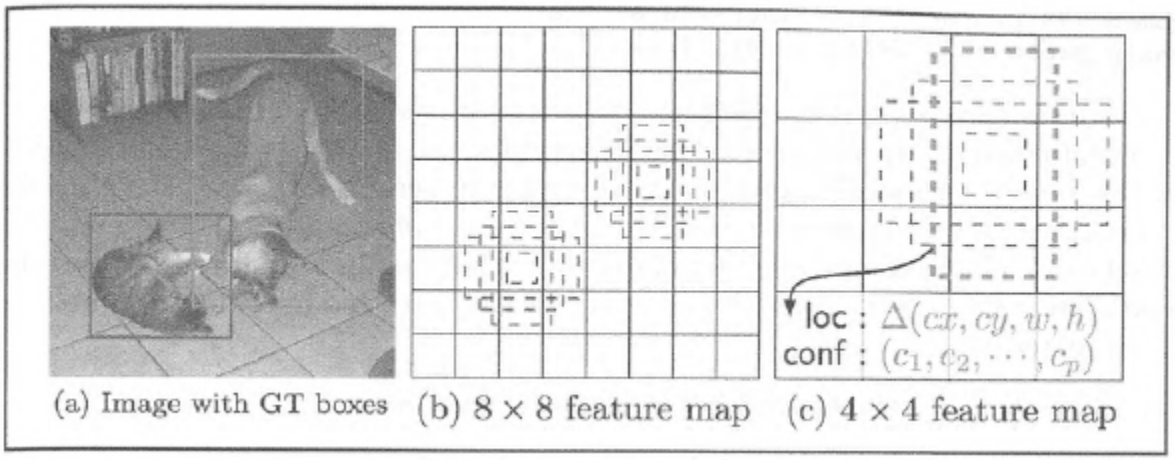
57. For example, R-CNN uses selective search to propose "regions by grouping the color/texture of windows of various sizes." These region proposals are passed through a standard CNN architecture, wherein a "linear regression model to predict a closer bounding box is trained with object region proposals" and the "last layer of the CNN is trained with an SVM identifying the object with a no-object class." *Id*., 120. The architecture of R-CNN is as follows:



58. In another example, Fast R-CNN uses Region of Interest (ROI) pooling that "takes the CNN features and pools them together according to the regions" in order to "propose the networks and select the bounding box" *Id*., 121. The architecture of Fast R-CNN is as follows:




59. In another example, SSD "simultaneously predicts the object and finds the bounding box," (*Id*., 122-123) such that the bounding boxes are labeled based on the object, as shown below:



60. Shanmugamani also provides instructions for "constructing, training, and deploying object detection models" using the "pre-trained models with various algorithms trained on the COCO dataset for public use" that are "built on top of TensorFlow." *Id*., 123. Additionally, Shanmugamani provides instructions on how to "retrain a model for our custom dataset," which involves "the preparation of a dataset, selecting the algorithm, and performing fine-tuning." *Id*., 126. As an example, Shanmugamani describes steps for training a pedestrian detection for a self-driving car using the YOLO object detection algorithm which includes a pooling layer, convolutional layer, and a dense (or fully connected) layer that include activation functions and cross-entropy functions.

61. Shanmugamani provides techniques for designing and implementing pose estimation CNNs. Specifically, Shanmugamani describes object localization that may be used for pose detection, which is "the act of finding joint locations in a human," by finding "multiple locations such as head, neck, shoulders, ankles, and hands," (*Id*., 117-118), as shown below:



62. This object localization "detects one object in an image within a label" by "predict[ing] several cropped portions of an image with an object" and regressing image coordinates using a regression encoder and classification encoder that function independently. *Id*., 110, 114, 119.

63. Thus, Shanmugamani provides description of many conventional image classification techniques, including object detection and pose estimation, and provides tutorials on how to implement said techniques in TensorFlow.

64. At a high level, I note that every challenged claim would have been obvious once the problem, detecting driver distraction from images, was posed. By 2019, the deep-learning community, informed by standard course materials such as Shanmugamani had already standardized shared-backbone convolutional networks with task-specific classification heads. A POSITA would therefore have recognized that implementing driver-distraction detection with these known frameworks involved only routine parameterization of an existing design pattern.

**4. Zhengyang discloses a multi-task network model for identifying an attribute of a driver.**

65. Zhengyang discloses a unified, end-to-end neural network architecture for recognizing multiple driver attributes in real time, corresponding closely to each limitation of claim 1. The disclosure addresses a fundamental Advanced Driver Assistance Systems (ADAS) challenge: detecting driver distraction or impairment efficiently and accurately by replacing multiple single-task models with one multi-task neural network that performs concurrent attribute recognition. As Zhengyang explains, prior systems were slow and fragmented: "the traditional method is to combine multiple models to recognize various attribute features of the driver… [which] will slow down [response speed]… hardware requirements will increase… and it is impossible to ensure that all attribute features are from the same person." EX1007, ¶[0003]. To overcome these inefficiencies, Zhengyang "adopt[s] a multi-task network model to simultaneously recognize multiple features of the driver… thereby improving the accuracy of driver state recognition" *Id*., ¶[0005].

66. During training, labeled driver images are input into the network to learn these attributes. Zhengyang describes "labeling the sample image data with multi-task attributes" and "inputting the preprocessed image data into the multi-

task main network to obtain a feature map… [then] inputting [those] into respective attribute recognition modules." *Id*., ¶¶[0024]-[0025], [0020].

67. The architecture itself matches the claimed backbone-plus-head configuration of the '243 patent: "The collected and cropped RGB image is input into a backbone network to extract the encoded features… These 7×7 feature maps are then processed through a one-squeeze multi-excitation (OSME) module to extract 4 different types of feature maps… [which are] used to determine the features of different attribute recognition tasks." *Id*., ¶[0069]; *see also* ¶¶[0020]-[0021], [0094]-[0095]).

68. Zhengyang further teaches end-to-end supervised optimization, where losses computed at each head drive updates across the entire model: "Comparing the attribute recognition results to the labeling results, reducing the difference through the corresponding loss function… to optimize network parameters" (*Id*., ¶[0022]), and training continues "by repeated forward and backward propagation… [until] the network converges… and the parameters in the network are the required final results." *Id*., ¶[0090]. A POSITA would understand that this describes minimizing head-derived losses that modify both head and backbone parameters.

69. Finally, Zhengyang's deployment discussion suggests that the trained model is finalized and stored for inference. After training, the system performs

model selection and pruning: "Selecting the network architecture and model with the best attribute analysis results…; Deleting the network that does not participate in the final attribute recognition output and its parameters to reduce the network model size and forward transmission time" (*Id*., ¶¶[0038-0039]) and then does not explicitly describe but suggests that the resulting model would be executed from its storage medium and device: "a computer-readable storage medium having stored thereon a computer program…" and "a device comprising a memory… [executing] the multi-task attribute recognition method" *Id*., ¶¶[0044]-[0045], [0137]-[0146]).

70. Taken together, Zhengyang teaches or suggests every key feature of claim 1: inputting labeled examples into a neural network, a backbone coupled to multiple prediction heads (including distraction-classification heads), minimizing losses that update the backbone parameters, and storing the resulting trained model for deployment and inference.

**5. He discloses a multi-task network architecture for object detection, instance segmentation, and human pose estimation.**

71. ***General Framework and Contributions*.** He et al., "Mask R-CNN" (2017), discloses a unified, end-to-end deep-learning framework for performing multiple visual-recognition tasks, including object detection, instance segmentation, and human pose estimation, within a single backbone-plus-head architecture. He extends the well-known Faster R-CNN pipeline by introducing a

new mask (segmentation) branch and a keypoint (pose-estimation) branch, both operating in parallel with the existing detection head. The system employs a shared convolutional backbone (*e.g.*, ResNet-50 or ResNet-101 with a Feature Pyramid Network, "FPN") that extracts dense, spatially rich feature maps. These shared features are routed to multiple, task-specific prediction heads, each trained via its own loss function.

72. ***Pose-Estimation Prediction Head.*** Section 5 of He explicitly discloses a dedicated pose-estimation prediction head implemented as a neural branch parallel to the detection and mask heads. He describes this extension as follows: "Our framework can easily be extended to human pose estimation. We model a keypoint's location as a one-hot mask, and adopt Mask R-CNN to predict K masks, one for each of K keypoint types (*e.g.*, left shoulder, right elbow)." EX1008, 2986. He details the architecture of this head: "The keypoint head consists of a stack of eight 3×3 512-d conv layers, followed by a deconv layer and 2× bilinear upscaling, producing an output resolution of 56×56." *Id*. Each of the K heads produces a spatial heatmap representing the probability distribution for a specific anatomical keypoint. The training objective minimizes a cross-entropy loss over an $m^2$-way softmax output for each visible keypoint. He reports keypoint performance metrics ("$AP_{kp}$") on the COCO dataset, confirming the effectiveness of this head for pose estimation.

73. ***Multi-Task Training and Backpropagation.*** He explicitly states that the keypoint head is trained jointly with the other heads using standard gradient-based optimization: "We have a unified model that can simultaneously predict boxes, segments, and keypoints while running at 5 fps." EX1008, 2987. The loss functions for detection, segmentation, and keypoints are combined into a total loss, and backpropagation updates both the shared backbone and the individual heads. *Id*., 2982. He emphasizes that multi-task learning improves performance and demonstrates that the pose-estimation head benefits from joint optimization with other tasks, highlighting its architectural compatibility with multi-head networks like Zhengyang's. *Id*., 2986-7.

74. At a high level, I note that every challenged claim would have been obvious once the problem, detecting driver distraction from images, was posed. By 2019, the deep-learning community, informed by architectures such as He and standard course materials had already standardized shared-backbone convolutional networks with task-specific classification heads. A POSITA would therefore have recognized that implementing driver-distraction detection with these known frameworks involved only routine parameterization of an existing design pattern.

# V. PROSECUTION HISTORY

## A. Examination

75. During the prosecution history of the ’243 patent, there was only a restriction requirement from the Examiner before allowance.

76. In the restriction, the Examiner stated that claims 21-45 are directed to training a neural network using labeled inputs and minimizing loss and claims 46-53 are directed to using a neural network in a vehicular imaging system to determine if a driver is distracted. EX1002, 95-100.

77. In response, the Applicant filed an amendment to claims 46 and 50 to recite the specific training steps of claim 21 or 35. The Applicant also filed a terminal disclaimer over U.S. Patent Nos. 11,798,298 and 11,532,169 and minor amendments to overcome clarity issues provided by the Examiner in an interview. EX1002, 110-120 and 122-125.

78. The Examiner then allowed all claims. EX1002, 145-146.

## B. Examiner’s search versus the documents discussed in my declaration

79. I have reviewed the Examiner’s Search Strategy and Results (“Search Strategy Statement”) recorded in the application file, as well as the Information Disclosure Statement (“IDS”) and the references cited therein (including U.S. Patent Application Publication Nos. 2021/0117728 A1 and 2017/0357866 A1). Based on that review, I understand that the Examiner’s search was conducted

primarily within the U.S. patent application database, focusing on CPC classes G06K 9/00, G06K 9/62, and G06N 20/00, and keyword strings associated with machine-learning model architectures and video classification.

80. The Examiner's search appears to have emphasized general feature-extraction networks and machine-learning prediction heads, but did not extend to non-patent literature, foreign publications, or search terms specifically addressing in-vehicle driver monitoring, multi-task convolutional neural networks (CNNs), pose-estimation heads, or multi-branch backbones. The Examiner's results are consistent with a conventional keyword and CPC-based automated search limited to high-level architectures for "video processing" and "image recognition."

81. By contrast, in connection with this declaration, I have reviewed the documents discussed in my declaration that appear to have been found by myself and others using searches that have been conducted and designed to capture the state of the art most relevant to the features of the challenged claims. The search appears to have included both patent and non-patent literature and used broader search terms and appears to have used technically targeted search strings such as "driver distraction CNN," "multi-task convolutional neural network," "pose estimation in-vehicle monitoring," "shared backbone classification head," and "TensorFlow driver state." From documents discussed in my declaration, the literature relating to multi-modal neural networks, self-supervised CNNs, and

driver-attention systems was also discovered, and international patent publications and academic papers were searched in addition to U.S. patent filings.

82. Based on the documents discussed in my declaration, the search appears to have focused on identifying references that explicitly describe multi-head or multi-task CNN architectures in the context of driver-monitoring or in-cabin vision systems, such as the prior-art references Chun, Chao, Zhengyang, and He, which the Examiner did not identify. These references disclose the precise architectural concepts recited in the claims, namely, shared backbones with dedicated classification, detection, and pose-estimation heads and thus are materially closer to the claimed subject matter than the art cited by the Examiner.

83. The Examiner's IDS and cited references (including Lee '728 and the '866 publication) generally relate to large-scale distributed training or cloud-based ML frameworks and do not address in-vehicle multi-head CNNs or driver-state classification. In my opinion, the Examiner's search was limited in both technical depth and domain scope. It focused on generic ML architectures rather than domain-specific applications such as in-cabin monitoring, and it did not incorporate standard TensorFlow terminology (*e.g.*, "conv2D," "batchnorm," "ReLU," "pose heatmaps") that a practitioner in this field would employ when searching for CNN implementations.

84. Based on the documents discussed in my declaration, the search for purposes of this proceeding, in contrast, appears to have been broader and more technically focused, incorporating cross-domain literature, source-code repositories, and academic datasets. As a result, the search located materially different, and in my view, more pertinent, prior art than that identified by the Examiner. From the additional documents I discussed, the independent search appears to demonstrate that relevant references were readily discoverable using standard, field-appropriate search techniques, and that the Examiner's narrower CPC and keyword approach was insufficient to locate them.

85. The additional prior-art references used in my declaration, Chun, Chao, Zhengyang, He, and Shanmugamani among others, form the foundation of the invalidity grounds set forth in the accompanying petition. These references collectively disclose the claimed elements of the challenged patent, including shared CNN backbones, multi-branch prediction heads, loss minimization across heads, and storage of trained backbone and head parameters.

## VI. PERSON OF ORDINARY SKILL IN THE ART

86. A person of ordinary skill in the art ("POSITA") in the technical field of the '243 patent at the relevant time would typically have had a Bachelor's degree in Computer Science with a focus on computer/machine vision, and at least two years of experience with each of: (1) machine learning (including neural

network) methods as applied to machine/computer vision and (2) classical machine/computer vision algorithms (*e.g.*, backbone networks and prediction heads for classification, object detection, pose estimation, etc.). Additional education could serve as a substitute for the experience requirement.

## VII. GROUNDS OF UNPATENTABILITY

87. The four Grounds of unpatentability in this declaration are shown below. This declaration explains why each Ground renders obvious the claims.

| Ground | Reference(s) | Basis | Claims |
|---|---|---|---|
| 1 | Chun and Chao | §103 | 1, 2, 6-14, 18-20, and 24-33 |
| 2 | Chun, Chao, and Shanmugamani | §103 | 3-5, 15-17, and 21-23 |
| 3 | Zhengyang and He | §103 | 1-6, 9-18, and 20-24 |
| 4 | Zhengyang, He, and Chao | §103 | 7-8, 19, and 25-33 |

## VIII. GROUND 1: CHUN IN VIEW OF CHAO RENDERS OBVIOUS CLAIMS 1, 2, 6-14, 18-20, and 24-33

88. **Under Ground 1**, Chun in view of Chao discloses each limitation of independent claims 1, 13, 20, 26, and 30, which recite the method, computer-readable medium, and device aspects of the same driver distraction classification.

89. The dependent claims are also disclosed or, at a minimum, rendered obvious by Chun in view of Chao.

### A. Motivation to Combine

90. A POSITA would have found it obvious to combine Chun and Chao to arrive at the claims because each reference addresses the same class of problems (*e.g.*, multi-task visual recognition within a shared-backbone neural-network architecture for distracted driving in vehicles) and provides complementary, well-understood techniques that a skilled practitioner would naturally integrate to enhance system performance.

91. While Chun strongly suggests but does not expressly disclose a specific distraction classification head and the well-known and conventional elements of loss minimization and storage of a neural network, the addition of these elements as disclosed in Chao would have been technically straightforward. This is because both architectures employ a shared CNN backbone feeding separate prediction heads and are expressly designed with modular branches that receive extracted features as input. Because Chun and Chao's backbones extract features of an image of a driver using convolutional layers and use at least one prediction head to perform driver state detection tasks using the input from the backbones, a POSITA would have recognized that the same features described in Chao would be applicable to Chun without altering the network's fundamental design. Specifically, to implement the distraction classification head of Chao into Chun, it would simply require a POSITA to deploy the distraction head of Chao in

addition to the pose estimation head and the object detection head of Chun, which is technically straightforward. With regards to loss minimization and storage of the neural network, Chao simply provides explicit disclosure of well-known and conventional methods that had already been used across a range of machine-learning models and devices. A POSITA would naturally apply the same technique described in Chao to Chun's driver state detection model, recognizing it would provide the same benefits, namely, a neural network implemented in devices that have optimized weights and improved accuracy, without requiring any inventive insight or leap. For these reasons, adding the elements of a distraction classification prediction head, loss minimization, and storage of the neural network would be technically straightforward and obvious to a POSITA.

92. The motivation to combine Chun and Chao would have been straightforward and strong because both Chun and Chao directly address the need for in-vehicle systems to be able to detect distracted driving using a multi-stage CNN architecture for detecting the state of a driver in a vehicle. As established, distracted driving is a major cause of vehicle accidents worldwide, and IVMS "are rapidly becoming a standard technology in consumer vehicles," for "preventing and mitigating traffic accidents" and are expected to be "even more critical" in autonomous driving. EX1005, 1. Chun provides an explicit teaching of a conventional CNN architecture including a backbone network and prediction heads

for performing "driver/passenger state detection tasks" in a vehicle, including pose estimation and object detection. *Id*. To this same effect, Chao describes a multi-stage CNN architecture comprising a CNN and spatio-temporal model that determines a response time for a driver in an autonomous vehicle "to enable the identification of emerging dangerous situations due to lack of driver attention." EX1006, ¶[0096]. Chao uses the multi-stage CNN for "recognizing a driver state… such as 'paying attention,' 'texting,' 'drinking,' 'looking at the mirror' or 'looking at the radio,' etc." EX1006, ¶[0094]. Both Chun and Chao address a need for in-vehicle monitoring systems to detect distracted drivers and provides a multi-stage CNN architecture for determining the state of a driver, and thus constitute analogous art addressing the same problem space. It would be straightforward and obvious to a POSITA to combine the systems described in Chun and Chao as they both emphasize well-known CNN methods in distracted driving detection.

93. Second, given the well-established terminology and modular design pattern in the field by the priority date, a POSITA would have understood that the elements of the CNN in Chun and Chao are functionally equivalent. By 2017, the field of computer vision had standardized the terminology of backbone networks and prediction heads (*e.g.*, Faster R-CNN and Mask R-CNN). EX1005, 2. A POSITA would have been motivated to apply this widely adopted terminology and modular architecture, as shown in Chun, to Chao's CNN for several reasons. First,

separating the backbone from prediction heads enables easier adaptation to multiple tasks (*e.g.*, distraction classification, object detection, and pose estimation). Chao's 2D CNN and ST model provide this separation as the ST model receives features extracted from images by the 2D CNN as input. EX1005, ¶[0101]. Second, the reuse of a backbone across tasks was well-known to improve training and generalization of a CNN. As described, Chao utilizes "feedback from the results produced by ST system 704" to further train the 2D CNN, which indicates that the outputs of the ST model train and generalize the backbone to "improve accuracy." EX1005, ¶¶[0043], [0098]. Finally, mainstream frameworks already encoded architectures in this multi-stage form (*i.e.*, Faster R-CNN and Mask R-CNN). Both Chun and Chao disclose this standard CNN detection/prediction method using a backbone and prediction head to determine a driver's distraction state. A POSITA would have reasonable expectation that structuring Chao's CNN as a backbone network with multiple prediction heads would succeed, as it requires no inventive step, merely adopting the well-known terminology and architectural division taught by Chun.

94. Third, a POSITA would have reasonably expected that introducing an explicit prediction head for classifying the state of a driver into Chun's architecture would be without difficulty. As the purpose of a multi-stage CNN architecture is to enable easier adaptation to multiple tasks, Chun demonstrates that the feature map

provided by the backbone network is adaptable such that multiple vision-based tasks can be completed for driver state detection tasks (*i.e.*, key point detection, pose estimation, and object detection). As Chao provides a successful driver distraction classification model that utilizes features extracted from a CNN, it would be obvious to a POSITA that implementing the driver state classification of Chao to a prediction head of Chun would be successful.

95. Finally, Chao provides the complementary disclosure that Chun implies but does not expressly state: *minimizing loss* and *storing the trained model*. Chao explicitly discloses training with an objective function and backpropagation, which, by definition, minimizes the loss and adjusts network parameters. The parameters of the backbone CNN (*e.g.*, CNN 802) and the parameters of the prediction heads (*e.g.*, the ST model outputting distraction classifications) are jointly updated during training as the ST model provides feedback to further train the 2D CNN. After training, Chao explicitly states that the resulting trained system (neural network) is stored in memory, which comprises both the backbone parameters and prediction head parameters. EX1005, ¶[0054]. A POSITA would have understood that once the Chun CNN architecture is trained, the learned parameters must be stored for runtime use. Chao's explicit storage mechanism thus completes the implementation pipeline, providing the routine, well-known step of

saving the trained backbone-and-head model for detection of distracted driving on vehicle hardware.

96. In sum, combining Chun's multi-stage CNN architecture for driver state detection tasks with Chao's multi-stage CNN architecture for driver state classification would have been a predictable, complementary, and a conventional adaptation of known components. Each reference operates within the same technological field of computer-vision-based human-state recognition for driver state detection, employs compatible architectures, and uses standard training workflows. A POSITA would have been motivated to integrate these elements to enhance distracted driving detection, thereby arriving at the claimed system through ordinary skill rather than any inventive leap.

### B. Independent Claim 1

97. Chun in view of Chao discloses every element in claim 1.

#### 1. [1Pre] "A method comprising"

98. Chun proposes the development and training of a CNN architecture, which is called NADS-Net, that "utilizes the feature pyramid network (FPN) backbone with multiple detection heads to achieve the optimal performance for driver/passenger state detection tasks." EX1005, 1. The disclosure for training and validating this CNN architecture establishes the invention as a method for "2D driver/passenger pose estimation and seat belt detection that is more nimble and,

thus, more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches," which frames the steps of claim 1.

**2. [1A] "inputting labeled examples into a neural network, the neural network comprising:"**

99. Chun discloses this element.

100. Chun explicitly discloses inputting labeled examples into a convolutional neural network (CNN) for training. Chun explicitly teaches a method for both capturing and annotating frames of videos for use as training data in the disclosed CNN. For example, section 3.2 of Chun describes that "annotators were instructed to segment all visible seat belts with a binary mask and to mark x and y pixel coordinates of body key points… following the common convention in other publicly available pose estimation data sets such as MS COCO." *Id*., 4. Each annotated frame thus constitutes a labeled training example identifying both the seat-belt region and key-point coordinates, as would have been understood by a POSITA. Examples of these annotations are shown in Figure 1:

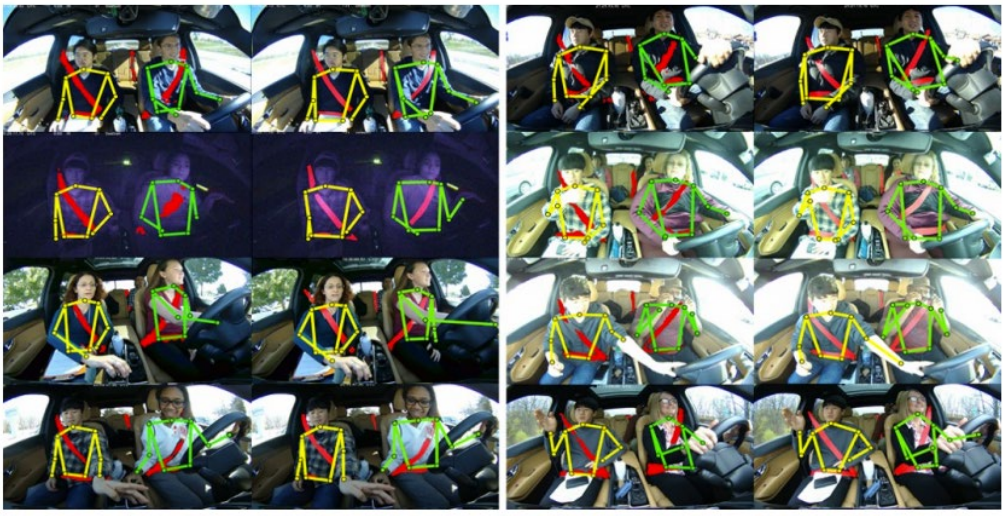


Figure 1: First and third columns in the figure are results produced by NADS-Net and second and fourth columns are the corresponding ground truth annotations.

**EX1005, FIG. 1**

101. Chun further teaches that these labeled examples were used to train the neural network. Specifically, the CNN model was "pretrained on the MS COCO data set and then transferred to the new driving data set," and performance was compared across three training regimes: "only on MS COCO data set…only on the new driving data set… and pretrained on MS COCO dataset and then transferred to the new driving data set." EX1005, 7. As described, "[t]he proposed NADS-Net was implemented in Keras with TensorFlow backend and an NVIDIA GeForce GTX 1080 Ti GPU was used for training and testing" and Chun "pre-trained the model with MS COCO train2017 data set and the corresponding human key point annotations." *Id*., 5. Thus, Chun demonstrates that the labeled examples,

*i.e.*, annotated images with key-point and segmentation labels, were input into the neural network for supervised learning. *Id*., 7.

102. Accordingly, Chun expressly discloses "inputting labeled examples into a neural network," as required by claim element [1A].

### 3. [1B] "a backbone network, and"

103. Chun discloses this element.

104. Chun expressly discloses a backbone network that performs feature extraction for images in the disclosed convolutional neural network (CNN) architecture. Chun explains that its proposed model, *NADS-Net*, "**utilizes the feature pyramid network (FPN) backbone with multiple detection heads to achieve the optimal performance for driver/passenger state detection tasks**" *Id*., 1 (emphasis added). Figure 3 of Chun depicts the NADS-Net architecture, which explicitly shows the FPN backbone feeding into three detection heads for key-point, part-affinity, and seat-belt segmentation tasks:

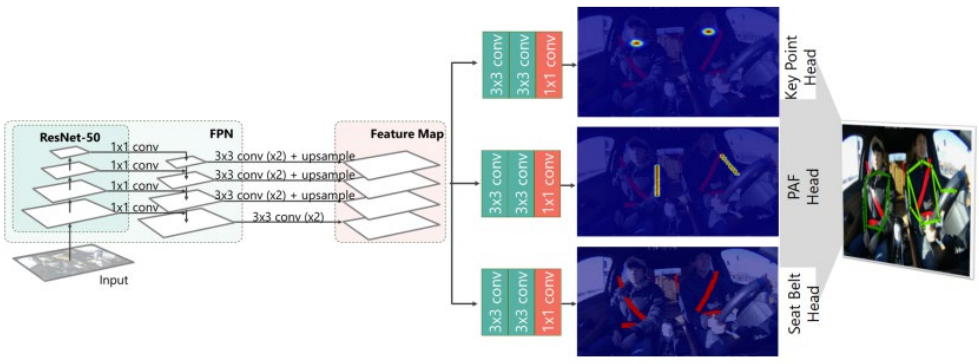



Figure 3: NADS-Net architecture.

**EX1005, FIG. 3**

105. Chun further describes that "[t]he FPN backbone of NADS-Net is comprised of ResNet50 and produces a rudimentary feature pyramid for the later detection branches" where "the inherent structure of ResNet can produce multi-resolution feature maps after each residual blocks, namely C2, C3, C4, and C5, which are sized 1/4, 1/8, 1/16, and 1/32 of the original input resolution, respectively." *Id*., 5. Chun explains that these multi-scale feature maps "are then further convolved with 1×1 convolutions … to produce a concatenated 96×96×512 feature map," which serves as the shared representation for the downstream detection heads. *Id*., 5. As explained by Chun, the backbone network provides specific benefits over the state-of-the-art approaches, specifically "generates feature maps faster and more efficient[ly]," and "can replace multiple refining stages of generic pose estimation." *Id*., 5-6.

106. Accordingly, Chun expressly discloses "a backbone network," as required by claim element [1B].

#### 4. [1C] "one or more prediction heads coupled to the backbone network, the one or more prediction heads comprising one or more of"

107. Chun discloses this element.

108. In my view, this limitation is satisfied by any single qualifying head. The phrase "one or more prediction heads" sets the count, and the phrase "comprising one or more of" the three listed categories means that the set can be

any subset of those three. A POSITA would therefore understand that the claim does not require all three head types. One head of one type is enough.

109. Chun expressly discloses one or more prediction heads coupled to a backbone network. First, Chun provides examples of previous approaches to pose estimation wherein Mask R-CNN supports "both person instance segmentation and human key point detection, on top of the Faster R-CNN's bounding box detection head." *Id*., 2. Thus, Chun establishes that the use of prediction heads in a neural network is not a novel approach to pose estimation. Using these previous approaches used for pose estimation, Chun describes that its proposed NADS-Net architecture "utilizes a feature pyramid network (FPN) backbone with multi-branch detection heads, namely, a key point detection head, a part-affinity field detection head, and a seat belt segmentation head." *Id*., 1. Figure 3 of Chun illustrates this architecture, showing the backbone network (FPN + ResNet-50) directly connected to multiple downstream detection heads that perform separate prediction tasks:

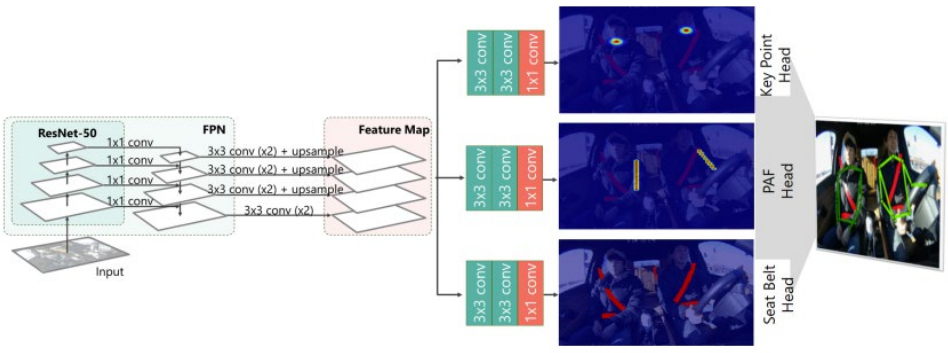



Figure 3: NADS-Net architecture.

**EX1005, FIG. 3**

110. As described, "[e]ach of the detection branches employs two 3×3 convolutions and a 1×1 convolution to predict a pixel-wise probability distribution." For the key point detection head, the pixel-wise probability "indicates the probability of the corresponding pixel being a certain joint," the part-affinity head "encode[s] pairwise relationships between body joints," and the seat belt detection head "generate[s] a binary seat belt detection mask" based on "the likelihood of each pixel being a seat belt." *Id*., 5. These components are all directly coupled to the shared feature map output of the backbone network, demonstrating multiple prediction heads that generate distinct pixel-level outputs.

111. Accordingly, Chun expressly discloses "one or more prediction heads coupled to the backbone network," as required by claim element [1C].

112. While all three prediction heads are taught by the combination of Chun and Chao, as described below in Secs. VIII.B.5-7, the claim only requires one of the distraction classification, object detection, or pose estimation prediction heads. Thus, the object detection, pose estimation, and distraction classification heads taught in Chun and Chao alone or in combination disclose the "one or more prediction heads" as required by claim element [1C].

### 5. [1D] "a distraction classification prediction head,"

113. Chun in view of Chao renders this element obvious.

#### (a) Chun

114. Chun explicitly identifies distracted driving as the primary safety problem motivating its invention. Chun presents the goal of improving the monitoring of distracted driving using IVMS because a "vast majority of vehicle accidents reported worldwide are caused by distracted driving behaviors," including "use of smartphones/mobile devices, smoking tobaccos, engaging in a conversation with other passengers, drinking beverages, eating foods, and such," in addition to "drowsiness, fatigues, medication effects, and other medical/physiological issues." EX1005, 1. As computer vision technology evolves, IVMS have become "standard technology in consumer vehicles as they can play a critical role in preventing and mitigating traffic accidents." *Id*. These systems detect and monitor "visual cues from face, eye-gaze, head-pose, hand gestures, and body poses," and generally rely upon CNNs to perform feature extraction. *Id*., 1-2. Due to this prevalence of distracted driving behaviors and the state-of-the-art approaches to vision-based sensing technologies in IVMS, a POSITA would have understood that the next logical extension of Chun's framework was to use the same CNN features not merely to detect pose or objects, but also to classify driver state (distracted vs. attentive).

115. To achieve this, Chun teaches a CNN-based IVMS that detects and tracks "visual cues from face, eye-gaze, head-pose, hand gestures, and body poses." *Id.*, 1 Chun teaches a specific CNN architecture for an IVMS that includes multiple detection heads for "optimal performance for driver/passenger state detection tasks," that is "more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches." *Id*., 1. Importantly, Chun discloses that the CNN was trained using data of drivers performing distracted driving behaviors, noting that during data collection, "the subjects were requested to perform certain gestures and motions such as drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id*., 4. These activities correspond directly to various distraction types (manual, visual, cognitive) recognized in the IVMS literature and represent labeled examples of distracted driving states.

116. Because the CNN is structured as a multi-task network that leverages a shared backbone and multiple prediction heads, a POSITA would have recognized that adding another lightweight classification branch to output a categorical distraction label was a routine extension of this architecture, consistent with multi-head CNN practices such as Mask R-CNN and OpenPose. *Id*. Because Chun's architecture already includes multiple prediction heads sharing the same backbone feature maps (*see* Figure 3), the addition of a "distraction classification

prediction head" would require only connecting a 1×1 convolution and SoftMax (or equivalent fullyconnected layer) to the shared feature map and training it on the existing annotated distraction gestures, involving no change to the backbone or data pipeline. The existing heads (key-point, part-affinity, and seat-belt segmentation) already extract pose and contextual cues necessary for distraction classification (*e.g.*, hand position, gaze direction, and seat-belt use) so extending the architecture with a distraction classification head would naturally reuse these shared features. Chun itself emphasizes that NADS-Net was designed to be a multi-branch architecture supporting "driver/passenger state detection tasks," providing an explicit motivation and structural pathway for such an extension. *Id*., 1.

117. A POSITA would have reasonable expectation of success when adding a distraction classification prediction head because the existing detection heads of the CNN all share the feature maps produced by the backbone, as shown in Figure 3. The CNN is trained using training data of a driver and passenger performing distracted driving gestures, including "drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id*., 4. Because Chun's backbone is already efficiently processing images to extract shared features necessary for pose estimation (which is integral to determining

distraction) and already includes multiple prediction heads, adding a distraction head would have been obvious as it would leverage the pose and visual data to classify the distracted behaviors the system was already designed to monitor.

118. Chun thus establishes that determining the state of a distracted driver is the primary goal of the described CNN architecture and presents architecture and training data that allow a POSITA to easily and effectively add a distraction classification head to the disclosed CNN.

119. To the extent Chun does not explicitly disclose a "distraction classification prediction head," Chao nonetheless supplies the structural (multi-branch CNN), functional (state detection), and data (annotated distraction gestures) framework that renders such a head an obvious addition in view of Chao, which provides an explicit example of a driver-distraction classification head.

**(b) Chao**

120. Chao explicitly discloses systems and methods for classifying driver distraction using a multi-stage CNN architecture that extracts spatial features and applies temporal classification to determine driver states "based on the monitoring of one or more occupants of a vehicle." EX1006, Abstract. The primary goal of the system is to determine "states" of occupants to "enable the identification of emerging dangerous situations due to lack of driver attention." *Id*., ¶[0096]. These "states" include conditions of driver attention and distraction, such as when the

driver is "'paying attention', 'texting', 'drinking,' 'looking at the mirror,' or 'looking at the radio,'" or interacting with "features/objects such as a hand, a bottle, a cellphone, a person holding the steering wheel with both hands, a profile of a face, [or] the back of a person's head." *Id*., ¶¶[0094], [0098]. These examples define the same categorical distraction states that the claimed head outputs.

121. To classify these states, Chao discloses a modular architecture comprising a CNN that acts as a backbone network and a spatio-temporal (ST) model or temporal model that act as prediction heads coupled to the CNN. The CNN is described as "a base model that identifies and extracts spatial features and/or objects from frames of a video." *Id*., ¶[0099]. The CNN is "configured to receive a plurality of subsets of image data (e.g., frames from video) and identify point locations in each of the plurality of subsets of image data." *Id*. Figure 8A explicitly shows the CNN feeding the ST model, which then produces "desired classes (recognized actions)", *i.e.*, functionally identical to a prediction head consuming backbone features and producing classification output.

122. The ST model uses the features extracted by the CNN to perform tasks such as classifying driver distraction, detecting objects, and inferring aspects of occupant pose to determine states and trigger vehicular actions. To perform these tasks, a SoftMax classification layer is "applied to the Spatio-temporal features output" of the ST model and each of the classes of the vector produced by

the SoftMax layer "is provided as a value, or "Action Score," indicating the determined class based on the input." Thus, the ST model corresponds to the claimed distraction classification prediction head by utilizing features extracted from the base model to output a class that "may correspond to a status of the driver." *Id*., ¶[0101].

123. Table 1 lists exemplary "classes (i.e., states)" such as "Drive both hands," "Switch gear," "Talk to passenger," "Adjust radio," "Drink," "Call," "Look right," "Makeup," and "Text". *Id*., ¶[0113]. Many of these directly correspond to driver distraction.

TABLE 1

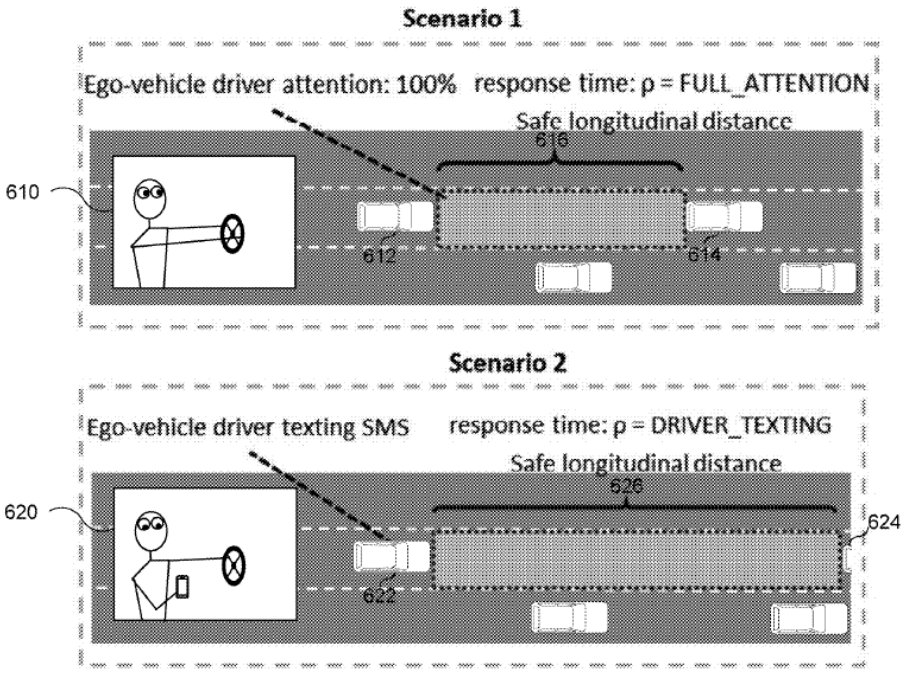
Exemplary list of classes (i.e. states) and corresponding reaction time values

| | Index | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Class | Drive both hands | Switch gear | Talk to passenger | Adjust Radio | Drink | Call left hand | Call right hand | Hair makeup | Reach side | Reach behind | Text left hand | Text right hand | Hands free |
| Reaction time | 2.0 sec | 2.1 sec | 2.4 sec | 3 sec | 3 sec | 3.2 sec | 3.2 sec | 3.5 sec | 4 sec | 4 sec | 4.2 sec | 4.2 sec | 5 sec |

**EX1006, Table 1**

124. Figure 6 of Chao provides an example illustrating "Ego-vehicle driver attention: 100% full attention" versus "Ego-vehicle driver texting SMS", which indicates the classification of driver distraction.

FIG 6




**EX1006, FIG. 6**

125. Chao therefore squarely teaches a system for classifying a distracted driver using a multi-stage CNN architecture. A POSITA would have recognized that these teachings provide conventional, well-understood methods for classifying a distracted driver using a prediction head. This is because by 2019, "backbone" was standard machine learning parlance (*e.g.*, ResNet, VGG, MobileNet serving as "backbones" in detection/segmentation pipelines), which is further exemplified by Chun's discussion of state-of the-art approaches to driver state detection tasks. A POSITA would have recognized the CNN and ST model in Chao are functionally

the same as a backbone and prediction head, even though the words "backbone" and "prediction head" are not used.

**(c) MTC**

126. Combining Chun's modular multi-head CNN with Chao's explicit distraction-classification head would have been a predictable, complementary use of prior-art elements according to their established functions. Chun provides the backbone and multi-branch framework; Chao provides the distraction prediction classification head that consumes backbone features and outputs driver-state scores.

127. A POSITA would have been motivated to combine Chun with Chao for several reasons including those outlined at the outset of Ground 1. *See* Section VIII.A. While Chun does not expressly disclose a distraction classification prediction head, it provides a multi-task neural architecture that is structurally and functionally compatible with one. Chun's system already incorporates a shared convolutional backbone (FPN backbone) and multiple specialized prediction heads for driver state detection, including pose estimation and seat belt detection. Each of these heads receives shared feature maps and produces task-specific outputs as the backbone "produces a rudimentary feature pyramid for the later detection branches." EX1005, 5. Chun's framework shows that pose estimation, segmentation, and detection can be trained from a shared backbone under separate

task-specific heads without architectural interference. Thus, this combination of a backbone network and task-specific heads provide the architectural framework on which a distraction classification head could operate.

128. Chao provides a concrete, enabling example of a distraction classification prediction head. Chao expressly discloses a distraction classification head implemented as a distinct model separate from a feature extraction model, wherein the model receives the extracted features as input.

129. A POSITA would have been motivated to combine Chao's distraction classification head with Chun's multi-task network because the combination would predictably enhance Chun's ability to perform driver state detection tasks. Chun already extracts features from and estimates a driver's pose based on images of distracted driving gestures "such as drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." EX1005, 4. Adding Chao's driver distraction classification would allow the same system to infer a driver's distraction/attentiveness state, thereby enabling a more complete assessment of distraction and readiness. The addition would have been technically straightforward: both architectures employ a shared backbone feeding separate prediction heads, and Chao's ST model for driver state classification is expressly designed as a modular branch that receives extracted features as input. Because

Chun's backbone produces feature maps of the driver's pose and seat belt donning, a POSITA would have recognized that the same feature maps could serve as input to a distraction classification head without altering the network's fundamental design.

130. Further, the motivation to combine would have been reinforced by well-established design principles in multi-task learning. Chun demonstrates that the NADS-Net architecture can perform key point detection, pose estimation, and object detection/segmentation using a shared FPN backbone, thereby improving efficiency through shared feature extraction. A POSITA would have understood that integrating a distraction classification head into Chun's existing backbone would yield similar efficiency and functional benefits, leveraging shared convolutional features to produce distraction-related classifications. The combination would therefore have been an obvious, predictable extension yielding an improved, more comprehensive driver-monitoring network consistent with Chun's stated goal of achieving "the optimal performance for driver/passenger state detection tasks." EX1005, 1.

131. Additionally, although Chun focuses its implementation on pose and seat-belt detection, its express purpose is to address distracted-driving behaviors, the same problem solved by the challenged claim. Chun identifies distracted behaviors such as "drinking, using smartphones, yawning, sneezing, leaning on the

door, [and] adjusting the sun visor" and collects training images of drivers performing those acts. A POSITA would have understood these gestures to be behavioral classes corresponding to distraction categories. Chun's reference to "driver/passenger state detection tasks" encompasses these higher-order driver states. Thus, the motivation to perform distraction classification is embedded in Chun's problem statement, not driven by hindsight.

132. Accordingly, it would have been obvious to a POSITA to extend Chun's backbone-plus-heads framework by incorporating Chao's distraction classification prediction head. Both references teach modular, multi-task networks trained for driver state detection. Chao provides an explicit blueprint for implementing a distraction classification head using nearly identical architectural conventions, even if there is a terminology gap, and Chun provides the motivating application context (*i.e.*, driver distraction detection) where distraction classification offers clear functional benefit. The combination would have been predictable and within the knowledge of one of ordinary skill in the art at the time.

133. A POSITA would have considered Chun and Chao to be analogous art addressing closely related problems within the same field of in-vehicle computer vision. Chun focuses on the structure of a CNN for detection of human key points, human pose estimation, and object (*i.e.*, seatbelt) detection within an image of a driver in a vehicle using a backbone and prediction heads. Chao addresses the

structurally similar problem of recognizing driver attributes and behaviors, such as drinking, phone use, and gaze direction, from image data by analyzing features and objects in a vehicle's internal environment using the same type of convolutional feature extraction and detection-head architecture. Both systems are directed to neural-network-based multi-task perception of human pose or behavior from visual input and rely on shared backbones feeding task-specific heads/models. Because both references lie squarely within the domain of deep-learning-based human detection and activity recognition in in-vehicle systems, a POSITA would have naturally looked to Chao when seeking to improve Chun's system with additional driver state inference capabilities.

134. Furthermore, Chao's disclosure is not only analogous but directly complementary to Chun's stated design goals. Chun's architecture seeks "achieve the optimal performance for driver/passenger state detection tasks." EX1005, 1. Chao's distraction classification head provides exactly that type of information (*e.g.*, recognizing a driver state by classifying a driver's features/objects) which would enhance Chun's ability to infer driver distraction. Thus, a POSITA would have found it entirely natural to apply Chao's distraction classification head within Chun's existing multi-head backbone to achieve a more holistic model of driver state.

135. A POSITA also would have had a reasonable expectation of success in making this combination. Both references use convolutional backbones that provide spatial feature extraction suitable for pose estimation and object detection. Chao includes a classification model that is separate from a convolutional backbone and draws from the features extracted by the CNN. Chun's network already includes multiple heads drawing from shared features, proving its compatibility with additional branches. The modification would therefore have been straightforward: the distraction classification head from Chao could be connected to Chun's shared feature maps and trained with annotated images, in the same manner as Chun's prediction heads. Further, adding a classification head to Chun's FPN backbone would have been a routine extension of a known multi-task learning paradigm by 2019. Multi-head CNNs (*e.g.*, Mask R-CNN, FPN) commonly employed shared feature extractors feeding specialized heads for detection, segmentation, and classification. Because Chun's backbone outputs dense feature maps encoding face, hand, and body-pose geometry, precisely the features used by Chao to infer distraction, a POSITA would have expected predictable success in training a new head to output categorical distraction labels. The modification would require only the addition of a soft-max classification branch and labeled examples already available in Chun's dataset. No redesign of the backbone, loss functions, or data pipeline would be necessary.

136. No technical obstacles would have discouraged a POSITA from making this combination/modification. Chao's distraction classification head is modular and uses standard convolutional layers already employed in Chun. Because both systems use CNNs to perform feature extraction, integration would not require redesign of the backbone or attention modules. A POSITA would have reasonably expected that combining Chao's distraction classification head with Chun's existing pose estimation and object detection heads would yield predictable success, producing a unified model capable of detecting both what the driver is doing and how the driver's body is oriented, thereby improving recognition accuracy consistent with Chun's objectives.

**6. [1E] "object detection prediction head"**

137. Chun discloses this element.

138. Chun discloses an object detection head for detecting a seat belt of a driver of passenger. First, Chun provides previous methods for detecting a seat belt as many IVMS use "vision-based sensing technologies" to recognize "correct donning of a seat belt." EX1005, 1. These previous approaches include common object detection methods, including "edge detection," "salient gradient maps," "radial basis functions (RBF)," "batch normalization," and "a method that crops passenger regions first using the single shot detector (SSD) and applies a CNN to detect seat belt non-use." *Id*., 2-3. Chun thus establishes that object detection in

IVMS using multi-stage CNNs is a well-known and conventional approach to seat belt detection.

139. The seat belt detection head, as described in Chun, provides detection results that "provide information on, not only use and non-use of seat belt, but also proper/improper use cases judged by the relative position of seat belt to the detected body position." *Id*., 3. The disclosed prediction head that performs seat belt detection "produces a probability density function defined over the image domain, indicating the likelihood of a pixel being a seat belt." *Id.*, 6. This probability density is then "thresholded to obtain a binary seat belt segmentation mask." *Id*. This binary seat belt mask provides "more granularity in detection results" in order to correctly detect the seat belt *and* proper use of a seat belt. *Id*., 3. An example of this seatbelt detection can be seen in Figure 5 of Chun:



Figure 5: Seat belt detection result (*left*) in comparison with the ground truth (*right*). With human inspection, the prediction result is of good quality as it correctly marks the seat belt area. However, IOU for this particular example was 46%, justifying the need for a better evaluation metric.

**EX1005, FIG. 5**

140. Thus, Chun expressly discloses a prediction head that performs object detection, specifically seat belt detection. A POSITA would have recognized that Chun's seat-belt head performs object detection within the meaning of the claim. The network detects an object class ("seat belt") and localizes it in image space via a dense probability map, functionally identical to the detection performed by bounding-box heads in architectures such as Faster R-CNN or SSD, both of which Chun expressly cites as analogous methods. *Id*., 2-3. In contemporary CNN literature, object segmentation is understood as a pixel-level generalization of object detection: both produce per-object localization outputs using convolutional prediction heads attached to a shared backbone. Thus, Chun's seat-belt detection head is a conventional object-detection head differing only in representation format (mask vs. box), not in technical substance.

141. While Chun describes seat belt detection using object segmentation, there are other well-known, conventional methods for performing object detection, including bounding box regression algorithms like "Faster R-CNN's bounding box detection head," and "single shot detector." *Id*., 2-3. A POSITA would therefore have recognized that the seat belt detection head may be implemented to detect objects using any well-known, conventional method for object detection using the CNN architecture provided by Chun.

**7. [1F] "and a pose estimation prediction head;"**

142. Chun discloses this element.

143. Both the key point and part-affinity field (PAF) prediction heads, as disclosed by Chun, perform "2D driver/passenger pose estimation." EX1005, 1. The CNN was trained using images of drivers performing specific gestures, including "drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id*., 4. These gestures provided additional pose training images to the "other publicly available pose estimation data sets such as MS COCO." *Id*.

144. Using the CNN trained on pose estimation training data, the key point head "predict[s] a pixel-wise probability" that "indicates the probability of the corresponding pixel being a certain joint." The PAF head predicts a pixel-wise probability to "encode pairwise relationships between body joints." As described, "[o]utputs from the first two heads are used to parse the key point instances into human skeletons." *Id*., 5.

145. Thus, Chun expressly discloses a pose estimation prediction head.

**8. [1G] "minimizing a loss based on outputs of the one or more prediction heads causing a change in parameters of the backbone network; and"**

146. Chun in view of Chao renders this element obvious.

147. At the outset, to the extent any claim-construction argument is raised on this point, I am not opining that claim element [1G] requires a "joint loss" or that multiple head-loss terms must be combined into a single scalar objective. In my opinion, the ordinary reading of [1G] requires only that the loss being minimized is "based on outputs of the one or more prediction heads" and that this loss minimization causes a change in the backbone parameters. That language is broad enough to cover (a) loss computed from a single head, (b) loss computed from multiple heads, or (c) loss computed by any aggregation function of the outputs (for example, a sum, weighted sum, or simple selection of one head's loss). My mapping below to TensorFlow-style multi-task training reflects one conventional implementation path, not a narrowing view that the claim requires minimizing "joint loss." Even a single-head loss feeding gradients into the backbone satisfies the plain language of [1G].

**(a) Chun**

148. Although Chun does not explicitly describe minimizing a loss function, its training setup, network architecture, and implementation details strongly suggest that such loss minimization was employed and certainly this is true in view of Chao.

149. Chun discloses that its model, NADS-Net, was implemented and trained in TensorFlow, using a multi-task convolutional neural network composed

of an FPN backbone and multiple detection heads for key-point, part-affinity, and seat-belt segmentation tasks. Specifically, Chun uses TensorFlow for training a multi-head architecture with a shared backbone and this would suggest to a POSITA that training involves a loss function for optimization and accuracy. In a multi-stage architecture, this may include combining the individual losses from the prediction heads (*e.g.*, key point, PAF, and seat belt prediction heads) into a total loss (*i.e.*, joint loss function). In turn, the CNN would minimize the loss of the total loss by updating the parameters of the backbone. To calculate an individual loss, each prediction head generates an output, which is compared to the ground truth. The total loss is the sum of these individual losses. State-of-the-art approaches minimize this combined loss by adjusting network parameters of the backbone through methods like backpropagation, which sends errors from all of the prediction heads to the backbone, and, in response, the backbone updates its weights to learn features for all tasks together.

150. Consistent with this understanding, Chun repeatedly references training, accuracy, and performance metrics, explaining that the model was "trained and validated" on labeled images from a newly created driving dataset and that "three different experiments" were performed to compare training strategies: (1) trained only on MS COCO, (2) trained only on the new driving dataset, and (3) "pretrained on MS COCO dataset and then transferred to the new driving dataset"

*Id*., 5-7. A POSITA would recognize that these experiments directly reflect standard supervised optimization procedures in CNNs, each variant corresponding to a distinct training-loss configuration. The very distinction between "trained only," "pretrained," and "transferred" implies iterative weight updates through backpropagation based on a defined loss function.

151. Moreover, Chun's architectural description reflects a joint optimization design. The reference explains that its FPN backbone "produces a rudimentary feature pyramid for the later detection branches" and that each detection head performs convolutions to generate probability maps, after which the outputs of these branches are concatenated to form the network's final prediction tensor. *Id*., 5. A POSITA would recognize that concatenating branch outputs implies a shared gradient path, meaning errors from all heads are propagated back to the shared backbone during training. In multi-task CNNs, such architectures are trained with joint loss functions combining individual task losses (*e.g.*, key-point L2 loss, segmentation cross-entropy, and affinity field loss) weighted to balance learning across tasks. This is the same joint-loss structure used in Faster R-CNN and Mask R-CNN, both of which Chun cites.

152. Further, Chun explicitly reports that its multi-branch configuration "replaces multiple refining stages of generic poseestimation algorithms" while maintaining or improving accuracy. *Id*., 6. A POSITA would understand this as

evidence of joint end-to-end training, since the ability to eliminate post-refinement depends on the backbone learning to optimize all tasks simultaneously through a unified loss objective.

153. Taken together, Chun's (1) TensorFlow-based implementation, (2) description of supervised, labeled training data, (3) multi-branch architecture with concatenated outputs, and (4) comparison of pretrained and transfer-learning regimes, all strongly suggest that its CNN was trained by minimizing a joint loss function derived from the outputs of the prediction heads, thereby updating the parameters of the shared backbone network.

154. While Chun strongly suggests that its multi-head CNN was trained by minimizing a joint loss function derived from the outputs of its prediction heads, it does not provide the explicit implementation details of that optimization process. A POSITA would have understood, however, that implementing such a loss-minimization routine was conventional to achieve the multi-task training results Chun reports. Given Chun's TensorFlow-based implementation, labeled training data, and multi-branch architecture, a POSITA seeking to understand or implement the underlying training mechanism would have looked to references such as Chao, which explicitly discloses those well-known implementation details.

**(b) Chao**

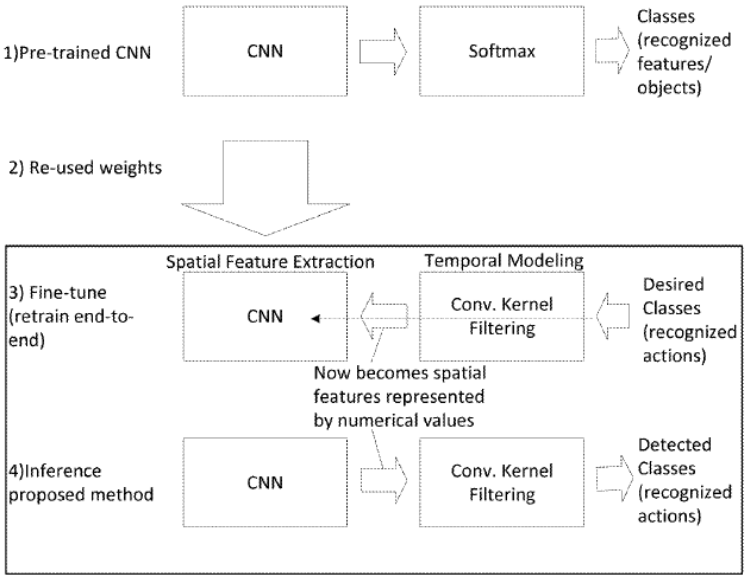
155. Chao discloses systems, methods, and devices for triggering vehicular actions "based on the monitoring of one or more occupants of a vehicle." EX1006, Abstract. The primary goal of the system is to determine "states" of occupants to trigger these vehicular actions. To classify these states, Chao discloses a modular architecture comprising a CNN that acts as a backbone network and a spatio-temporal (ST) model or temporal model that act as prediction heads coupled to the CNN.

156. Chao explicitly discloses that the CNN-ST architecture is trained using supervised learning based on objective functions and conventional optimization algorithms. In particular, Chao explains that the CNN-based models are trained "using an objective function" (i.e., optimization algorithms such as loss functions) and "any kind of training principle, such as backpropagation" EX1006, ¶¶[0041], [0046]. During training, the system receives positive or negative feedback based on the outputs of the prediction heads, forming the basis of the loss function EX1006, ¶[0043]. This feedback is used to refine the entire network: "[the] CNN may be configured to be further trained based on the feedback from the results produced by the ST system" *Id.,* ¶[0098]. A POSITA would understand that these statements describe the standard supervised training loop in which prediction head outputs are compared against ground-truth labels, and the resulting

loss is minimized via backpropagation, which propagates gradients backward through the prediction heads and into the CNN backbone to update its parameters.

157. FIG. 8A illustrates and confirms this process:

FIG 8A




**EX1006, FIG. 8A**

158. As illustrated, a "Pre-trained CNN" has its "re-used weights" fed into a subsequent "CNN" (which is the backbone) for "Spatial Feature Extraction," which then feeds into "Temporal Modeling" to produce "Desired Classes (recognized actions)." The arrow labeled "3) Fine-tune (retrain end-to-end)" explicitly shows the entire pipeline, including the initial CNN (backbone), being retrained based on the desired outputs. "End-to-end" retraining means that the

parameters (weights and biases) of all layers, including those of the backbone network, are adjusted during this process based on the computed loss from the final output.

159. Additionally, Chao states that the 2D CNN "may change the output of the 2D CNN 802 to, instead of representing the probability of the pre-trained classes, to become a "set of features" represented by values for the new desired action classes." *Id*., ¶[0099]. This emphasizes that the training process, driven by the "desired 'action classes'" (outputs from the prediction heads), directly causes the backbone's behavior and, consequently, its internal parameters to change to achieve these new desired outputs.

160. A POSITA would have understood that Chao's "feedback from the results produced by ST system" to "train the CNN" and "backpropagation" discloses that gradients of the loss function, calculated from the final outputs (of the prediction heads), are propagated backward through all layers of the network, including the backbone, to update their parameters. *Id*., ¶¶[0046],[0098].

161. Therefore, Chao discloses that the neural network architecture is trained by using the outputs of its prediction heads (*e.g.*, action class scores) against desired outputs as feedback, and this process, involving techniques like backpropagation and end-to-end fine-tuning, causes the parameters of the backbone network (the 2D CNN) to change.

162. Thus, Chao provides explicit disclosure of conventional deeplearning practice that prediction head outputs feed into a loss function, and backpropagation updates both prediction head and backbone parameters. Chao confirms this, both textually and via FIG. 8A.

**(c) MTC**

163. Due to the nature of state-of-the-art approaches to multi-task CNN training, it would be obvious to a POSITA that integrating the loss minimization of Chao into Chun's architecture would be a conventional practice in training and optimizing a neural network. The motivation, as before, would be to enhance the system's ability to infer driver state while leveraging the well-known and conventional optimization framework suggested in Chun's disclosure and taught explicitly by Chao.

164. Both Chun and Chao operate within analogous multi-task convolutional architectures that use a shared backbone and prediction heads for driver state detection. Chun already teaches training multiple heads for pose estimation and seat belt detection utilizing well-known and conventional training methods (*i.e.*, utilizing TensorFlow, EX1005, 5), while Chao demonstrates that a distraction classification head can be trained by minimizing a loss function based on feedback from the prediction head and propagating its gradients through the shared backbone to optimize shared parameters.

165. Combining Chao's explicit loss function for the CNN model with Chun's end-to-end driver-monitoring network would have been a straightforward engineering step, producing exactly what Chun implies but doesn't detail: a driver-state recognition model that achieves "optimal performance" for distraction classification, object detection, and pose estimation in a vehicle's runtime environment. EX1005, 1.

166. Element [1G] does not reflect a technical breakthrough and the combination of Chun and Chao is nothing more than the routine combination of familiar elements in the field. A POSITA would have understood that Chun already disclosed an end-to-end multi-task classifier that jointly trained backbone and prediction head parameters to recognize driver distraction attributes such as distracted poses or improper use of seat belts. Chao, in turn, disclosed the standard practice of minimizing loss using the output of a prediction head to update parameters of the backbone. Bringing these two teachings together involves no inventive step; it simply unites familiar elements (*i.e.*, jointly trained models and joint loss functions) using conventional methods to achieve the expected outcome of enabling inference in a vehicle system. The result is entirely predictable: the classifier is optimized to perform with better accuracy, as both references contemplate.

167. Moreover, the application of loss minimization techniques to Chun's CNN model represents the straightforward use of a known technique to improve a closely related system. By the relevant date, it was well understood that for any trained neural network, re-training weights and parameters based on the output was essential and routine. Applying Chao's explicit loss function disclosure to Chun's model does not alter the operation of either reference but merely extends an established technique to a nearly identical context: a driver-monitoring perception network trained to classify distractions, detect objects, and estimate driver pose. A POSITA would have recognized that this conventional step would improve Chun by ensuring its trained parameters were optimized for increased accuracy, consistent with Chun's own emphasis on improving distracted driving detection for in-vehicle use. EX1005, 1.

168. Updating parameters of a neural network based on the output to increase neural network accuracy was a technique that had already been used across a range of machine-learning models and devices. A POSITA would naturally apply the same technique to Chun's driver state detection model, recognizing it would provide the same benefits, namely, optimized weights and improved accuracy, without requiring any inventive insight or leap. In other words, the claimed step reflects nothing more than applying an established solution to a system that was already ready for that improvement.

169. A POSITA would have reasonable expectation of success when implementing the loss minimization of Chao in Chun's CNN model. As loss functions and backpropagation are common and conventional techniques of multi-stage deep learning training, integration of Chao's loss function would not require redesign of Chun's backbone or prediction heads, or novel training methods. Rather, a POSITA would understand that Chao is describing an element that is suggested by Chun's CNN model. At the relevant time, deep-learning frameworks such as TensorFlow were the dominant tools used by practitioners to develop and deploy neural networks. The frameworks provided standardized, built-in mechanisms for minimizing loss for trained models that optimize and improve accuracy by updating parameters based on the output of the CNN model. A POSITA would have been well aware that these serialization features made it straightforward to minimize loss for inference. In practice, these frameworks enabled the deployment of loss functions while training neural networks, including backbone and prediction head components, without bespoke engineering.

170. The combination does not require adapting incompatible architectures or inventing new storage techniques. Both references rely on the same class of neural network models, and the step of minimizing loss is not expressly different for each specific task (distraction classification in Chao or pose estimation and object detection in Chao). As such, a POSITA would have expected that

minimizing the loss of Chun's trained backbone and head weights in the manner explicitly taught by Chao would predictably yield an optimal driver detection model. The result would simply be the ordinary outcome of applying a well-established technique to a closely related use case.

171. In conclusion, both systems operate within the same domain of prediction for autonomous or driver-monitoring applications and employ conventional convolutional backbones with task-specific heads. In view of these aligned contexts and compatible components, a POSITA would have found it obvious to perform loss minimization on Chun's distraction-classification model, including updating the backbone parameters based on the outputs of the prediction heads.

**9. [1H] "storing a distraction classification model after minimizing the loss, the distraction classification model comprising the parameters of the backbone network and parameters of at least one of the one or more prediction heads."**

172. Chun in view of Chao renders this element obvious.

**(a) Chun**

173. While Chun does not expressly state that the trained model is stored, its teachings would require that the trained CNN architecture, NADS-Net, is saved and deployed for inference once training and validation were complete. Chun presents NADS-Net as a complete, deployable CNN designed for in-vehicle

monitoring tasks, describing it as "more nimble and thus more suitable for in-vehicle monitoring tasks" and capable of "non-invasive, non-obstructive means to monitor and detect in-cabin activities." EX1005, 1. From these statements, a POSITA would understand that Chun envisions the trained model being implemented in production IVMS systems and that the trained parameters of both the backbone and the detection heads must be stored for subsequent inference.

174. Chun also describes the full training and evaluation lifecycle of NADS-Net: it was "trained and validated" using supervised, annotated image data and evaluated under three training configurations— "only on MS COCO dataset," "only on the new driving dataset," and "pretrained on MS COCO dataset and then transferred to the new driving dataset." *Id.,* 7. These experiments reflect distinct trained versions of the same network. A POSITA would understand that each version represents a finalized model with a corresponding set of learned parameters which a POSITA would want retained for comparison, validation, and later deployment.

175. Where Chun suggests this step but omits implementation detail, Chao provides explicit teaching that would have made such storage obvious to a POSITA.

**(b) Chao**

176. Chao supplies the explicit disclosure of storing the trained model, including both backbone and prediction head parameters. Chao expressly states that "[a] memory of the one or more memories 104 may store … a trained system, such as a neural network, or a deep neural network, for example." EX1006, ¶[0054]. This memory is described to be integrated in a vehicle comprising a control system. *Id*., ¶[0049]. As described in element [1G], Chao provides "feedback from the results produced by ST system" to "train the CNN" and "backpropagation" indicates that gradients of the loss function, calculated from the final outputs of the prediction heads, are propagated backwards through all layers of the network, including the backbone, to update their parameters. *Id*., ¶¶[0046],[0098]. It would be obvious to a POSITA that the updated CNN and prediction head would be stored in the memory as explicitly disclosed in Chao.

**(c) MTC**

177. A POSITA would have found it obvious to combine the complementary teachings of Chun and Chao to store a distraction-classification model (including Chun's pose estimation prediction head, object detection head and Chao's distraction classification head) after minimizing the loss. A POSITA reading Chun and Chao would understand that once the network is trained and converged, its learned parameters (backbone and prediction heads, including a

distraction classification head) must be persisted, *i.e.*, saved to non-volatile memory, to actually run in a real-time driving system. Chao provides a clear example of how that saving step is done: storing the trained model in system memory so it can later be loaded and executed in a vehicle. Combining Chao's explicit storage for the CNN model with Chun's end-to-end driver-monitoring network, storing NADS-Net would have been a straightforward engineering step, producing exactly what Chun implies but doesn't detail: a fully deployable driver-state recognition model capable of performing distraction classification, object detection, and pose estimation in a vehicle's runtime environment.

178. Element [1H] does not reflect a technical breakthrough and the combination of Chun and Chao merely represent the routine combination of familiar elements in the field. A POSITA would have understood that Chun already disclosed an end-to-end multi-task classifier that jointly trained backbone and prediction head parameters to recognize driver distraction attributes such as distracted poses or improper use of seat belts. Chao, in turn, disclosed the standard practice of storing a trained backbone-plus-head model in memory for inference. Bringing these two teachings together involves no inventive step; it simply unites familiar elements (*i.e.*, jointly trained models and storage of trained weights) using conventional methods to achieve the expected outcome of enabling inference in a

vehicle system. The result is entirely predictable: the classifier can be deployed for runtime use once stored, as both references contemplate.

179. Moreover, the application of model-storage techniques to Chun's CNN model represents the straightforward use of a known technique to improve a closely related system. By the relevant date, it was well understood that for any trained neural network intended for deployment, persisting the trained weights in memory was essential and routine. Applying Chao's explicit storage disclosure to Chun's model does not alter the operation of either reference but merely extends an established technique to a nearly identical context: a driver-monitoring perception network trained to classify distractions, detect objects, and estimate driver pose. A POSITA would have recognized that this conventional step would improve Chun by ensuring its trained parameters were available for immediate inference, consistent with Chun's own emphasis on a nimble architecture for in-vehicle use. EX1005, 1.

180. Storing trained parameters to enable deployment was a technique that had already been used across a range of machine-learning models and devices. A POSITA would naturally apply the same technique to Chun's driver state detection model, recognizing it would provide the same benefits, namely, persistence of optimized weights and reliable runtime execution, without requiring any inventive insight or leap. In other words, the claimed step reflects nothing more than

applying an established solution to a system that was already ready for that improvement.

181. A POSITA would have had a reasonable expectation of success in combining Chun with Chao because both references describe techniques that were already well understood, fully compatible, and routinely implemented in practice. Chun discloses a conventional multi-task neural network with a backbone and multiple prediction heads, trained end-to-end until convergence. Chao discloses the common practice of persisting trained model parameters, including both backbone and head weights, in memory for later inference. At the relevant time, deep-learning frameworks such as PyTorch and TensorFlow were the dominant tools used by practitioners to develop and deploy neural networks. Both frameworks provided standardized, built-in mechanisms for storing trained models, such as checkpoints in TensorFlow and state dictionaries in PyTorch, that captured the complete set of learned parameters. A POSITA would have been well aware that these serialization features made it straightforward to save a trained network in its entirety and later reload it into memory for inference. In practice, these frameworks enabled trained networks, including backbone and prediction head components, to be stored as a single artifact and deployed without bespoke engineering. Thus, the use of such frameworks confirms that saving and reloading trained models was not an inventive step, but rather a routine, well-understood

practice at the time, as shown in Christoph Maurer, *Integration and Deployment of Machine Learning Models* (Master's Thesis, Graz University of Technology, Feb. 2021) (EX1039).

182. The combination does not require adapting incompatible architectures or inventing new storage techniques. Both references rely on the same class of neural network models, and the step of storing the optimized parameters is independent of the specific task (distraction classification in Chao or pose estimation and object detection in Chun). As such, a POSITA would have expected that saving Chun's trained backbone and head weights in the manner explicitly taught by Chao would predictably yield a working deployed model, ready for inference in runtime environments. The result would simply be the ordinary and reliable outcome of applying a well-established technique to a closely related use case.

183. Neither reference teaches away from storing the trained model. On the contrary**,** Chun's post-training testing and validation suggests saving the optimized model for deployment, while Chao's disclosures on storing trained machine-learned models are expressly applicable to architectures comprising a backbone and one or more prediction heads. Both systems operate within the same domain of prediction for autonomous or driver-monitoring applications, employ conventional convolutional backbones with task-specific heads, and share a common software-

hardware workflow in which a model is trained on one system and deployed on another (*i.e.*, a vehicle). In view of these aligned contexts and compatible components, a POSITA would have found it obvious to store the converged distraction-classification model, including both the backbone parameters and the parameters of the selected prediction head(s), for deployment and inference.

### C. Claim 2: The method of claim 1, wherein the distraction classification prediction head includes a convolutional layer, pooling layer, and fully connected layer.

184. Chun in view of Chao renders this claim obvious.

#### 1. Chun

185. While Chun does not expressly disclose a distraction classification head, Chun expressly discloses one or more prediction heads comprising at least 3 convolutional layers coupled to a backbone network, as illustrated in Figure 3.

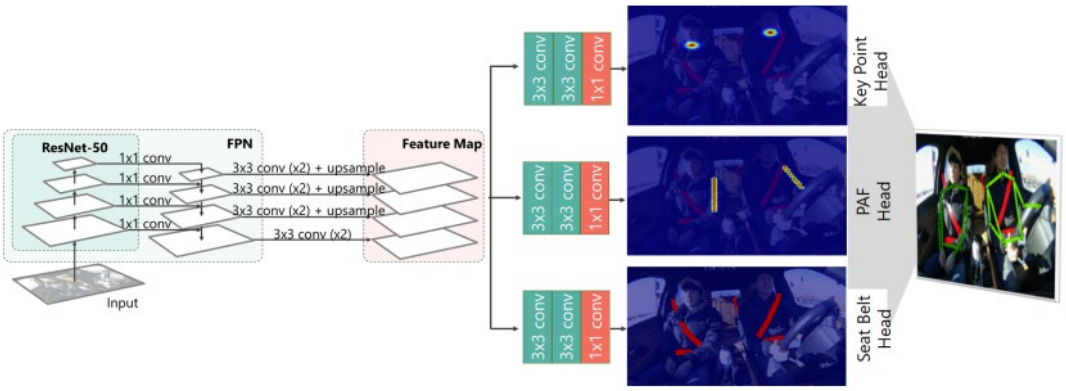



Figure 3: NADS-Net architecture.

**EX1005, FIG. 3**

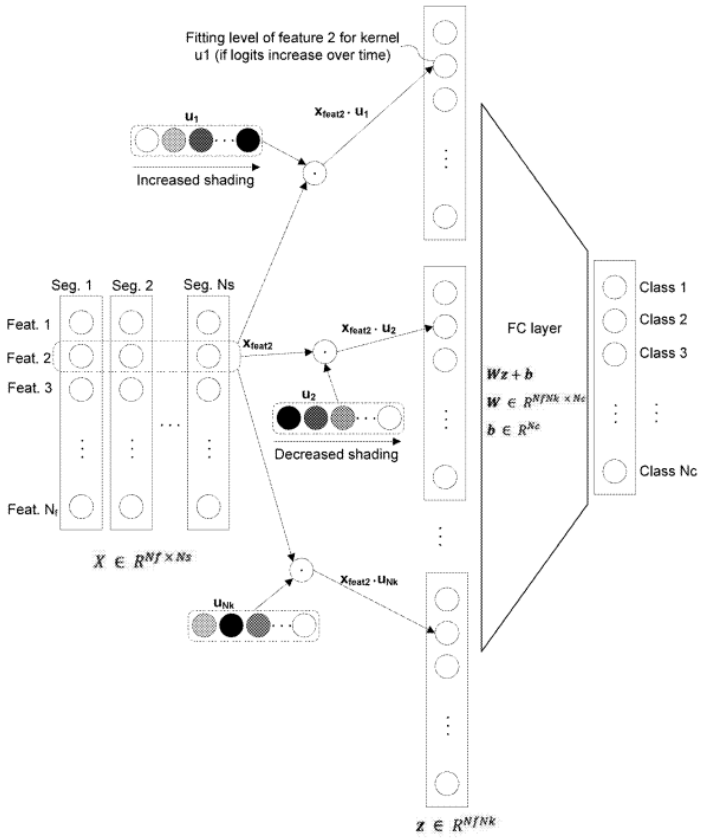
186. As described, "[e]ach of the detection branches employs two 3×3 convolutions and a 1×1 convolution to predict a pixel-wise probability distribution." For the key point detection head, the pixel-wise probability "indicates the probability of the corresponding pixel being a certain joint," the part-affinity head "encode[s] pairwise relationships between body joints," and the seat belt detection head "generates a binary seat belt detection mask" based on "the likelihood of each pixel being a seat belt." EX1005, 5. These components are all directly coupled to the shared feature map output of the backbone network, demonstrating multiple prediction heads that generate distinct pixel-level outputs.

187. While Chun does disclose convolutional layers, it does not expressly disclose a pooling layer and fully connected layer. However, a POSITA would have understood that at the time of the priority date, convolution → pooling → fully connected was the conventional classifier-head pattern in multi-task CNNs. Pooling layers were routinely inserted after the final convolution to aggregate spatial features (*e.g.*, global average pooling) before passing to a dense layer for classification. Fully connected layers then map the pooled features to class scores via SoftMax or sigmoid outputs. Because Chun's FPN backbone already produces multi-resolution feature maps, a POSITA would have recognized that adding a classification branch with a convolutional layer, a pooling operation (global average or max), and a fully connected SoftMax layer was a routine

implementation of a prediction head for categorical tasks such as distraction classification. This understanding is confirmed in view of Chao.

**2. Chao**

188. Chao discloses a modular architecture comprising a CNN that acts as a backbone network and a spatio-temporal (ST) model or temporal model that act as prediction heads coupled to the CNN. The CNN is described as "a base model that identifies and extracts spatial features and/or objects from frames of a video." EX1006, ¶[0099]. The ST model (analogous to the claimed distraction classification prediction head) uses the features extracted by the CNN to perform tasks such as classifying driver distraction, detecting objects, and inferring aspects of occupant pose to determine states and trigger vehicular actions. The ST model is illustrated in Figure 9.




**EX1006, FIG. 9**

189. As shown, the ST model used for determining classes, or states, of a driver comprises multiple convolutional layers. A **convolutional layer** uses a kernel to detect features in image data. These kernels are matrices of weights learned during training and at each position in the matrices, the layer computes a dot product between the weight and the input it is currently over. As described in Chao, the first convolutional layer "convert[s] the absolute feature logits from the identified objects and/or features extracted from the 2D CN model to provide an indication of increasing or decreasing values." *Id.*, ¶[0106]. In Figure 9, Chao

describes an example for performing this layer for Feature 2 wherein the "dot product of $x_{feat2}$ and $u_1$ provides a fitting level for kernel $u_1$ if the logits increase over time and is included in the corresponding Feat. 2 entries of the indicator sets." *Id*., ¶[0109].

190. While Chao does not expressly use the term "pooling layer", a POSITA would understand that a pooling layer reduces the spatial dimensions of a feature map to decrease computational load. Chao provides a layer in the ST model that "simplifies the FC layer by providing a number of kernels, $N_k$, less than the number of segments, $N_S$, *i.e.*, $N_k<N_S$, with a reduction of the dimensions of the feature vector input to the FC layer," which is functionally performing the same spatial reduction as a pooling layer. *Id*., ¶[0106].

191. Lastly, Chao describes that the feature vector is input to a fully connected (FC) layer to "provide a vector containing values…to use in determining action scores for each of the classes which identify the one or more passengers' action (e.g., if the driver is at full attention)." *Id*., ¶[0107]. This final FC/SoftMax block directly outputs the driver's distraction state.

192. Accordingly, Chao's ST classification head implements all three claimed layer types within a single head: convolutional for feature refinement, pooling for dimensional aggregation, and fully connected for class decision;

mirroring the standard classifier-head structure used in multi-task CNN frameworks and as required by claim 2.

**3. MTC**

193. Claim 2 specifies specific layers of the distraction classification prediction head. As discussed above for claim [1D], Chun provides a framework for implementing a multi-stage CNN architecture for distracted driving detection, including prediction heads comprising multiple convolutional layers. Chao simply provides a prediction head for distraction classification that includes disclosure of specific convolutional layers, namely a convolutional layer, pooling layer, and fully connected layer. A POSITA would have readily recognized that combining these disclosures yields exactly the claimed feature as both references teach modular, multi-task CNN networks trained for driver state detection using prediction heads having a plurality of convolutional layers. A POSITA would have been motivated to implement the distraction head of Chao within Chun's multi-head CNN because the conv-pool-FC pattern was the conventional design for classification branches attached to shared feature maps (*e.g.*, global average pooling followed by a dense SoftMax classifier). Integrating that sequence into Chun's architecture would have been technically straightforward and predictable.

194. **How:** A POSITA would have implemented the distraction-classification prediction head of claim 2 by incorporating the layer sequence

expressly shown in Chao (a convolutional layer for feature refinement, a pooling layer for dimensionality reduction, and a fully connected layer for generating class scores) into the multi-head CNN framework of Chun. In practice, this would involve attaching Chao's classifier branch to Chun's shared FPN feature map so that the convolutional layer processes extracted spatial features, the pooling layer aggregates them, and the fully connected layer outputs categorical distraction states such as "texting," "drinking," or "looking away."

195. **Why:** A POSITA would have been motivated to combine these teachings because both references address the same technical objective (*e.g.*, driver-state recognition within a multi-task CNN) and employ compatible modular architectures. By the 2018-2019 timeframe, the conv → pool → FC sequence was the conventional design for classification heads in multi-task CNNs. Incorporating this standard head into Chun's existing architecture would have improved its capability to classify distraction states while reusing the same backbone features already optimized for pose and object detection, thereby enhancing efficiency without architectural redesign.

196. **Reasonable Expectation of Success:** A POSITA would have reasonably expected this integration to succeed because Chun's backbone produces multi-scale feature maps encoding facial, hand, and pose cues, which is the same information used by Chao's classification head**.** Attaching a conv-pool-FC branch

to those features was a routine and predictable engineering step that would yield accurate categorical predictions of driver distraction. The resulting configuration merely applies well-understood CNN components according to their established functions, producing no unexpected results.

197. **Accordingly,** for the same reasons articulated above in Ground 1, Chun in view of Chao renders claim 2 obvious, as the combination predictably yields a distraction-classification prediction head comprising a convolutional layer, pooling layer, and fully connected layer.

### D. Claim 6: The method of claim 1, further comprising inputting image data into the distraction classification model and receiving a distraction classification tag from the distraction classification model based on the image data.

198. Chun in view of Chao renders this claim obvious.

199. At the outset, for clarity, I am not construing Claim 6 as requiring that the "distraction classification model" includes a distraction-classification head in all embodiments. Claim 1 and Claim 6 read broadly enough that a model with a single prediction head performing the classification could satisfy the claim. My analysis below simply explains that, where a model lacks an explicit prediction head that outputs a categorical distraction label, a POSITA would have understood it to be routine to add such a head. Chun discloses the shared-backbone multi-head CNN framework that ingests image data and produces task outputs. Chao supplies

an explicit classification branch that outputs the categorical distraction label required by Claim 6 ("texting," "drink," etc.). For that reason, even though Chun strongly suggests the same result, integrating Chao's classification branch into Chun's architecture is the obvious path that yields the claimed classification tag.

**1. Chun**

200. A POSITA would understand that Chun discloses or at minimum strongly suggests the inference-time process of inputting image data into a trained neural-network model and receiving corresponding output predictions. Chun teaches a specific CNN architecture for an IVMS that includes multiple detection heads for "optimal performance for driver/passenger state detection tasks," that is "more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches." EX1005, 1. Chun explicitly teaches that image data captured from an in-vehicle driver-facing camera is annotated and input into a multi-task neural network model for both training and testing. As described, "[f]or the training data, 30 driving sessions out of total 50 were used. The rest were reserved for testing." *Id*., 5. Although Chun primarily discusses training and testing, a POSITA would understand that the same trained model executes an inference phase in operation, *e.g.*, receiving real-time camera images as input and producing per-task outputs such as pose estimation or seat-belt detection. As described, "[f]or the training data, 30 driving sessions out of total 50 were used. The rest were reserved for

testing." *Id*., 5. Because the architecture is identical in both phases, the act of inputting driver images and obtaining a predicted output (pose, seat-belt mask, etc.) performs the same type of input/output flow recited in claim 6.

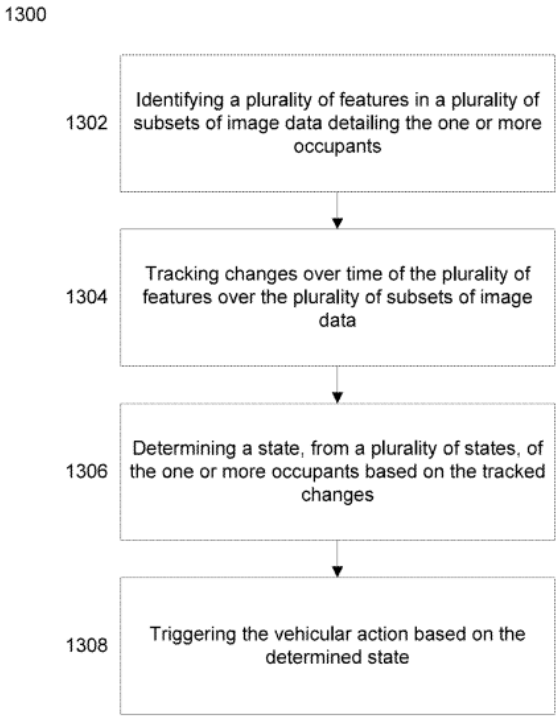
201. Importantly, Chun discloses that the CNN was trained using data of drivers performing distracted driving behaviors, noting that during data collection, "the subjects were requested to perform certain gestures and motions such as drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id*., 4. These activities correspond directly to various distraction types (manual, visual, cognitive) recognized in the IVMS literature and represent labeled examples of distracted driving states. It would thus be understood that the testing data is input into the backbone network to generate a feature map which would in turn be used to generate a pose estimate or seat belt detection. Thus, Chun supplies both the functional context and the operational mechanism (receiving image data, producing task-specific results) from which adding a distraction-classification tag would have been an obvious extension, as expressly taught in Chao.

**2. Chao**

202. Chao expressly discloses feeding image data into a CNN-based driver-monitoring model and outputting a categorical tag identifying the driver's

distraction state. Similar to Chun, Chao describes a multi-stage neural network that provides "inputs, based on the plurality of subsets of image data, to a two-dimensional convolutional neural network (2D CNN)." EX1006, ¶[0136]. Using these inputs, Chao describes a distraction classification head that provides a classification, or tag, of a driver's distraction state attached to a shared backbone (*i.e.*, 2D CNN), similar to the shared backbone of Chun. As shown in FIG. 13, the system receives image data that details one or more occupants and triggers a vehicular reaction based on a "determined state" of the one or more occupants of a vehicle. EX1006, ¶[0129].

FIG 13




**EX1006, FIG. 13**

203. To determine states and trigger vehicular actions, the ST model uses the features extracted by the CNN to perform tasks such as classifying driver distraction, detecting objects, and inferring aspects of occupant pose. To perform these tasks, a SoftMax classification layer is "applied to the Spatio-temporal features output" of the ST model and each of the classes of the vector produced by the SoftMax layer "is provided as a value, or 'Action Score,' indicating the determined class based on the input." Thus, the ST model outputs a class, or tag, that "may correspond to a status of the driver." *Id*., ¶[0101].

204. Table 1 lists exemplary "classes (i.e., states)" such as "Drive both hands," "Switch gear," "Talk to passenger," "Adjust radio," "Drink," "Call," "Look right," "Makeup," "Read," and "Text". *Id*., ¶[0113].

TABLE 1

Exemplary list of classes (i.e. states) and corresponding reaction time values

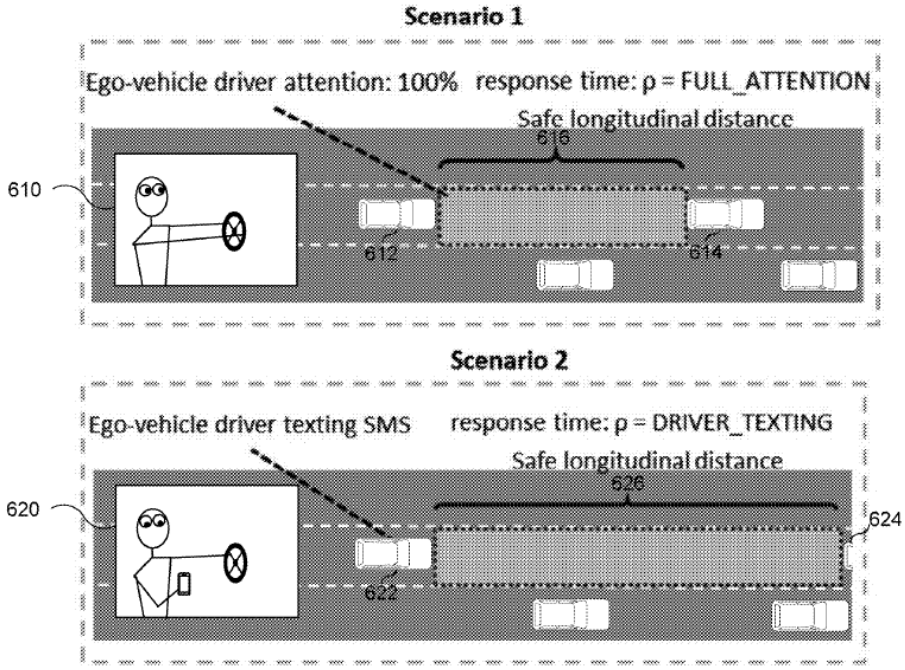
| | Index | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Class | Drive both hands | Switch gear | Talk to passenger | Adjust Radio | Drink | Call left hand | Call right hand | Hair makeup | Reach side | Reach behind | Text left hand | Text right hand | Hands free |
| Reaction time | 2.0 sec | 2.1 sec | 2.4 sec | 3 sec | 3 sec | 3.2 sec | 3.2 sec | 3.5 sec | 4 sec | 4 sec | 4.2 sec | 4.2 sec | 5 sec |

**Chao, Table 1**

205. Figure 6 of Chao provides an example illustrating "Ego-vehicle driver attention: 100% full attention" versus "Ego-vehicle driver texting SMS", which

indicates a specific classification, or tag, of driver distraction. Specifically, scenario 1 tags the driver attention as "FULL_ATTENTION" and scenario 2 tags the driver attention as "DRIVER_TEXTING," as shown below.

FIG 6




**EX1006, FIG. 6**

206. Chao therefore squarely teaches a system for inputting image data into a distraction classification model and receiving a distracted driver tag for the image data using a multi-stage CNN architecture. A POSITA would have understood that Chao's class determination represents a classification tag because (1) it is a binary categorical prediction identifying whether the driver is or is not performing a distracting activity, and (2) in CNN practice, these results were presented as

classification labels or tags for downstream decision modules. In Chao, the class with the "highest score" is selected to yield a discrete prediction (e.g., "texting" or "full attention"), which a POSITA would interpret as a classification tag. *Id*., ¶[0153]. Although Chao does not explicitly use the term "tag," its binary, per-task categorical outputs function as such, indicating the driver's detected distraction state. A POSITA would further understand that these outputs are received by the vehicle's driver-state monitoring system for subsequent alert generation or vehicle action, thereby satisfying the claimed step of "receiving a distraction-classification tag from the model."

**3. MTC**

207. **How:** A POSITA would have implemented the claimed step by integrating Chao's distraction-classification head into Chun's existing multi-head CNN. At inference time, the vehicle's camera would provide live driver-image frames as input to Chun's shared backbone. The extracted feature maps would then be routed to Chao's SoftMax-based classifier branch, which outputs a distraction-classification tag (*e.g.*, 'Texting,' 'Drinking,' 'Hands on Wheel'). The tag is then received by the in-vehicle monitoring subsystem for alerting or control logic. This process directly satisfies both "inputting image data … into the model" and "receiving … a distraction-classification tag."

208. A POSITA would have recognized that the teachings of both Chun and Chao provide conventional, well-understood methods for classifying a distracted driver. This is because by 2019, "backbone" was standard machine learning parlance (*e.g.*, ResNet, VGG, MobileNet serving as "backbones" in detection/segmentation pipelines), which is further exemplified by Chun's discussion of state-of the-art approaches to driver state detection tasks.

209. Chun already describes a multi-task driver-state recognition system that captures driver images, inputs them into a shared convolutional backbone, and produces "driver/passenger state detection tasks" for multiple tasks, including pose estimation and seat belt detection. EX1005, 1. The backbone generates shared feature maps that feed into task-specific attention heads. Chao discloses a nearly identical backbone-to-head architecture for driver state detection tasks: inputting image data, generating shared convolutional features, and routing them to parallel heads, including a classification branch that outputs a class label for each detected instance ("a value, or 'Action Score,' indicating the determined class based on the input." EX1006, ¶[0101].) A POSITA would recognize that both systems follow the same modular CNN paradigm: shared feature extraction followed by specialized prediction branches.

210. Because Chun already extracts features from and estimates a driver's pose and seat belt detection based on images of distracted driving gestures "such as

drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc.," EX1005, 4, adding Chao's driver distraction classification that detects "classes (i.e., states)" such as "Drive both hands," "Switch gear," "Talk to passenger," "Adjust radio," "Drink," "Call," "Look right," "Makeup," "Read," and "Text," EX1006, ¶[0113] would be a simple and conventional extension to Chun's driver state detection model.

211. The addition would have been technically straightforward: both architectures employ a shared backbone feeding separate prediction heads, and Chao's ST model for driver state classification is expressly designed as a modular branch that receives extracted features as input. Because Chun's backbone produces feature maps of the driver's pose and seat belt donning, a POSITA would have recognized that the same feature maps could serve as input to a distraction classification head without altering the network's fundamental design.

212. Since Chun already receives input image data to perform driver state detection tasks that provide an annotation, or label, for a driver's pose and seat belt donning and Chao already receives input image data to provide a classification label of a driver's state detection task, it would be clear that implementing Chao's driver distraction classification head into Chun's architecture would be a simple

technological addition of a head into the neural network to receive a distraction classification tag.

213. **Why:** A POSITA would have been motivated to combine these teachings to produce interpretable, real-time outputs from the driver-monitoring CNN. Chun's stated objective is to "achieve the optimal performance for driver/passenger state detection tasks," as a "majority of vehicle accidents reported worldwide are caused by distracted driving behaviors," including the "use of smartphones/mobile devices, smoking tobaccos, engaging in a conversation with other passengers, drinking beverages, eating foods, and such, that are irrelevant to the task of driving itself." EX1005, 1. Chao explicitly provides a model that provides "classes (i.e., states)" such as "Drive both hands," "Switch gear," "Talk to passenger," "Adjust radio," "Drink," "Call," "Look right," "Makeup," "Read," and "Text". EX1006, ¶[0113]. Producing explicit categorical tags for each task as described in Chao directly supports the goal of Chun by detecting the specific drivers' states described in Chun that could cause vehicle accidents.

214. By the time the '243 patent was filed, systems that used convolutional neural networks (CNNs) to monitor drivers were already producing simple, easy-to-read results, like tags saying "distracted" or "hands off wheel." *See* corroborating evidence EX1032; EX1033. A POSITA would therefore have been motivated to include Chao's class-label output feature in Chun's model. Doing so

would make the system's predictions of a driver's state clearer and allow a vehicle to make quick decisions based on the driver's state to avoid vehicle accidents. Since Chao's task heads already provide a driver state classification output, adding that step to Chun would be a straightforward design improvement, not a major change or new concept.

215. **Reasonable Expectation of Success:** Both Chun and Chao employ convolutional backbones configured for real-time image inference. Chun's NADS-Net architecture includes a ResNet-50 FPN backbone that produces multi-resolution feature maps and feeds multiple prediction heads for key-point, part-affinity, and seat-belt detection. Chao similarly describes a CNN backbone coupled to a spatio-temporal (ST) classification model that receives video frames from an in-vehicle camera and outputs driver-state classes such as "paying attention," "texting," or "drinking" through a SoftMax classification stage. A POSITA would recognize that these architectures share identical inference mechanics: each takes driver image data as input, extracts spatial features, and outputs task-specific predictions, making integration technically straightforward. The distraction-classification head of Chao could thus be attached to Chun's shared feature maps and trained using the same annotated driver images already used for Chun's pose and seat-belt heads, without redesigning the backbone or data pipeline.

216. Further, adding a classification head to Chun's FPN backbone would have been a routine extension of a known multi-task learning paradigm by 2019. Multi-head CNNs (*e.g.*, Mask R-CNN, FPN) commonly employed shared feature extractors feeding specialized heads for detection, segmentation, and classification. Because Chun's backbone outputs dense feature maps encoding face, hand, and body-pose geometry, precisely the features used by Chao to infer distraction, a POSITA would have expected predictable success in training a new head to output categorical distraction labels. The modification would require only the addition of a soft-max classification branch and labeled examples already available in Chun's dataset. No redesign of the backbone, loss functions, or data pipeline would be necessary.

217. No technical obstacles or incompatibilies would have discouraged a POSITA from making this combination/modification. Chao's distraction classification head is modular and uses standard convolutional layers already employed in Chun. Because both systems use CNNs to perform feature extraction, integration would not require redesign of the backbone or attention modules. A POSITA would have reasonably expected that combining Chao's distraction classification head with Chun's existing pose estimation and object detection heads would yield predictable success, producing a unified model capable of detecting both what the driver is doing (via distraction classification) and how the driver's

body is oriented (via pose estimation), thereby improving recognition accuracy consistent with Chun's objectives.

218. Accordingly, a POSITA would conclude that the combination of Chun and Chao renders Claim 6 of the '243 patent obvious. Chun provides the full framework of a driver-monitoring system that captures image data, processes it through a shared FPN backbone, and produces per-task results, while Chao contributes the explicit class-label output branch and soft-max classification implementation used in analogous multi-head CNNs. Because Chun already defines each head as a convolutional neural network, adopting Chao's explicit class-label output from a CNN would have been a routine, design-consistent step that yields predictable, interpretable results. The resulting system would input driver image data, execute a forward pass through the network, and output categorical distraction tags, such as "texting," "hands free," or "drive both hands", that are received by the driver-monitoring module, thereby meeting the functional and structural requirements of Claim 6.

### E. Claim 7: The method of claim 6, further comprising performing at least one action based on the distraction classification tag.

219. Chun in view of Chao renders this claim obvious.

220. Claim 7 adds the step of performing at least one downstream action based on the distraction-classification tag. Once the model outputs the categorical

label (for example, smoking or phone use), the system then performs some action that uses that tag. That action does not need to be specified yet. The requirement is simply that at least one action is taken because of the tag.

221. As discussed for claim 6, a POSITA would understand that Chun in view of Chao discloses the generation of a distraction classification tag that identifies driver states such as texting and calling. Building upon that foundation, it would have been obvious to a POSITA that the system performs or enables downstream action based on that tag, particularly in light of Chao, which expressly teaches using the outputs of the driver classification to trigger vehicle alerts or actions.

**1. Chun**

222. Chun teaches a specific CNN architecture for an in-vehicle monitoring system (IVMS) that includes multiple detection heads for "optimal performance for driver/passenger state detection tasks." EX1005, 1. IVMS is described as a "standard technology in consumer vehicles as they can play a critical role in preventing and mitigating traffic accidents by alerting the distracted driver and adaptively adjusting the safety mechanisms." *Id*. Chun provides "a new convolutional neural network (CNN) architecture for 2D driver/passenger pose estimation and seat belt detection that is more nimble and, thus, more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches." *Id*.

Although Chun does not detail the alert or control logic, a POSITA would recognize that Chun's architecture supplies the perception output used by a conventional IVMS controller to issue such alerts or safety adjustments, as is the purpose of IVMS.

**2. Chao**

223. Chao discloses a comparable multi-task convolutional architecture that processes image data through a shared backbone and produces multiple outputs, including a classification branch that assigns a class label of a driver's detected state. Chao then utilizes the determined state for "triggering the vehicular action based on the determined state." EX1006, ¶[0129]. The determined state (*e.g.*, "texting," "drinking," "full attention") serves as the decision variable for subsequent control responses.

224. Similar to Chun, Chao provides methods and devices that "use in-vehicle passenger monitoring data" to "provide accurate and quick mechanisms to recognize a driver's state" using "one or more in-vehicle data acquisition devices (e.g. a camera providing RGB, depth, and/or infrared data)." The methods and devices then use the driver's state determined by the "spatio-temporal analysis modeling scheme to trigger vehicular actions," including "providing an appropriate distance from a vehicle located in front or providing a notification to the driver of the vehicle." *Id*., ¶[0022]. In some instances, the determined state may trigger

actions including "modifying one or more of an acceleration, braking, and/or steering of the vehicle or communicating one or more notifications via the UIs." *Id*., ¶[0095]. The notification may include "a visual notification, a sound notification, or a vibration notification." *Id*., ¶[0168]. These disclosures correspond exactly to the claim's requirement of performing at least one action based on the distraction-classification tag.

**3. MTC**

225. **How:** A POSITA would have implemented the claimed step by routing the distraction-classification tag produced by Chao's classification head (as integrated into Chun's multi-head CNN) to the control module described in Chao ¶¶[0095]-[0168]. At runtime, the vehicle's camera supplies image frames to Chun's backbone; the shared features feed Chao's classification head, which outputs a categorical distraction tag. That tag is then used as the conditional input to trigger a driver alert or automatic vehicle-control adjustment, completing the perception-to-action pipeline with standard message-passing or API integration between the CNN and the IVMS controller.

226. As discussed in claim 6, because Chun already extracts features from and estimates a driver's pose and seat belt detection based on images of distracted driving gestures "such as drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and

the center console, adjusting sun visor, and etc.," EX1005, 4, adding Chao's driver distraction classification that detects "classes (i.e., states)" such as "Drive both hands," "Switch gear," "Talk to passenger," "Adjust radio," "Drink," "Call," "Look right," "Makeup," "Read," and "Text," EX1006, ¶[0113] would be a simple and conventional extension to Chun's driver state detection model. The addition would have been technically straightforward: both architectures employ a shared backbone feeding separate prediction heads, and Chao's ST model for driver state classification is expressly designed as a modular branch that receives extracted features as input. Because Chun's backbone produces feature maps of the driver's pose and seat belt donning, a POSITA would have recognized that the same feature maps could serve as input to a distraction classification head without altering the network's fundamental design.

227. As Chun already discusses implementing vision-based sensing technologies in IVMS to trigger alerts and adjust safety mechanisms of a vehicle, the implementation of Chao's determined states triggering vehicular action would be obvious and straightforward as the combination would require only routine signal routing and software integration.

228. **Why:** A POSITA would have been motivated to combine these teachings to achieve a closed-loop driver-monitoring system providing real-time, actionable outputs rather than raw detections. Integrating Chao's action-trigger

logic with Chun's perception model reduces latency and eliminates redundant processing, improving responsiveness and reliability of safety interventions. Chun's stated objective is to "achieve the optimal performance for driver/passenger state detection tasks," as a "majority of vehicle accidents reported worldwide are caused by distracted driving behaviors." EX1005, 1. IVMS include vision-based sensing technologies, including Chun's NADS-Net, that are designed to prevent and mitigate "traffic accidents by alerting the distracted driver and adaptively adjusting the safety mechanisms." *Id*. Chao explicitly provides a model that triggers actions in response to a driver's state being detected to enable "identification and performance of proper responses to dangerous situations such that self-perpetrated accidents can be avoided." EX1006, ¶[0065]. Triggering actions based on the categorical tags for each task as described in Chao directly supports the goal of Chun by detecting the specific drivers' states described in Chun that could cause vehicle accidents and triggering actions to prevent said vehicle accidents. The addition of Chao's vehicular actions directly advances Chun's stated purpose, improves risk prevention and mitigation, and aligns with industry trends toward closed-loop driver-monitoring systems.

229. **Reasonable Expectation of Success:** A POSITA would have had a high expectation of success implementing this combination. Chun and Chao describe mature, compatible deep-learning components: CNN-based backbones,

multi-head inference architectures, and downstream control integration in IVMS. Chao already demonstrates stable operation of a perception-to-action pipeline using CNN outputs to drive alerts and control adjustments, and Chun's backbone generates the same types of driver-state features (pose, gaze, hand position) on which Chao's control logic depends. Linking Chun's NADS-Net framework to Chao's in-vehicle system that performs vehicular actions would have been a routine engineering task, involving only standard software interfaces.

230. Commercial driver-monitoring and IVMS systems already demonstrated reliable operation of similar perception-to-action pipelines. A POSITA would thus expect predictable results: a driver-monitoring system that detects distraction via image classification and performs an appropriate safety action based on that tag, fulfilling the requirements of Claim 7 of the '243 patent.

231. For these reasons, a POSITA would conclude that Chun in view of Chao discloses or renders obvious the limitation of Claim 7: "performing at least one action based on the distraction classification tag." The combination yields a complete and predictable system consistent with established driver-monitoring and intelligent-vehicle design practices.

**F. Claim 8: The method of claim 7, the at least one action comprising one or more of updating a display of a computing device, playing an audible alarm, transmitting or a prediction to a remote server[1].**

232. Chun in view of Chao renders this claim obvious.

233. Claim 8 then gives specific examples of such actions by identifying three categories: (1) updating a display of a computing device, (2) playing an audible alarm, and (3) transmitting a prediction to a remote server. A POSITA would read claim 8 as confirming that the downstream "action" from claim 7 includes these typical HMI/IVMS response channels. Claim 8 is satisfied by demonstrating a system that performs a single one of those actions in response to the distraction-classification tag.

234. As discussed for claim 6 and 7, a POSITA would understand that Chun in view of Chao discloses the generation of a distraction classification tag that identifies driver states such as texting and calling and triggering an action based on the identified driver state. Building upon that foundation, a POSITA

---

[1] The claim language recites "transmitting or a prediction to a remote server."A POSITA would understand this to mean "transmitting a prediction to a remote server," and I interpret it as such. This revision is purely grammatical and does not alter the substance of the limitation.

would recognize and have found it obvious that the same combined architecture naturally performs or enables additional downstream actions, such as updating a display, issuing an audible alert, or transmitting the predicted distraction state to a remote server, as expressly illustrated in Chao.

**1. Chun**

235. Chun teaches a specific CNN architecture for an IVMS that includes multiple detection heads for "optimal performance for driver/passenger state detection tasks." EX1005, 1. IVMS is becoming a "standard technology in consumer vehicles as they can play a critical role in preventing and mitigating traffic accidents by alerting the distracted driver and adaptively adjusting the safety mechanisms." *Id*. Chun provides "a new convolutional neural network (CNN) architecture for 2D driver/passenger pose estimation and seat belt detection that is more nimble and, thus, more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches." *Id*. Although Chun does not specify the alert or transmission mechanics, a POSITA would understand that Chun's network produces the perception output consumed by downstream vehicle-alert or control modules. Thus, Chun supplies the trained model that provides the driver-state signal later used for the claimed notification or transmission actions.

**2. Chao**

236. Chao discloses a comparable multi-task convolutional architecture that processes image data through a shared backbone and produces multiple outputs, including a classification branch that assigns a class label of a driver's detected state. Chao then utilizes the determined state for "triggering the vehicular action based on the determined state." EX1006, ¶[0129].

237. Similar to Chun, Chao provides methods and devices that "use in-vehicle passenger monitoring data" to "provide accurate and quick mechanisms to recognize a driver's state" using "one or more in-vehicle data acquisition devices (e.g. a camera providing RGB, depth, and/or infrared data)." The methods and devices then use the driver's state determined by the "spatio-temporal analysis modeling scheme to trigger vehicular actions," including "providing an appropriate distance from a vehicle located in front or providing a notification to the driver of the vehicle." *Id*., ¶[0022]. Thus, a vehicle "may be capable of sensing its environment and/or sense changes inside the vehicle and navigate without direct human input and/or provide notifications to occupants of the vehicle." *Id*., ¶[0068].

238. In regards to updating a display of a computing device or playing an audible alarm, the determined state may trigger actions including "communicating one or more notifications via the UIs." *Id*., ¶[0095]. As described, the vehicular action "may be sent to other parts of the vehicle, such as vehicular controller 320

of mobility system 120 and/or the application processor 216 to send a notification to one or more passengers via one or more of the UIs 206." *Id*., ¶[0117]. The notification may include "a visual notification, a sound notification, or a vibration notification." *Id*., ¶[0168]. A POSITA would understand that displaying a visual notification communicated to a UI would be the same as updating a display of a computing device and a sound notification is an audible alarm.

239. In regards to transmitting a prediction to a remote server, Chao describes that a perception system 302 "may determine state data for each object/feature" to determine the classification of a driver. *Id*., ¶[0075]. Using the output of perception system 302, a planning system provides "a plan to a vehicle controller 320 of the mobility system 120 that controls one or more vehicle controls such as Engine Control 322, Brake Control 324, and/or Steer Control 326 to execute the plan," in response to a driver's state detection and "may additionally or alternatively provide a notification to the Application Processor 216 to communicate via one or more UIs 206." *Id*., ¶[0078]. A POSITA would understand that mobility system 120 or Application Processor 216 may be a remote entity separate from perception system 302, as "any single processor, controller, or logic circuit detailed herein may be realized as two (or more) separate entities with equivalent functionality." *Id*., ¶[0032].

240. A POSITA would recognize that these notifications and transmissions are all triggered based on the distraction-classification tag output by the model's SoftMax layer. Accordingly, Chao supplies the explicit mechanism by which the predictions from Chun or Chao can be routed to a speaker, display, or communication module for a safety-related action in real time.

**3. MTC**

241. **How:** A POSITA would have connected the distraction-classification output of the combined Chun + Chao network to Chao's existing user-interface and communication subsystems. In operation, driver-camera images feed Chun's backbone; extracted features enter Chao's classification head, which outputs a categorical distraction tag. That tag triggers a visual update on the vehicle display, activates an audible or haptic alert, or transmits the predicted distraction state to a remote server via the communication processor 218 using standard API or bus messaging. Each action is directly conditioned on the tag, satisfying the claim's causal requirement. As described in claim 7, Chun already discusses implementing vision-based sensing technologies in IVMS to trigger alerts and adjust safety mechanisms of a vehicle. The implementation of Chao's in-vehicle system for triggering vehicular action would be obvious and straightforward since the combination would require only routine signal routing and software integration. Specifically, the implementation path is straightforward: integrate Chun's NADS-

Net in the in-vehicle control systems disclosed in Chao. The resulting system would update the in-vehicle display, issue an audible alarm, or transmit the data to a remote server using hardware and software already present in Chao's system.

242. **Why**: Chun's stated objective is to "achieve the optimal performance for driver/passenger state detection tasks," as a "majority of vehicle accidents reported worldwide are caused by distracted driving behaviors." EX1005, 1. IVMS include vision-based sensing technologies, including Chun's NADS-Net, that are designed to prevent and mitigate "traffic accidents by alerting the distracted driver and adaptively adjusting the safety mechanisms." *Id*. Chao explicitly provides a model that triggers actions in response to a driver's state being detected to enable "identification and performance of proper responses to dangerous situations such that self-perpetrated accidents can be avoided." EX1006, ¶[0065]. Triggering actions that alert a driver or provide a plan to a remote system that performs vehicular actions based on the categorical tags for each task as described in Chao directly supports the goal of Chun by detecting the specific drivers' states described in Chun that could cause vehicle accidents and triggering actions to prevent said vehicle accidents. The addition of Chao's vehicular actions directly advances Chun's stated purpose, improves risk prevention and mitigation, and aligns with industry trends toward closed-loop driver-monitoring systems.

243. **Reasonable Expectation of Success:** A POSITA would have had a high expectation of success implementing this combination. Chun and Chao describe mature, compatible deep-learning components: CNN-based backbones, multi-head inference architectures, and downstream control integration in IVMS. Linking Chun's NADS-Net framework to Chao's in-vehicle system that performs vehicular actions including UI display and alert systems would have been a routine engineering task, involving only standard software interfaces.

244. Commercial driver-monitoring and IVMS systems already demonstrated reliable operation of similar perception-to-action pipelines. A POSITA would thus expect predictable results: a driver-monitoring system that detects distraction via image classification and performs an appropriate safety action based on that tag including driver notification and remote transmission, fulfilling the requirements of Claim 8 of the '243 patent.

245. For these reasons, a POSITA would conclude that Chun in view of Chao discloses or renders obvious the limitation of Claim 8: "at least one action comprising one or more of updating a display of a computing device, playing an audible alarm, transmitting a prediction to a remote server." The combination yields a complete and predictable system consistent with established driver-monitoring and intelligent-vehicle design practices.

### G. Claim 9: The method of claim 1, wherein the distraction classification prediction head classifies an input vector with a label, the label comprising one or more of an undistracted, mobile usage, or other distracted label.

246. Chun in view of Chao renders this claim obvious.

#### 1. Chun

247. As described in [1D], because Chun's architecture already includes multiple prediction heads sharing the same backbone feature maps (*see* Figure 3), a POSITA would have found it obvious to add an additional distraction classification prediction head to output a categorical "distracted/not distracted" or multi-class distraction type label. The existing heads (key-point, part-affinity, and seat-belt segmentation) already extract pose and contextual cues necessary for distraction classification (*e.g.*, hand position, gaze direction, and seat-belt use) so extending the architecture with a distraction classification head would naturally reuse these shared features. As previously discussed, because these feature maps already encode hand position, gaze direction, and body pose (*i.e.*, visual cues necessary for distraction inference) a POSITA would have found it obvious to add an additional classification head that consumes those shared feature vectors and outputs a categorical driver-state label such as "distracted" or "not distracted."

248. Additionally, the PAF head proposes PAF that "encompass vector fields indicating how individual key points should be connected" by "encod[ing]

pairwise relationships between body joints," thus indicating that the task heads may utilize vectors for performing their tasks. EX1005, 2, 5. Chun itself emphasizes that NADS-Net was designed to be a multi-branch architecture supporting "driver/passenger state detection tasks," providing an explicit motivation and structural pathway for such an extension of a classification head that classifies input vectors. *Id*., 1.

249. Chun thus establishes that determining the state of a distracted driver is the primary goal of the described CNN architecture and presents architecture and training data that allow a POSITA to easily and effectively add a distraction classification head that classifies input vectors to the disclosed CNN.

250. While Chun does not itself classify distraction states, it establishes the architectural and data framework: multi-head CNN trained on annotated images of distracted behaviors such as "drinking," "using smartphones," "adjusting sun visor," and the like, EX1005, 4, that would supply the labeled training examples used by an added classification head. A POSITA would therefore view Chun as supplying the perception foundation on which a distraction-classification head could operate. Such a distraction-classification head is disclosed by Chao, which further teaches classification of driver distraction states using vectors.

**2. Chao**

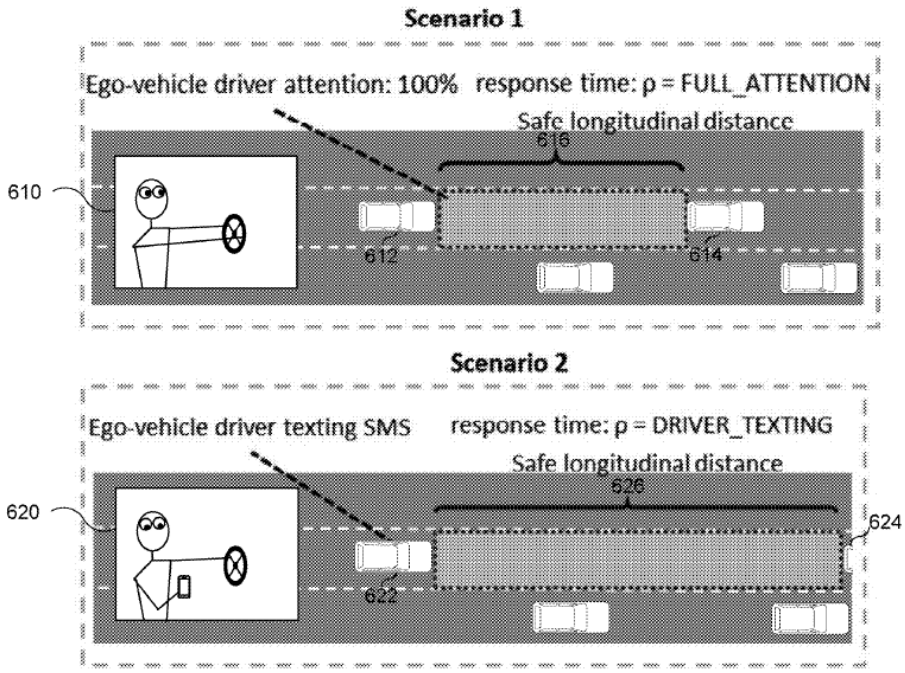
251. Similar to Chun, Chao describes a multi-stage neural network that provides "inputs, based on the plurality of subsets of image data, to a two-dimensional convolutional neural network (2D CNN)." EX1006, ¶[0136]. Using these inputs, Chao describes a distraction classification head that provides a classification of a driver's distraction state attached to a shared backbone (*i.e.*, 2D CNN), similar to the shared backbone of Chun.

252. Chao's distraction classification prediction head includes a temporal filtering section 804 whose output "may be in the form of a vector of $N_C$ values (where $N_C$ is the number of classes)." The vector is provided as input to a SoftMax layer, which may "normalize this vector into a probability distribution vector consisting of $N_C$ probabilities proportional to the exponentials of the vector input to this SoftMax layer." As described in Chao, "[e]ach of the classes of the vector (i.e., each $N_C$) is provided as a value, or 'Action Score,' indicating the determined class based on the input. This class may correspond to a status of the driver." *Id*., ¶[0101]. Stated differently, each class of this vector represents a driver-state label, called an "Action Score," identifying the determined class based on the input vector. Thus, Chao explicitly describes classifying an input vector with a label corresponding to the driver's state

253. A POSITA would understand that the determined class is a label determined by the ST model. The "Drive both hands," corresponds to an undistracted label, "text left hand," "text right hand," "call left hand," and "call right hand," corresponds to a mobile usage label, and "switch gear," "talk to passenger," "adjust radio," "drink," "hair makeup," "reach side," "reach behind," and "hands free" correspond to other distracted labels. *Id*., Table 1.

254. Additionally, Figure 6 of Chao provides an example illustrating "Ego-vehicle driver attention: 100% full attention" versus "Ego-vehicle driver texting SMS", which indicates a specific classification label of driver distraction. Specifically, scenario 1 tags the driver attention as "FULL_ATTENTION" and scenario 2 tags the driver attention as "DRIVER_TEXTING," as shown below.

FIG 6




**EX1006, FIG. 6**

255. Chao therefore squarely teaches a distraction classification head that labels an input vector as either undistracted, mobile usage, or other distracted labels.

### 3. MTC

256. **How:** A POSITA would integrate Chao's SoftMax-based classification branch into Chun's existing FPN backbone by routing the shared 96 × 96 × 512 feature tensor into the classifier as the "input vector." EX1005, 4-5. Chao's temporal-filter output vector and Chun's feature-pyramid vector share dimensional compatibility, making the connection straightforward. The resulting head would classify the feature vector into categorical distraction labels, *e.g.*,

"undistracted," "mobile usage," or "other distracted." This integration requires no alteration of the convolutional blocks or loss functions, only the addition of a SoftMax layer and label-mapping table as shown in Chao ¶0101 and Table 1.

257. While Chun does not expressly disclose a distraction classification prediction head, it provides a multi-task neural architecture that is structurally and functionally compatible with one. Chun's system already incorporates a shared convolutional backbone (FPN backbone) and multiple specialized prediction heads for driver state detection, including pose estimation and seat belt detection. Each of these heads receives shared feature maps and produces task-specific outputs as the backbone "produces a rudimentary feature pyramid for the later detection branches." EX1005, 5. Chun's framework shows that pose estimation, segmentation, and detection can be trained from a shared backbone under separate task-specific heads without architectural interference, including task heads that utilize input vectors. Thus, this combination of a backbone network and task-specific heads provide the architectural framework on which a distraction classification head could operate.

258. Chao provides a concrete, enabling example of a distraction classification prediction head that labels input vectors. Chao expressly discloses a distraction classification head implemented as a distinct model separate from a feature extraction model, wherein the model receives the extracted features as

input. It would be a conventional and straightforward technical implementation to include the distraction classification head in Chun's multi-stage architecture as both architectures employ a shared backbone feeding separate prediction heads, and Chao's ST model for driver state classification is expressly designed as a modular branch that receives extracted features as input. Because Chun's backbone produces feature maps of the driver's pose and seat belt donning, a POSITA would have recognized that the same feature maps could serve as input to a distraction classification head without altering the network's fundamental design.

259. **Why**: A POSITA would have been motivated to make this combination to achieve a unified model capable of both low-level pose detection and high-level distraction classification using shared features. This approach eliminates redundant networks, reduces inference latency, and enables multitask training with common feature extraction. A POSITA would have considered Chun and Chao to be analogous art addressing closely related problems within the same field of in-vehicle computer vision. Both references teach modular, multi-task networks trained for driver state detection. Chun focuses on the structure of a CNN for detection of human key points, human pose estimation, and object (*i.e.*, seatbelt) detection within an image of a driver in a vehicle using a backbone and prediction heads. Chao addresses the structurally similar problem of recognizing driver attributes and behaviors, such as drinking, phone use, and gaze direction,

from image data by analyzing features and objects in a vehicle's internal environment using the same type of convolutional feature extraction and detection-head architecture. Both systems are directed to neural-network-based multi-task perception of human pose or behavior from visual input and rely on shared backbones feeding task-specific heads/models. Because both references lie squarely within the domain of deep-learning-based human detection and activity recognition in in-vehicle systems, a POSITA would have naturally looked to Chao when seeking to improve Chun's system with additional driver state inference capabilities.

260. **Reasonable Expectation of Success:** Both references use convolutional backbones that provide spatial feature extraction suitable for pose estimation and object detection. Chao includes a classification model that is separate from a convolutional backbone and draws from the features extracted by the CNN. Chun's network already includes multiple heads drawing from shared features, proving its compatibility with additional branches. The modification would therefore have been straightforward: the distraction classification head from Chao could be connected to Chun's shared feature maps and trained with annotated images, in the same manner as Chun's prediction heads. Further, adding a classification head to Chun's FPN backbone would have been a routine extension of a known multi-task learning paradigm by 2019. Multi-head CNNs (*e.g.*, Mask

R-CNN, FPN) commonly employed shared feature extractors feeding specialized heads for detection, segmentation, and classification. Because Chun's backbone outputs dense feature maps encoding face, hand, and body-pose geometry, precisely the features used by Chao to infer distraction, a POSITA would have expected predictable success in training a new head to output categorical distraction labels. The modification would require only the addition of a soft-max classification branch that classifies input vectors and labeled examples already available in Chun's dataset. No redesign of the backbone, loss functions, or data pipeline would be necessary.

261. No technical obstacles would have discouraged a POSITA from making this combination/modification. Chao's distraction classification head is modular and uses standard convolutional layers and vector encoding already employed in Chun. Because both systems use CNNs to perform feature extraction, integration would not require redesign of the backbone or attention modules. A POSITA would have reasonably expected that combining Chao's distraction classification head with Chun's existing pose estimation and object detection heads would yield predictable success, producing a unified model capable of detecting both what the driver is doing and how the driver's body is oriented, thereby improving recognition accuracy consistent with Chun's objectives.

### H. Claim 10: The method of claim 1, wherein the object detection prediction head detects a location of an object within an input vector.

262. Chun renders this claim obvious.

263. Chun describes the NADS-Net architecture as employing a feature-pyramid backbone with multiple prediction heads, including a seat belt detection head. EX1005, 1. The CNN is trained using images of drivers performing specific gestures while wearing a seatbelt, including "drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id*., 4. These gestures provided additional pose training images to the "other publicly available pose estimation data sets such as MS COCO." *Id*.

264. The disclosed prediction head that performs seat belt detection "produces a probability density function defined over the image domain, indicating the likelihood of a pixel being a seat belt." This probability density is then "thresholded to obtain a binary seat belt segmentation mask." *Id*., 6. This binary seat belt mask provides "more granularity in detection results" in order to correctly detect the seat belt *and* proper use of a seat belt. *Id*., 3. While Chun does not expressly disclose using an input vector in the seat belt detection head, it does describe using an input vector as input in a different prediction head. Using the CNN trained on the collected training data, a PAF head proposes PAF that

"encompass vector fields indicating how individual key points should be connected" by "encod[ing] pairwise relationships between body joints," thus indicating that the orientation of the one or more body parts of a driver is depicted by input vectors. *Id*., 2, 5. Although Chun describes the output as a "vector field," a POSITA would understand that the input to the PAF head is a feature vector (*i.e.*, encoded image tensor) representing spatial embeddings of detected keypoints. Since each detection head receives the same input from the backbone network, EX1005, 5, it would be obvious to a POSITA that the seat belt prediction head would also receive the feature vector representing spatial features as input. Thus, Chun discloses a backbone network that provides an input vector to a prediction head, which includes the seat belt detection head.

### I. Claim 11: The method of claim 1, wherein the pose estimation prediction head detects an orientation of one or more body parts of a user depicted in an input vector.

265. Chun discloses this claim.

266. Chun describes the NADS-Net architecture as employing a feature-pyramid backbone with multiple prediction heads, including a key-point detection head and a part-affinity-field (PAF) detection head for "2D driver / passenger pose estimation." EX1005, 1. The CNN is trained using images of drivers performing specific gestures, including "drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and

the center console, adjusting sun visor, and etc." *Id*., 4. These gestures provided additional pose training images to the "other publicly available pose estimation data sets such as MS COCO." *Id*.

267. Using the CNN trained on pose estimation training data, the PAF head proposes PAF that "encompass vector fields indicating how individual key points should be connected" by "encod[ing] pairwise relationships between body joints," thus indicating that the orientation of the one or more body parts of a driver is depicted by input vectors. EX1005, 2,5. Although Chun describes the output as a "vector field," a POSITA would understand that the input to the PAF head is a feature vector (*i.e.*, encoded image tensor) representing spatial embeddings of detected keypoints. The PAF head processes these input vectors to predict orientation vectors linking pairs of joints. Thus, Chun discloses a pose-estimation prediction head that detects orientation of body parts represented within an input feature vector.

268. Thus, Chun expressly discloses a pose estimation prediction head that "detects an orientation of one or more body parts of a user depicted in an input vector."

### J. Claim 12: The method of claim 1, wherein storing a distraction classification model comprises storing the distraction classification model in a dash-mounted camera device.

269. Chun in view of Chao renders this claim obvious.

#### 1. Chun

270. As described in element [1H], while Chun does not expressly disclose the act of storing a trained model, its teachings strongly suggest that the trained CNN architecture, NADS-Net, would be saved and deployed for inference once training and validation were complete. Chun presents NADS-Net as a complete, deployable CNN designed for in-vehicle monitoring tasks, describing it as "more nimble and thus more suitable for in-vehicle monitoring tasks" and capable of "non-invasive, non-obstructive means to monitor and detect in-cabin activities" EX1005, 1. From these statements, POSITA would understand that Chun envisions the trained model being implemented in production IVMS systems and that the trained parameters of both the backbone and the detection heads are stored for subsequent inference.

271. In order to train NADS-Net, Chun "equipped the research vehicle with IR lights and two cameras. One of the cameras was mounted below the rear-view mirror and another was above the center media console. IR lights were installed on the dashboard and behind the sun visors." EX1005, 4. While these cameras served as image-capture units during training and testing, a POSITA

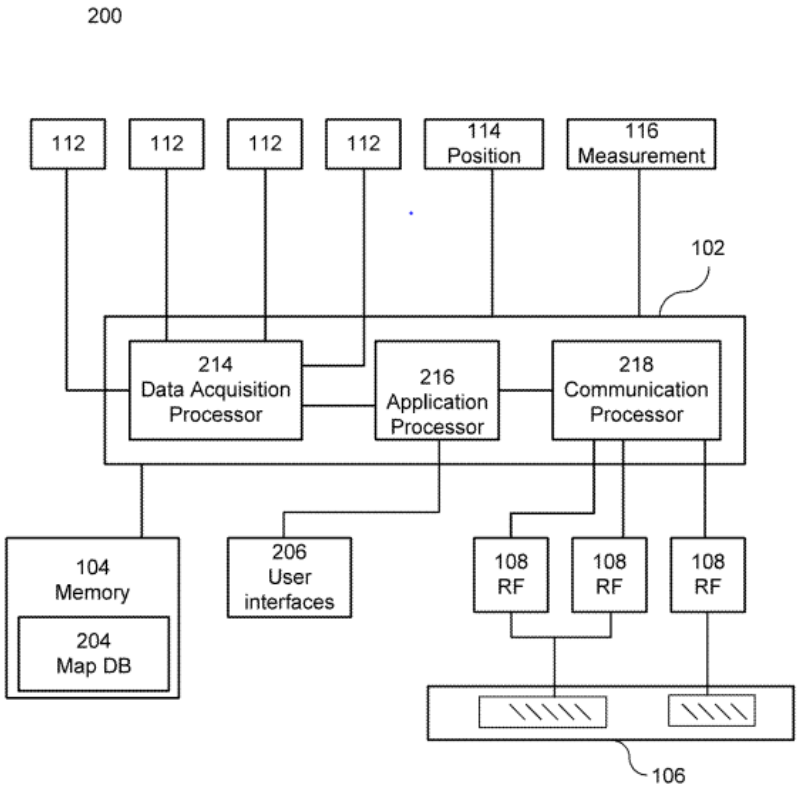
would recognize them as representative of in-vehicle sensing hardware used during deployment.

272. Where Chun suggests this step but omits implementation detail, Chao provides explicit teaching that would have made such storage in a dash-mounted device obvious to a POSITA.

**2. Chao**

273. Chao describes a system wherein a dash-mounted device stores the processors and memories that are configured to store and deploy the driver state classification system. As shown in Figure 2, Chao describes a control system 200 that includes data acquisition devices 112 (*i.e.*, cameras), a memory 104, and a plurality of processors.

FIG 2

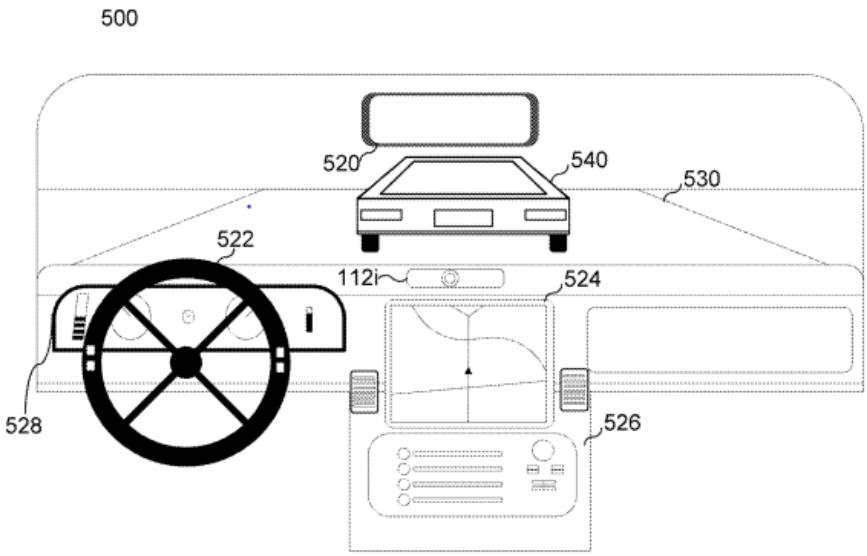



**EX1006, FIG. 2**

274. As described, "[a]ny of the processors 214, 216, 218 disclosed herein may be configured to perform certain functions in accordance with program instructions which may be stored in a memory of the one or more memories 104… ***A memory of the one or more memories 104 may store*** one or more databases and image processing software, as well as ***a trained system, such as a neural network***, or a deep neural network." (*emphasis added*) EX1006, ¶[0054]. Thus, the memory comprising the trained neural network, processor, and camera may all be configured in the same system, namely the control system 200.

275. As a part of this control system 200, the data acquisition devices 112 "may include cameras (e.g., standard cameras, digital cameras, video cameras, single-lens reflex cameras, infrared cameras, stereo cameras, etc.), charge coupling devices (CCDs) or any type of image sensor." *Id*., ¶[0060]. As shown in Figure 5A, the data acquisition device 112 is integrated in an internal vehicle setup and may be placed on the dash, or "at various other locations, such as rear view mirror 520, steering wheel 522, front-side display monitor 524, center console 526, dashboard 528, etc." *Id*., ¶[0086]. Accordingly, the trained network stored in memory 104 may reside within or adjacent to a dash-mounted unit that also houses the camera sensor. Although Chao lists multiple possible locations, one explicit option is a dashboard configuration.

FIG 5A



**EX1006, FIG. 5A**

276. Finally, Chao notes that "skilled persons will appreciate the various possibilities to combine or integrate discrete elements into a single element," including "executing discrete software components on a common processor core" EX1006, ¶[0258]. A POSITA would thus understand that the trained network and its memory could be co-located within the dash-mounted camera housing, satisfying the claimed configuration.

**3. MTC**

277. **How:** Both Chun and Chao employ convolutional networks for driver-monitoring tasks and rely on interior vehicle cameras for data capture. Integrating Chao's memory-and-processor configuration with Chun's trained CNN would involve storing the model weights on the local memory of the dash-mounted camera and loading them at runtime for on-device inference on captured frames. This modification would simply apply Chao's explicit deployment architecture to Chun's trained NADS-Net model. Both references rely on the same class of neural network models, and the step of storing the optimized parameters is independent of the specific task (distraction classification in Chao or pose estimation and object detection in Chun). Additionally, both references rely on cameras mounted to the dash or interior of a vehicle to obtain data. As such, a POSITA would have expected that saving Chun's trained backbone and head weights to a dash-mounted

device in the manner explicitly taught by Chao would predictably yield a working deployed model, ready for inference in runtime environments. The result would simply be the ordinary and reliable outcome of applying a well-established technology of dash-mounted cameras to a closely related driver state detection task.

278. Storing trained parameters to enable deployment in an in-vehicle system was a technique that had already been used across a range of machine-learning models and devices. A POSITA would naturally apply the same technique of implementing Chao's in-vehicle monitoring system to Chun's driver state detection model, recognizing it would provide the same benefits, namely, in-vehicle monitoring to prevent and mitigate risks associated with distracted driving, without requiring any inventive insight or leap. In other words, the claimed step reflects nothing more than applying an established solution to a system that was already prepared for implementation in in-vehicle systems.

279. **Why**: Chun's stated objective is to "achieve the optimal performance for driver/passenger state detection tasks," as a "majority of vehicle accidents reported worldwide are caused by distracted driving behaviors." EX1005, 1. IVMS include vision-based sensing technologies, including Chun's NADS-Net, that are designed to prevent and mitigate "traffic accidents by alerting the distracted driver and adaptively adjusting the safety mechanisms." *Id*. Chao explicitly provides a

model that is stored and implemented in a vehicle to enable "identification and performance of proper responses to dangerous situations such that self-perpetrated accidents can be avoided." EX1005, ¶[0065]. Providing an IVMS using the dash-mounted device in Chao directly supports the goal of Chun by providing a system that may quickly and efficiently detect a driver's state described in order to prevent vehicle accidents. The addition of Chao's dash-mounted device directly advances Chun's stated purpose, improves risk prevention and mitigation, and aligns with industry trends toward closed-loop driver-monitoring systems.

280. **Reasonable Expectation of Success** A POSITA would have had a complete and reasonable expectation of success (1) **Architectural Compatibility:** Chun and Chao both use convolutional backbones with shared features and task-specific heads; storing trained weights is agnostic to the specific tasks; (2) **Operational Compatibility:** Chun and Chao demonstrate real-time, multi-branch inference; (3) **Predictable Outcome:** Combining Chao's storage mechanism with Chun's unified driver-monitoring model yields the expected, ordinary result: a trained model stored in the dash-mounted camera for immediate inference on driver images. No redesign or inventive insight would be needed.

281. Accordingly, Chun in view of Chao renders Claim 12 obvious. Chun supplies the compact, modular architecture compatible with embedded inference; and Chao provides the in-vehicle, camera-based hardware and training-to-

deployment workflow and the explicit teaching of storing trained model parameters in device memory. A POSITA would thus find it obvious and routine to store the finalized distraction-classification model, comprising the backbone and prediction-head parameters, within the dash-mounted camera device for real-time driver-state monitoring, exactly as recited in Claim 12 of the ’243 patent.

### K. Claims 13, 14, 18-20, 24, and 25

282. Claims 13, 14, 18-20, 24, and 25 recite the same substantive limitations as claims 1, 2, 6-9, 11, and 12 but expressed in different statutory formats (computer-readable medium and device claims rather than method claims).

283. Claim 13 mirrors the method of claim 1, except that it is directed to “a non-transitory computer-readable storage medium” storing program instructions. Claims 14, 18, and 19 then correspond directly to dependent claims 2, 6, and 7, adding the same minor refinements such as a distraction classification head comprising multiple convolutional layers (claim 14 vs. claim 2), receiving a distraction classification tag (claim 18 vs. claim 6), and performing an action based on the distraction classification tag (claim 19 vs. claim 7).

284. Likewise, claim 20 recites the same functional steps as claims 1 and 13 but in the context of a “device” with a processor and storage medium, and claims 24 and 25 track the same refinements as claims 6, 7, 18 and 19.

285. The only differences between claims 1, 6-9, 11, and 12 and claims 13, 18-20, 24, and 25 are the claim formats (method vs. storage medium vs. device).

286. A POSITA would have understood that the same disclosed training methods could be implemented as software instructions stored on a non-transitory medium (claims 13, 18, and 19) or as logic executed on a processor within a device (claims 20, 24, and 25). Chun in view of Chao discloses as much.

287. Specifically, Chao already describes implementing distraction classification through software and hardware modules, including in-vehicle sensors and/or camera 702, spatio-temporal system 704, and associated processors and storage. *See, e.g.*, EX1006, ¶¶[0092]-[0093]. Thus, recasting the same operations into "computer-readable medium" or "device" form involves no substantive difference, but merely reflects conventional statutory claiming techniques.

288. Accordingly, for the same reasons set forth with respect to claims 1, 6-9, 11, and 12, claims 13, 18-20, 24, and 25 are rendered obvious by Chun in view of Chao.

### L. Independent Claim 26

289. Claim 26 recites substantially the same subject matter as claim 1, with additional limitations corresponding to the features of dependent claims 6-8 and 12. Each of those component limitations has already been addressed in the analysis

of claims 1, 6, 7, 8, and 12 under Ground 1. Accordingly, the same combination of Chun and Chao renders claim 26 obvious for the same reasons.

290. In particular, **claim 26[A]** which recites "storing a distraction classification model in a dash-mounted camera device" is identical to claim 12. As discussed above in Sec. VIII.I, Chun in view of Chao discloses storing a distraction classification model in a dash-mounted camera device. *See* EX1006, ¶¶[0244-0256[.

291. **Claim 26[B]** which recites "a backbone network and one or more prediction heads coupled to the backbone network, the one or more prediction heads comprising one or more of a distraction classification prediction head, object detection prediction head, and a pose estimation prediction head, wherein the distraction classification model comprises a model trained by inputting labeled examples into the distraction classification model and minimizing a loss based on outputs of the one or more prediction heads causing a change in parameters of the backbone network;" is identical to claim [1A]-[1G]. As discussed above in Sec. VIII.B.1-8, Chun in view of Chao discloses the distraction classification model described in claim 26. *See* EX1006 ¶¶[0080-0151].

292. **Claim 26[C]** which recites "capturing an image of an interior of a vehicle using a sensor in the dash-mounted camera device;" is expressly disclosed by Chun, as discussed above in Sec. VIII.I. Specifically, in order to train NADS-

Net, Chun "equipped the research vehicle with IR lights and two cameras. One of the cameras was mounted below the rear view mirror and another was above the center media console. IR lights were installed on the dashboard and behind the sun visors." EX1005, 4. While these cameras served as image-capture units during training and testing, a POSITA would recognize them as representative of in-vehicle sensing hardware used during deployment.

293. Chao describes a system wherein a dash-mounted device stores the processors and memories that are configured to store and deploy the driver state classification system. As shown in Figure 2, Chao describes a control system 200 that includes data acquisition devices 112 (*i.e.*, cameras), a memory 104, and a plurality of processors. The data acquisition processors "include a perception system 302, a prediction system 304, and a planning system 306 that cooperate to ***perceive the… internal (i.e., inside of the vehicle) environment of vehicle 100*** and determine a plan for controlling the mobility or positioning of vehicle 100 and/or issue notifications to one or more occupants." EX1006, ¶[0069] (emphasis added).

294. Both Chun and Chao employ convolutional networks for driver-monitoring tasks and rely on interior vehicle cameras for data capture. Therefore, as discussed above in Sec. VIII.I, Chun in view of Chao discloses capturing an image of an interior of a vehicle using a sensor in the dash-mounted camera device. *See* EX1006, ¶¶[0244-0256].

295. **Claim 26[D]** which recites "inputting the image into the distraction classification model; receiving a distraction classification tag output by the distraction classification model;" is nearly identical to claim 6. As discussed above in Sec. VIII.D, Chun in view of Chao discloses inputting an image into the distraction classification model and receiving a distraction classification tag. *See* EX1006, ¶¶[0179-0198].

296. **Claim 26[E]** which recites "performing at least one action by the dash-mounted camera device based on the distraction classification tag, the at least one action comprising one or more of updating a display of a computing device, playing an audible alarm, transmitting or a prediction to a remote server" is nearly identical to claims 7 and 8. As discussed above in Secs. VIII.E and VIII.F, Chun in view of Chao discloses performing actions based on the distraction classification tag. *See* EX1006, ¶¶[0199-0223].

### M. Claim 27: The method of claim 26, wherein the distraction classification prediction head classifies the image with a label, the label comprising one or more of an undistracted, mobile usage, or other distracted label.

297. Claim 27 is substantially the same subject matter as claim 9. As discussed above in Sec. VIII.G, Chun in view of Chao discloses classifying images with a label. *See* ¶¶244-259.

**N. Claim 28: The method of claim 26, wherein the object detection prediction head detects a location of an object within the image.**

298. The seat belt detection head, as described in Chun, provides detection results that "provide information on, not only use and non-use of seat belt, but also proper/improper use cases judged by the relative position of seat belt to the detected body position." EX1005, 3. The disclosed prediction head that performs seat belt detection "produces a probability density function defined over the image domain, indicating the likelihood of a pixel being a seat belt." This probability density is then "thresholded to obtain a binary seat belt segmentation mask." *Id*., 7. This binary seat belt mask provides "more granularity in detection results" in order to correctly detect the seat belt *and* proper use of a seat belt. *Id*., 2. An example of this seatbelt detection can be seen in Figure 5 of Chun:



Figure 5: Seat belt detection result (*left*) in comparison with the ground truth (*right*). With human inspection, the prediction result is of good quality as it correctly marks the seat belt area. However, IOU for this particular example was 46%, justifying the need for a better evaluation metric.

**EX1005, FIG. 5**

299. This seat belt detection prediction head provides the exact location of a seatbelt in an image of a driver/passenger using the pixels of the image. Thus, Chun expressly discloses an object detection prediction head that "detects a location of an object within the image data."

**O. Claim 29: The method of claim 26, wherein the pose estimation prediction head detects an orientation of one or more body parts of a user depicted in the image.**

300. Chun describes the NADS-Net architecture as employing a feature-pyramid backbone with multiple prediction heads, including a key-point detection head and a part-affinity-field (PAF) detection head for "2D driver / passenger pose estimation." EX1005, 1. The CNN is trained using images of drivers performing specific gestures, including "drinking, using smart phones, exercising, yawning, sneezing, leaning on the door, putting hands out the window, searching floor and the center console, adjusting sun visor, and etc." *Id*., 4. These gestures provided additional pose training images to the "other publicly available pose estimation data sets such as MS COCO." *Id*.

301. Using the CNN trained on pose estimation training data, the PAF head proposes PAF that "encompass vector fields indicating how individual key points should be connected" by "encod[ing] pairwise relationships between body joints," thus indicating that the orientation of the one or more body parts of a driver is depicted in the image. EX1005, 2, 5.

302. Thus, Chun expressly discloses a pose estimation prediction head that "detects an orientation of one or more body parts of a user depicted in the image data."

**P. Claims 30-33**

303. Claims 30-33 recite the same substantive limitations as claims 26-29 but expressed in different statutory formats (device claims rather than method claims).

304. Claim 30 mirrors the method of claim 26, except that it is directed to "a dash-mounted camera device" comprising a memory and processor. Claims 31-33 then correspond directly to dependent claims 27-29, adding the same minor refinements such as classifying images with a label (claim 31 vs. claim 27), detecting a location of an object within an image (claim 32 vs. claim 28), and detecting an orientation of a body part of a user in the image (claim 33 vs. claim 29).

305. The only differences between claims 26-29 and claims 30-33 are the claim formats (method vs. device). A POSITA would have understood that the same disclosed training methods could be implemented as logic executed on a processor within a device (claims 30-33). Chun in view of Chao discloses as much.

306. Specifically, Chao already describes implementing distraction classification through software and hardware modules, including in-vehicle sensors

and/or camera 702, spatio-temporal system 704, and associated processors and storage. *See, e.g.*, EX1006, ¶¶[0092]-[0093]. Thus, recasting the same operations into "device" form involves no substantive difference, but merely reflects conventional statutory claiming techniques.

307. Accordingly, for the same reasons set forth with respect to claims 26-29, claims 30-33 are rendered obvious by Chun in view of Chao.

## IX. GROUND 2: CHUN, CHAO, AND SHANMUGAMANI RENDERS OBVIOUS CLAIMS 3-5, 15-17, and 21-23

308. Under Ground 2, Chun in view of Chao and further in view of Shanmugamani discloses each limitation of dependent claims 3-5, 15-17, and 21-23, which recite conventional neural network functions and structures of the same driver distraction classification model.

### A. Motivation to Combine

309. Ground 1 already demonstrated that the independent claims are obvious in light of Chun and Chao. Building on that foundation, Ground 3 shows that dependent claims 3-5, 15-17, 21-23, 28, and 32 are likewise unpatentable. These claims add additional limitations that describe conventional CNN architecture and design for object detection and image classification. As explained below, these additional conditions were well known in the art, and a POSITA would have found it obvious to incorporate these conditions into Chun as further

modified by Chao (Ground 1), in view of Shanmugamani's express disclosure of computer vision techniques for object detection and image classification.

**1. Chun's CNN Architecture for Distracted Driving Detection.**

310. Chun is directed towards a convolutional neural network (CNN) architecture for driver and passenger state detection tasks, named NADS-Net. Chun explains that "a vast majority of vehicle accidents reported worldwide are caused by distracted driving behaviors," so they propose a new CNN architecture "that is more nimble and, thus, more suitable for in-vehicle monitoring tasks compared to other state-of-the-art approaches." EX1005, 1. Chun describes that NADS-Net is used to detect and alert drivers that are exhibiting distracted behavior, including but not limited to "use of smartphones/mobile devices, smoking tobaccos, engaging in a conversation with other passengers, drinking beverages, eating foods, and such," and "drowsiness, fatigues, medication effects, and other medical/physiological issues," *Id*., 1.

311. Chun expressly discloses that NADS-Net includes one or more prediction heads comprising at least 3 convolutional layers coupled to a backbone network, as illustrated in Figure 3.

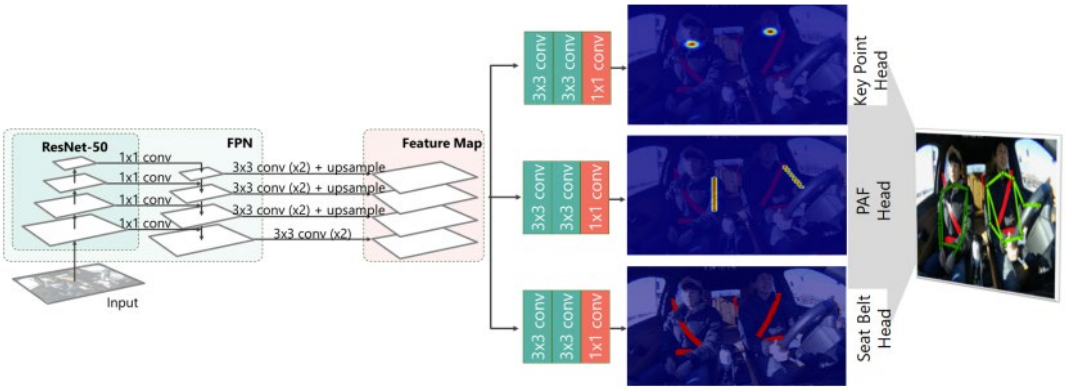



Figure 3: NADS-Net architecture.

**EX1005, FIG. 3**

312. This multi-stage architecture "was implemented in Keras with TensorFlow backend and an NVIDIA GeForce GTX 1080 Ti GPU was used for training and testing." EX1005, 1. As described, "[e]ach of the detection branches employs two 3×3 convolutions and a 1×1 convolution to predict a pixel-wise probability distribution." For the key point detection head, the pixel-wise probability "indicates the probability of the corresponding pixel being a certain joint," the part-affinity head "encode[s] pairwise relationships between body joints," and the seat belt detection head "generates a binary seat belt detection mask" based on "the likelihood of each pixel being a seat belt." *Id*., 5. These components are all directly coupled to the shared feature map output of the backbone network, demonstrating multiple prediction heads that generate distinct pixel-level outputs, demonstrating a multi-head structure consistent with the '243 patent.

313. While Chun discloses that each prediction head comprises convolutional layers, it does not specifically disclose specific layers (*e.g.*, pooling, fully connected, activation, and batch normalization layers) and functions (*e.g.*, bounding box regression and object class prediction) that are common and conventional in computer vision. However, a POSITA would understand that any CNN requires the use of these specific layers and functions for optimized image analysis. Because NADS-Net was implemented by TensorFlow, which is an open-source, end-to-end platform for machine and deep learning that provides simple tools for coding standard functions and layers, the use of the layers and functions described in the '243 patent would have been commonly implemented in NADS-Net and obvious to a POSITA.

**2. Chao's CNN Architecture for Distracted Driving Classification.**

314. Chao discloses systems, methods, and devices for triggering vehicular actions "based on the monitoring of one or more occupants of a vehicle." EX1006, Abstract. The primary goal of the system is to determine "states" of occupants to trigger these vehicular actions. These states often relate to driver attention and distraction, including "features/objects such as a hand, a bottle, a cellphone, a person holding the steering wheel with both hands, a profile of a face, the back of a person's head, etc." *Id*., ¶[0098].

315. Similar to Chun, Chao describes a multi-stage neural network that provides "inputs, based on the plurality of subsets of image data, to a two-dimensional convolutional neural network (2D CNN)." EX1006, ¶[0136]. Using these inputs, Chao describes a distraction classification head that provides a classification of a driver's distraction state attached to a shared backbone (*i.e.*, 2D CNN), similar to the shared backbone of Chun. Chao's distraction classification prediction head "uses several convolutional kernels to convert the absolute feature logits from the identified objects and/or features extracted from the 2D CN model to provide an indication of increasing or decreasing values," in order to determine "action scores for each of the classes which identify the one or more passengers' action (e.g., if the driver is at full attention)." EX1006, ¶¶[0106]-[0107]. A POSITA would understand this as teaching a standard backbone-plus-head structure for driver state classification.

316. A POSITA would understand from this disclosure that Chao teaches a conventional end-to-end learning workflow: train a backbone-plus-head network for driver distraction classification, optimize its parameters through loss minimization, and store the trained model (including all learned weights) in memory for subsequent loading and inference.

317. Accordingly, Chao provides both (1) the backbone-plus-head parameter structure that mirrors Chun's architecture for driver state detection and

(2) the standard loss minimization that Chun strongly suggests. It therefore confirms the routine nature of combining multi-head CNNs for detection and classification tasks.

### 3. Shanmugamani's Explicit Disclosure of Conventional CNN Computer Vision Techniques.

318. Shanmugamani is a technical reference detailing TensorFlow and Keras implementations for core computer-vision tasks such as object detection, classification, and pose estimation for applications "including autonomous driving" EX1010, 7. Shanmugamani teaches how to construct and train deep-learning models for "person detection, face recognition, product search, medical image segmentation, … and video classification." EX1010, Preface. Although it does not focus on driver distraction specifically, it provides explicit TensorFlow examples for the same technical tasks performed by Chun's and Chao's prediction heads

319. Specifically, Shanmugamani provides expert techniques to train advanced neural networks using TensorFlow and Keras, as was used by Chun to implement NADS-Net, for applications "including autonomous driving." EX1010, 7. As described, the book was written for readers "who want[] to know how to apply deep learning computer vision problems such as classification, detection, retrieval, segmentation, generation, captioning, and video classification." Using the information provided by Shanmugamani "the reader should be able to develop

code for problems of person detection, face recognition, product search, medical image segmentation,… video classification, and so on." *Id*., Preface. Because Chun identifies TensorFlow as the implementation framework but does not describe the operational details of its architecture, a POSITA would have looked to contemporaneous TensorFlow literature, such as Shanmugamani, for the specific layer arrangements and training procedures used to implement those functions. A POSITA would have been motivated to use Shanmugamani's known TensorFlow techniques to improve Chun's TensorFlow-implemented CNN in the same way those techniques are used in other computer-vision systems, *i.e.*, to streamline implementation and ensure training stability, thereby yielding predictable performance improvements without changing the network's fundamental operation. Further, applying known TensorFlow routines to an existing CNN architecture would have been an "obvious to try" solution and a predictable use of known methods to achieve a familiar result, namely, the implementation of standard CNN layer operations within the disclosed TensorFlow environment.

**4. Motivation to Combine**

320. It would be obvious to a POSITA to combine Chun, Chao, and Shanmugamani because Shanmugamani provides explicit instructions on how to perform the tasks described by the prediction heads of Chun and Chao. Since Chun is implemented using TensorFlow, a POSITA would simply utilize the well-known

and standard TensorFlow techniques for image classification and detection described in Shanmugamani to perform the pose estimation, object detection, and distraction classification task-heads for distracted driving described by Chun and Chao. This implementation would not require an inventive step by a POSITA and would be technically straightforward to implement in TensorFlow, as revealed by Shanmugamani.

321. Because Chun was designed to utilize vision-based sensing technologies in order to "achieve the optimal performance for driver/passenger state detection tasks," a POSITA would consider applying standard computer vision techniques to in-vehicle systems as they utilize image data to "monitor and detect in-cabin activities." EX1005, 1. A POSITA that intends to perform driver state image classification would observe that the CNN of Chun was implemented in TensorFlow and understand that the conventional computer vision techniques described in Shanmugamani would be the most conventional and straightforward method for performing image classification and detection on images of a driver.

322. A POSITA would have reasonable expectation of success when utilizing the techniques described in Shanmugamani to perform the task-heads described in Chun and Chao because Shanmugamani provides explicit instructions on how to implement the exact tasks of pose estimation, object detection, and image classification and explicitly discusses the use of these tasks for computer

vision in self-driving vehicles. EX1010, 23. Specifically, Shanmugamani is written such that a person "will learn how to train deep learning models for computer vision applications and deploy them on multiple platforms," including computer vision applications for classification and detection in autonomous vehicles. EX1010, 7.

323. As shown by Shanmugamani, the '243 patent does not describe any new CNN techniques, but rather describes conventional approaches to image classification, object detection, and pose estimation that were taught well before the filing date of the '243 patent.

**B. Claim 3: The method of claim 1, wherein the object detection prediction head comprises a bounding box regression network and an object class prediction network, wherein the bounding box regression network outputs coordinates of a bounding box enclosing a detected object and the object class prediction network outputs a class corresponding to the detected object.**

324. Chun in view of Chao and further in view of Shanmugamani renders this claim obvious.

325. Dependent claim 3 depends from claim 1 and as I previously discussed, Chun in view of Chao renders claim 1 obvious. While the combination of Chun and Chao does not expressly disclose a bounding box regression and object class prediction network, these features would have also been obvious in

view of Shanmugamani. These components were well-known and conventional in CNN-based object-detection systems, as explicitly taught by Shanmugamani.

**1. Chun**

326. As discussed above for Ground 1, Chun already contemplates an object detection prediction head. Chun establishes that object detection in IVMS using multi-stage CNNs is a well-known and conventional approach to seat belt detection. The seat belt detection head, as described in Chun, provides detection results that "provide more information on, not only use and non-use of seat belt, but also proper/improper use cases judged by the relative position of seat belt to the detected body position." EX1005, 3.

327. While Chun's seatbelt detection head detects an object class ("seat belt") and localizes it in image space via a dense probability map, it is functionally identical to the detection performed by bounding-box heads in architectures such as Faster R-CNN or SSD, both of which Chun expressly cites as analogous methods. *Id*., 2-3. In contemporary CNN literature, object segmentation is understood as a pixel-level generalization of object detection: both produce per-object localization outputs using convolutional prediction heads attached to a shared backbone. Thus, Chun's seat-belt detection head is a conventional object-detection head differing only in representation format (mask vs. box), not in technical substance. A POSITA would therefore have recognized that the seat belt

detection head may be implemented to detect objects using any well-known, conventional method for object detection using the CNN architecture provided by Chun.

**2. Chao**

328. Chao expressly discloses using a neural network that detects bounding boxes for objects using a neural network that "may include any number of layers" and the "training of the neural network (e.g., adapting the layers of the neural network) may use or may be based on any kind of training principle." EX1006, ¶[0046]. Specifically, Chao's perception system 302 monitors an internal environment of a vehicle and "may implement one or more machine learning models" to "determine, for each object/feature, state data that describes a current state of such object as described," which may be "an estimate of the object's… size/footprint (e.g., as represented by a bounding shape such as a bounding polygon or polyhedron)." *Id*., ¶[0075]. Using this state data, a classification component 408 may "classify each of the object/feature instances from a predetermined set of objects/features." *Id*., ¶[0083].

329. While Chao does not expressly teach bounding box regression and object class prediction networks for an object detection head, Chao utilizes bounding boxes to classify and track objects, which is the purpose of Chun's seat belt detection head. Accordingly, a POSITA would have looked to well-known

TensorFlow object-detection techniques to implement these functions within the architecture disclosed by Chun and Chao, and Shanmugamani provides those implementation details.

### 3. Shanmugamani

330. Shanmugamani provides explicit disclosure of techniques for performing object detection, which include bounding box regression and object class prediction. These teachings include the use of activation, batch-normalization, and backpropagation layers in TensorFlow for training standard CNNs to detect objects in images. Shanmugamani describes a neural network as "a collection of perceptrons and activation functions," wherein "inputs of *x* are passed through a hidden layer of perceptrons and summed to the output." EX1010, 12. Shanmugamani provides the standard architecture of CNNs and the conventional functions and layers that are utilized for training a neural network, including activation, batch normalization, backpropagation, pooling layers, hidden layers, fully connected (or dense) layers, and convolutional layers. As described, "CNNs have weights, biases, and outputs through nonlinear activation" that are trained to "detect features or patterns in images." *Id*., 18.

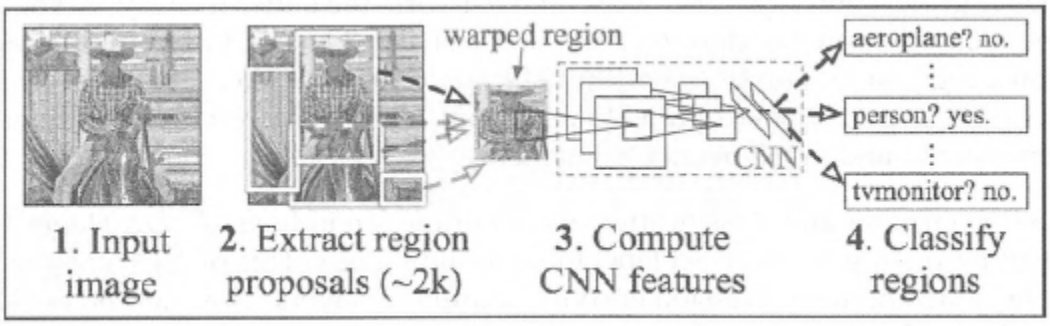
331. Using these conventional techniques, Shanmugamani provides methods for designing and implementing object detection CNNs using bounding boxes and object class prediction. As described, "[o]bject detection is the act of

finding the location of an object in an image" by "find[ing] all the objects within the image along with the labels." *Id*., 109-110. In order to describe object detection, Shanmugamani first provides description of localization, which is a fundamental principle of object detection and is an extension of image classification wherein "an image is passed through several layers of a CNN," and "the final layer of CNN outputs the probabilistic value, belonging to each of the labels." *Id*., 114. The "way to think about localization is modeling the problem as a regression problem," wherein the "bounding box is four numbers and hence can be predicted in a direct manner with a setting for regression." *Id*., 117. Within this localization, there is also a "need to predict the label, which is a classification problem." *Id*. In light of the need to predict both bounding boxes and their labels, "the regression encoder and classification encoder function independently." *Id*., 119. In order to perform localization, "[t]here are different parameterizations available to define the bounding boxes," which includes a "center of the coordinates with the height and width of the bounding box." *Id*., 117. Thus, Shanmugamani teaches the standard dual-head object-detection structure claimed in claim 3.

332. Using these principles, Shanmugamani provides "several variants of object detection algorithms" including R-CNN, Fast R-CNN, Faster R-CNN, Single shot multi-box detector (SSD), and YOLO. *Id.*, 120. Each of these

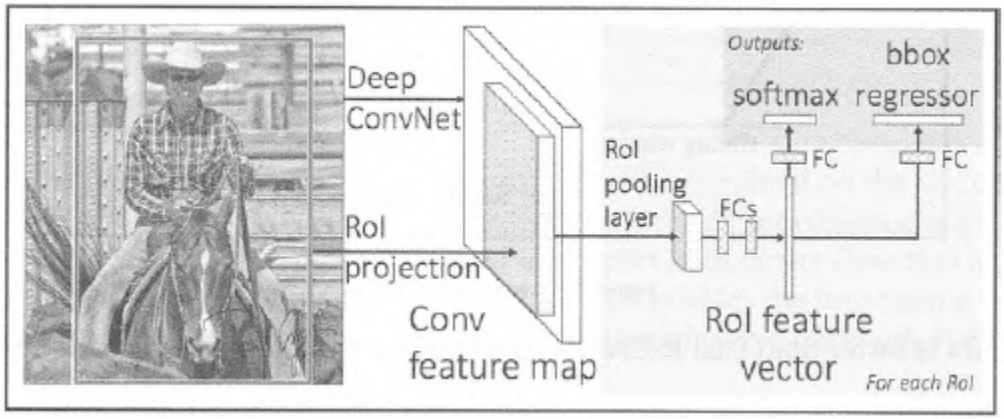
algorithms provide a CNN architecture that determines bounding boxes and classifies the objects in the bounding boxes.

333. For example, R-CNN uses selective search to propose "regions by grouping the color/texture of windows of various sizes." These region proposals are passed through a standard CNN architecture, wherein a "linear regression model to predict a closer bounding box is trained with object region proposals" and the "last layer of the CNN is trained with an SVM identifying the object with a no-object class." *Id*., 120. The architecture of R-CNN is as follows:



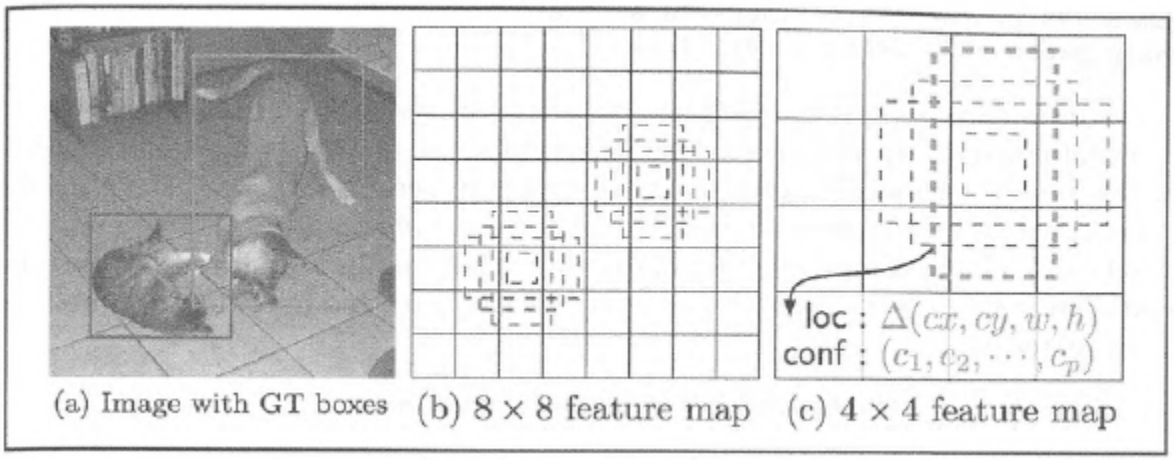

334. As illustrated, the R-CNN object detection technique includes a bounding box linear regression network and an object classification network (*e.g.*, classifying the regions as aeroplane, person, or tv monitor).

335. In another example, Fast R-CNN uses Region of Interest (ROI) pooling that "takes the CNN features and pools them together according to the regions" in order to "propose the networks and select the bounding box" *Id*., 121. The architecture of Fast R-CNN is as follows:




336. As illustrated, the Fast R-CNN object detection technique includes both a softmax layer and a bounding box regressor. With the combination of "one-hot encoding with softmax… multi-class classification becomes possible." *Id*., 13.

337. In another example, SSD "simultaneously predicts the object and finds the bounding box," (*Id*., 122-123) such that the bounding boxes are labeled based on the object, as shown below:




338. In view of these and other object detection techniques, Shanmugamani provides instructions for "constructing, training, and deploying object detection

models" using the "pre-trained models with various algorithms trained on the COCO dataset for public use" that are "built on top of TensorFlow." *Id*., 123. Additionally, Shanmugamani provides instructions on how to "retrain a model for our custom dataset," which involves "the preparation of a dataset, selecting the algorithm, and performing fine-tuning." *Id*., 126. As an example, Shanmugamani describes steps for training a pedestrian detection for a self-driving car using the YOLO object detection algorithm which includes a pooling layer, convolutional layer, and a dense (or fully connected) layer that include activation and cross-entropy functions.

339. Based on this disclosure in Shanmugamani, a POSITA would understand that an object detection CNN comprising a bounding box regressor and an object class prediction network are fundamental principles of object detection. Additionally, it is standard for the bounding box regressor to output coordinates of the bounding box and the object class prediction network to output a class of the object, as taught in Shanmugamani.

**4. MTC**

340. **How:** Because Chun expressly discloses a TensorFlow-implemented CNN but omits the operational details of how its detection head is structured and trained, a POSITA would have looked to Shanmugamani for those implementation details. As Chun implements a seat belt detection prediction head, or simply an

object detection head, comprising multiple convolutional layers using TensorFlow, a POSITA would understand that any standard technique for performing seat belt detection as described in Chun and Chao may be used. This may include the bounding polygon determination and classification as described in Chao. A POSITA would simply look to Shanmugamani to provide explicit instructions for how to structure and train the object detection head of Chun in order to detect a driver donning a seat belt. Additionally, Shanmugamani describes using the COCO dataset to pre-train for object detection, which is the dataset used by Chun to train the NADS-Net architecture. A POSITA may easily implement the methods and code provided by Shanmugamani to train the seat belt detection head of Chun, with the only change to the training being the object detection head detecting a seat belt, which is a standard and conventional alteration a POSITA would understand how to implement since Chun already provides training data for seatbelt detection using the COCO dataset and custom dataset.

341. **Why**: A POSITA would be motivated to combine Chun, Chao, and Shanmugamani because Shanmugamani teaches the conventional techniques and code for training CNNs to perform object detection as described in Chun and Chao. Since Chun establishes that a CNN prediction head may perform seat belt detection, a POSITA would look to examples of how to best implement the object detection head in the language (*i.e.*, Python) and platform (*i.e.*, TensorFlow)

utilized by Chun. Shanmugamani provides this explicit instruction, including instruction and code in Python for implementing bounding box regression and object class prediction for in-vehicle detection systems using TensorFlow. Since Chun's seat belt head was implemented in TensorFlow, a POSITA would turn to Shanmugamani as it describes the conventional approach for designing an object detection model in TensorFlow. No inventive step would be necessary to perform the object detection described in Chun by using bounding box regression and object class prediction, rather a POSITA would understand the bounding box regression and object class prediction described in Shanmugamani is a straightforward and optimal method for implementing Chun's seat belt detection prediction head.

342. **Reasonable Expectation of Success**. A POSITA would have reasonable expectation of success when utilizing the techniques described in Shanmugamani to perform the seat belt detection head described in Chun because Shanmugamani provides explicit instructions on how to implement the exact tasks of object detection using TensorFlow, which is an end-to-end, open-source platform for building, training, and deploying deep learning models. Specifically, Shanmugamani is written such that a person "will learn how to train deep learning models for computer vision applications and deploy them on multiple platforms," including computer vision applications for object detection in autonomous

vehicles. EX1010, 7. A POSITA would understand that coding a prediction head to perform object detection using bounding boxes with TensorFlow as described in Shanmugamani is an obvious and conventional technique for seat belt detection and would be successful since Chun's CNN is already implemented in TensorFlow.

343. Accordingly, for the same reasons articulated above in Ground 3, Chun in view of Chao and further in view of Shanmugamani renders claim 3 obvious, as the combination predictably yields an object prediction head comprising a bounding box regression and object class prediction network.

### C. Claim 4: The method of claim 3, wherein each of the bounding box regression network and the object class prediction network comprise deep neural networks having a plurality of hidden layers, each hidden layer in the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and a batch activation layer.

344. Chun in view of Chao and further in view of Shanmugamani renders this claim obvious.

345. Dependent claim 4 depends from claim 3 and as I previously discussed, Chun in view of Chao and further in view of Shanmugamani renders claim 3 obvious. While the combination of Chun and Chao does not expressly disclose a bounding box regression and object class prediction network comprising

a convolutional layer, a batch normalization layer, and a batch activation layer, these features would have also been obvious in view of Shanmugamani.

**1. Chun**

346. As discussed above in Claim 3, Chun provides an object detection prediction head that may be implemented to detect seat belts, which any well-known, conventional method for object detection may be used, including bounding box regression. As described in Chun, this seat belt detection head may employ "two 3x3 convolutions and a 1x1 convolution to predict a pixel-wise probability distribution." EX1005, 5. While Chun does not expressly disclose that these convolutional layers are hidden layers that comprise convolutional, activation, and batch normalization layers, these are well-known and conventional layers that are used by POSITA to implement and train CNNs, as shown by Chao and Shanmugamani.

**2. Chao**

347. As discussed above in claim 3, Chao provides disclosure of a CNN comprising "an input layer, an output layer, and one or more hidden layers between the input and output layers," that may use bounding polygons to classify objects, which a POSITA would understand as a conventional method for object detection. EX1006, ¶¶[0075], [0098]. While Chao does not expressly disclose that these hidden layers comprise convolutional, activation, and batch normalization layers,

these are well-known and conventional layers that are used by POSITA to implement and train CNNs, as shown by Shanmugamani.

**3. Shanmugamani**

348. Shanmugamani provides explicit disclosure of the techniques and implementation for bounding box regression and object class prediction networks, including the use of convolutional, batch activation, and batch normalization layers. It is widely understood that an artificial neural network "is a collection of perceptrons and activation functions," wherein "inputs of *x* are passed through a hidden layer of perceptrons and summed to the output" and "[e]very layer can have one of the activation functions" as described in Shanmugamani. EX1010, 12. Activation functions "make neural nets nonlinear" and during training, activation functions "play an important role in adjusting the gradients." *Id*., 8. These activation functions include sigmoid, hyperbolic tangent, and rectified linear unit (ReLU) functions. *Id*., 7-9. Batch normalization "increase[s] the stability and performance of neural network training," as it "reduces overfitting and makes the network train faster" by "normaliz[ing] the output from a layer with zero mean and a standard deviation of 1." *Id*., 14.

349. Shanmugamani provides instructions on how to "retrain a model for our custom dataset," which involves "the preparation of a dataset, selecting the algorithm, and performing fine-tuning." *Id*., 126. As an example, Shanmugamani

describes the conventional model for training a pedestrian detection for a self-driving car using the YOLO object detection algorithm, which includes a pooling layer, convolutional layer, and a dense (or fully connected) layer that include convolutional, activation, and regularization layers to avoid overfitting. The example code is as follows:

```
def pooling_layer(input_layer, pool_size=[2, 2], strides=2,
padding='valid'):
    layer = tf.layers.max_pooling2d(
        inputs=input_layer,
        pool_size=pool_size,
        strides=strides,
        padding=padding
    )
    add_variable_summary(layer, 'pooling')
    return layer
```

```
def convolution_layer(input_layer, filters, kernel_size=[3, 3],
padding='valid',
                      activation=tf.nn.leaky_relu):
    layer = tf.layers.conv2d(
        inputs=input_layer,
        filters=filters,
        kernel_size=kernel_size,
        activation=activation,
        padding=padding,
        weights_initializer=tf.truncated_normal_initializer(0.0, 0.01),
        weights_regularizer=tf.l2_regularizer(0.0005)
    )
    add_variable_summary(layer, 'convolution')
    return layer

def dense_layer(input_layer, units, activation=tf.nn.leaky_relu):
    layer = tf.layers.dense(
        inputs=input_layer,
        units=units,
        activation=activation,
        weights_initializer=tf.truncated_normal_initializer(0.0, 0.01),
        weights_regularizer=tf.l2_regularizer(0.0005)
    )
    add_variable_summary(layer, 'dense')
    return layer


yolo = tf.pad(images, np.array([[0, 0], [3, 3], [3, 3], [0, 0]]),
name='pad_1')
yolo = convolution_layer(yolo, 64, 7, 2)
yolo = pooling_layer(yolo, [2, 2], 2, 'same')
yolo = convolution_layer(yolo, 192, 3)
yolo = pooling_layer(yolo, 2, 'same')
yolo = convolution_layer(yolo, 128, 1)
yolo = convolution_layer(yolo, 256, 3)
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = pooling_layer(yolo, 2, 'same')
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
```

```
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = convolution_layer(yolo, 512, 1)
yolo = convolution_layer(yolo, 1024, 3)
yolo = pooling_layer(yolo, 2)
yolo = convolution_layer(yolo, 512, 1)
yolo = convolution_layer(yolo, 1024, 3)
yolo = convolution_layer(yolo, 512, 1)
yolo = convolution_layer(yolo, 1024, 3)
yolo = convolution_layer(yolo, 1024, 3)
yolo = tf.pad(yolo, np.array([[0, 0], [1, 1], [1, 1], [0, 0]]))
yolo = convolution_layer(yolo, 1024, 3, 2)
yolo = convolution_layer(yolo, 1024, 3)
yolo = convolution_layer(yolo, 1024, 3)
yolo = tf.transpose(yolo, [0, 3, 1, 2])
yolo = tf.layers.flatten(yolo)
yolo = dense_layer(yolo, 512)
yolo = dense_layer(yolo, 4096)

dropout_bool = tf.placeholder(tf.bool)
yolo = tf.layers.dropout(
        inputs=yolo,
        rate=0.4,
        training=dropout_bool
    )
yolo = dense_layer(yolo, output_size, None)
```

EX1010, 128-130.

350. As illustrated, the model includes a plurality of hidden layers wherein each hidden layer has convolutional layer (input_layer), an activation layer (activation=tf.nn.leaky_relu), and a regularization layer (weights_regularizer=tf.l2_regularizer(0.0005)). While the example model uses a L2 regularization layer to reduce overfitting, overfitting can be solved by

"[r]egularizing using techniques such as dropout and batch normalization," which are simple functions provided by TensorFlow. *Id*., 77.

351. Thus, Shanmugamani provides explicit disclosure of conventional methods and techniques for implementing an object detection head using bounding box regression and object classification networks that include hidden layers comprising convolutional, activation, and batch normalization layers.

**4. MTC**

352. **How:** Because Chun discloses a TensorFlow-based CNN with convolutional layers but omits layer-level operational details, a POSITA would have looked to Shanmugamani for the standard TensorFlow implementation of hidden layers. Shanmugamani teaches that CNN hidden layers conventionally include convolutional, activation (ReLU), and batch-normalization operations to stabilize training. As Chun implements a seat belt detection prediction head, or simply an object detection head, comprising multiple convolutional layers using TensorFlow, a POSITA would understand that any standard CNN techniques for performing seat belt detection as described in Chun and Chao may be used. This may include a CNN comprising "two 3x3 convolutions and a 1x1 convolution to predict a pixel-wise probability distribution," (EX1005, 5.) or hidden layers that determine bounding polygons and classify objects as described in Chao. A POSITA would simply look to Shanmugamani to provide explicit instructions for

how to structure and train the CNN for the object detection described in Chun in order to detect a driver donning a seat belt, which includes the conventional utilization of convolutional, activation, and batch normalization layers. A POSITA may easily implement the object detection CNN provided by Shanmugamani to model the seat belt detection head of Chun, as it is a standard and conventional model.

353. **Why**: A POSITA would be motivated to combine Chun, Chao, and Shanmugamani because Shanmugamani teaches the conventional techniques and code for training CNNs to perform object detection as described in Chun and Chao. Since Chun establishes that a prediction head may perform seat belt detection using a plurality of convolutional layers, a POSITA would look to examples of how to best implement the object detection head in the language (*i.e.*, Python) and platform (*i.e.*, TensorFlow) utilized by Chun. Shanmugamani provides this explicit instruction, including instruction and code in Python for implementing convolutional, activation, and batch normalization layers for bounding box regression and object class prediction for in-vehicle detection systems using TensorFlow. Since Chun's seat belt head was implemented in TensorFlow, a POSITA would turn to Shanmugamani as it describes the conventional techniques for designing an object detection model in TensorFlow. No inventive step would be necessary to perform the object detection described in Chun by using the

described layers, but rather a POSITA would understand the convolutional, activation, and batch normalization layers described in Shanmugamani provide a straightforward and optimal method for implementing the CNN for Chun's seat belt detection prediction head. Further, applying these known TensorFlow operations to Chun's CNN represents a predictable use of known techniques to improve a similar system in the same way, yielding improved stability and faster convergence without altering function.

354. **Reasonable Expectation of Success**. A POSITA would have reasonable expectation of success when utilizing the techniques described in Shanmugamani to perform the seat belt detection head described in Chun because Shanmugamani provides explicit instructions on how to implement the exact tasks of object detection using TensorFlow, which is an end-to-end, open-source platform for building, training, and deploying deep learning models. Specifically, Shanmugamani is written such that a person "will learn how to train deep learning models for computer vision applications and deploy them on multiple platforms," including computer vision applications for object detection in autonomous vehicles. EX1010, 7. A POSITA would understand that coding a CNN with TensorFlow to perform bounding box regression and object class prediction using convolutional, activation, and batch normalization layers as described in Shanmugamani is an obvious and conventional technique for seat belt detection

and would be successful since Shanmugamani provides conventional techniques for Chun's stated purpose of detecting a seat belt. Further, Shanmugamani provides explicit TensorFlow code and guidance for training such CNN architectures, giving a POSITA high confidence of success.

355. Together, Chun in view of Chao and further in view of Shanmugamani describe an object prediction head that utilizes bounding box regression and object class prediction networks that include hidden layers comprising convolutional, activation, and batch normalization layers.

356. Accordingly, for the same reasons articulated above in Ground 3, Chun in view of Chao and further in view of Shanmugamani renders claim 4 obvious, as the combination predictably yields an object prediction head comprising a convolutional layer, activation layer, and batch normalization layer.

### D. Claim 5: The method of claim 1, wherein the pose estimation prediction head includes a plurality of hidden layers and an output layer, each of the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and an activation layer, and the output layer comprising a convolutional layer.

357. Chun in view of Chao and further in view of Shanmugamani renders this claim obvious.

358. Dependent claim 5 depends from claim 1 and, as I previously discussed, Chun in view of Chao renders claim 1 obvious. While the combination

of Chun and Chao does not expressly disclose a pose estimation head comprising a convolutional layer, a batch normalization layer, and an activation layer, and the output layer comprising a convolutional layer, these features would have also been obvious in view of Shanmugamani.

#### 1. Chun-Chao

359. As discussed above for claim [1F], Chun already contemplates a pose estimation prediction head. Both the key point and part-affinity field (PAF) prediction heads, as disclosed by Chun, perform "2D driver/passenger pose estimation." EX1005, 1. Each of the "detection branches employs two 3×3 convolutions and a 1×1 convolution to predict a pixel-wise probability distribution." EX1005, 5. Thus, Chun discloses a pose-estimation branch (keypoint and PAF heads) implemented in TensorFlow and composed of multiple convolutional layers but does not spell out the layer-level operational details. While Chun-Chao does not expressly disclose that the convolutional layers include a batch normalization layer, an activation layer, and an output layer comprising a convolutional layer, these are conventional and well-known techniques for optimal CNN performance for pose estimation.

#### 2. Shanmugamani

360. Shanmugamani provides the missing, conventional TensorFlow implementation details for CNN hidden layers used in vision tasks (including

pose). Shanmugamani provides specific disclosure of CNNs for pose estimation that comprise a convolutional layer, a batch normalization layer, an activation layer, and an output layer comprising a convolutional layer. It is widely understood that an artificial neural network "is a collection of perceptrons and activation functions," wherein "inputs of *x* are passed through a hidden layer of perceptrons and summed to the output" and "[e]very layer can have one of the activation functions" as described in Shanmugamani. EX1010, 12. Activation functions "make neural nets nonlinear" and during training, activation functions "play an important role in adjusting the gradients." *Id*., 8. These activation functions include sigmoid, hyperbolic tangent, and rectified linear unit (ReLU) functions. *Id*., 7-9. Batch normalization "increase[s] the stability and performance of neural network training," as it "reduces overfitting and makes the network train faster" by "normaliz[ing] the output from a layer with zero mean and a standard deviation of 1." *Id.*, 14.

361. Using these conventional techniques, Shanmugamani provides methods for designing and implementing pose estimation CNNs using localization. Shanmugamani describes localization as an extension of image classification wherein "an image is passed through several layers of a CNN," and "the final layer of CNN outputs the probabilistic value, belonging to each of the labels." *Id*., 114. Shanmugamani describes that object localization may be used for pose detection,

which is "the act of finding joint locations in a human," by finding "multiple locations such as head, neck, shoulders, ankles, and hands," *Id*., 117-118, as shown below:



EX1010, 117.

362. This object localization "detects one object in an image within a label" by "predict[ing] several cropped portions of an image with an object" and regressing image coordinates using a regression encoder and classification encoder that function independently. *Id*., 110, 114, 119.

363. Shanmugamani provides instructions on how to "retrain a model for our custom dataset," which involves "the preparation of a dataset, selecting the algorithm, and performing fine-tuning." *Id*., 126. As an example, Shanmugamani describes the conventional model for training a pedestrian detection for a self-driving car using the YOLO object detection algorithm, which includes a pooling layer, convolutional layer, and a dense (or fully connected) layer that include

convolutional, activation, and regularization layers to avoid overfitting. The example code is as follows:

```
def pooling_layer(input_layer, pool_size=[2, 2], strides=2,
padding='valid'):
    layer = tf.layers.max_pooling2d(
        inputs=input_layer,
        pool_size=pool_size,
        strides=strides,
        padding=padding
    )
    add_variable_summary(layer, 'pooling')
    return layer


def convolution_layer(input_layer, filters, kernel_size=[3, 3],
padding='valid',
                      activation=tf.nn.leaky_relu):
    layer = tf.layers.conv2d(
        inputs=input_layer,
        filters=filters,
        kernel_size=kernel_size,
        activation=activation,
        padding=padding,
        weights_initializer=tf.truncated_normal_initializer(0.0, 0.01),
        weights_regularizer=tf.l2_regularizer(0.0005)
    )
    add_variable_summary(layer, 'convolution')
    return layer

def dense_layer(input_layer, units, activation=tf.nn.leaky_relu):
    layer = tf.layers.dense(
        inputs=input_layer,
        units=units,
        activation=activation,
        weights_initializer=tf.truncated_normal_initializer(0.0, 0.01),
        weights_regularizer=tf.l2_regularizer(0.0005)
    )
    add_variable_summary(layer, 'dense')
    return layer
```

```
yolo = tf.pad(images, np.array([[0, 0], [3, 3], [3, 3], [0, 0]]),
name='pad_1')
yolo = convolution_layer(yolo, 64, 7, 2)
yolo = pooling_layer(yolo, [2, 2], 2, 'same')
yolo = convolution_layer(yolo, 192, 3)
yolo = pooling_layer(yolo, 2, 'same')
yolo = convolution_layer(yolo, 128, 1)
yolo = convolution_layer(yolo, 256, 3)
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = pooling_layer(yolo, 2, 'same')
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
```

```
yolo = convolution_layer(yolo, 256, 1)
yolo = convolution_layer(yolo, 512, 3)
yolo = convolution_layer(yolo, 512, 1)
yolo = convolution_layer(yolo, 1024, 3)
yolo = pooling_layer(yolo, 2)
yolo = convolution_layer(yolo, 512, 1)
yolo = convolution_layer(yolo, 1024, 3)
yolo = convolution_layer(yolo, 512, 1)
yolo = convolution_layer(yolo, 1024, 3)
yolo = convolution_layer(yolo, 1024, 3)
yolo = tf.pad(yolo, np.array([[0, 0], [1, 1], [1, 1], [0, 0]]))
yolo = convolution_layer(yolo, 1024, 3, 2)
yolo = convolution_layer(yolo, 1024, 3)
yolo = convolution_layer(yolo, 1024, 3)
yolo = tf.transpose(yolo, [0, 3, 1, 2])
yolo = tf.layers.flatten(yolo)
yolo = dense_layer(yolo, 512)
yolo = dense_layer(yolo, 4096)

dropout_bool = tf.placeholder(tf.bool)
yolo = tf.layers.dropout(
        inputs=yolo,
        rate=0.4,
        training=dropout_bool
    )
yolo = dense_layer(yolo, output_size, None)
```

EX1010, 128-130.

364. As illustrated, the model includes a plurality of hidden layers wherein each hidden layer has convolutional layer (input_layer), an activation layer (activation=tf.nn.leaky_relu), and a regularization layer (weights_regularizer=tf.l2_regularizer(0.0005)). While the example model uses a L2 regularization layer to reduce overfitting, overfitting can be solved by "[r]egularizing using techniques such as dropout and batch normalization," which

are simple functions provided by TensorFlow. *Id*., 77. Additionally, the final output layer is a dense layer that comprises a convolutional layer as its input layer is a convolutional layer.

365. Thus, Shanmugamani provides explicit disclosure of conventional methods and techniques for implementing a pose estimation head that includes hidden layers comprising convolutional, activation, and batch normalization layers and an output layer that comprises a convolutional layer.

**3. MTC**

366. **How:** Because Chun identifies a TensorFlow pose estimation head but omits the layer-level operational details, a POSITA would look to Shanmugamani for the standard TensorFlow implementation (convolutional → batch normalization → activation) per hidden layer and a convolutional output producing heatmaps. As Chun implements a pose estimation prediction head comprising multiple convolutional layers using TensorFlow, a POSITA would understand that any standard CNN techniques for performing pose estimation as described in Chun may be used. This may include a CNN comprising "two 3x3 convolutions and a 1x1 convolution to predict a pixel-wise probability distribution," EX1005, 5 or hidden layers that determine bounding polygons and classify objects as described in Chao. A POSITA would simply look to Shanmugamani to provide explicit instructions for how to optimally structure and train the CNN for the object

detection described in Chun in order to detect a driver donning a seat belt, which includes the conventional utilization of convolutional, activation, and batch normalization layers and an output layer comprising a convolutional layer. A POSITA may easily implement the pose estimation CNN provided by Shanmugamani to model the pose estimation head of Chun, as it is a standard and conventional model.

367. **Why**: A POSITA would be motivated to combine Chun, Chao, and Shanmugamani because Shanmugamani teaches the conventional techniques and code for training CNNs to perform pose estimation as described in Chun and Chao. Since Chun establishes that a prediction head may perform pose estimation using a plurality of convolutional layers, a POSITA would look to examples of how to best implement the pose estimation head in the language (*i.e.*, Python) and platform (*i.e.*, TensorFlow) utilized by Chun. Shanmugamani provides this explicit instruction, including instruction and code in Python for implementing convolutional, activation, and batch normalization layers for pose estimation for in-vehicle detection systems using TensorFlow. Since Chun's pose estimation head was implemented in TensorFlow, a POSITA would turn to Shanmugamani as it describes the conventional techniques for designing a pose estimation model in TensorFlow. No inventive step would be necessary to perform the pose estimation described in Chun by using the described layers, but rather a POSITA would

understand the convolutional, activation, batch normalization, and output layers described in Shanmugamani provide a straightforward and optimal method for implementing the CNN for Chun's pose estimation prediction head.

368. **Reasonable Expectation of Success**. A POSITA would have reasonable expectation of success when utilizing the techniques described in Shanmugamani to perform the pose estimation head described in Chun because Shanmugamani provides explicit instructions on how to implement the exact tasks of pose estimation using TensorFlow, which is an end-to-end, open-source platform for building, training, and deploying deep learning models. Specifically, Shanmugamani is written such that a person "will learn how to train deep learning models for computer vision applications and deploy them on multiple platforms," including computer vision applications for autonomous vehicles. EX1010, 7. A POSITA would understand that coding a prediction head to perform pose estimation with TensorFlow as described in Shanmugamani is an obvious and conventional technique for pose estimation and would be successful since Shanmugamani provides conventional techniques for Chun's stated purpose of pose estimation.

369. Accordingly**,** for the same reasons articulated above in Ground 3, Chun in view of Chao and further in view of Shanmugamani renders claim 5 obvious, as the combination predictably yields a pose estimation head comprising a

convolutional layer, activation layer, batch normalization layer, and an output layer.

### E. Claims 15-17 and 21-23

370. Claims 15-17 and 21-23 recite the same substantive limitations as claims 3-5 but expressed in different statutory formats (computer-readable medium and device claims rather than method claims).

371. Claim 13 mirrors the method of claim 1, except that it is directed to "a non-transitory computer-readable storage medium" storing program instructions. Claims 15-17 then correspond directly to dependent claims 3-5, adding the same minor refinements such as an object detection head comprising bounding box regression and class object prediction networks (claim 15 vs. claim 3), the bounding box regression and class object prediction networks comprising specific hidden layers (claim 16 vs. claim 4), and pose estimation head comprising specific hidden layers (claim 17 vs. claim 5).

372. Likewise, claim 20 recites the same functional steps as claims 1 and 13 but in the context of a "device" with a processor and storage medium, and claims 21-23 track the same refinements as claims 3-5.

373. The only differences between claims 3-5 and claims 15-17 and 21-23 are the claim formats (method vs. storage medium vs. device).

374. A POSITA would have understood that the same disclosed training methods could be implemented as software instructions stored on a non-transitory medium (claims 15-17) or as logic executed on a processor within a device (claims 21-23). Chun in view of Chao discloses as much.

375. Specifically, Chao already describes implementing distraction classification through software and hardware modules, including in-vehicle sensors and/or camera 702, spatio-temporal system 704, and associated processors and storage. *See, e.g.*, EX1006, ¶¶[0092]-[0093]. Thus, recasting the same operations into "computer-readable medium" or "device" form involves no substantive difference, but merely reflects conventional statutory claiming techniques.

376. Accordingly, for the same reasons set forth with respect to claims 3-5, claims 15-17 and 21-23 are rendered obvious by Chun in view of Chao.

## X. GROUND 3: ZHENGYANG IN VIEW OF HE RENDERS OBVIOUS CLAIMS 1-6, 9-18, 20-24

377. Under Ground 3, Zhengyang in view of He discloses each limitation of independent claims 1, 13, 20, 26, and 30, which recite the method, computer-readable medium, and device aspects of the same driver distraction classification.

378. The dependent claims are also disclosed or, at a minimum, rendered obvious by Zhengyang in view of He.

### A. Motivation to Combine

379. A POSITA would have combined the teachings of Zhengyang and He, to arrive at the claimed invention by incorporating He's pose-estimation head into Zhengyang's shared-backbone, multi-task driver-state recognition network. Both references provide a complementary piece of the architecture: Zhengyang supplies the multi-task framework and He adds an additional prediction head operating on shared feature maps. Further, the motivation to combine these references would have been clear and strong. Zhengyang already aims to "comprehensively reflect the driver's driving state" and "improv[e] the accuracy of driver state recognition" through multi-task learning. EX1007, ¶[0046]. He demonstrates that adding a pose-estimation head to a shared backbone improves human-state recognition by capturing keypoints such as head tilt, shoulder rotation, and hand position, which are features directly relevant to detecting driver distraction. By the priority date, a POSITA would have recognized pose estimation as a routine and complementary technique that augments driver-monitoring performance. Finally, a POSITA would have had a reasonable expectation of success in making this combination because each reference employs the same underlying deep-learning components: ResNet-style backbones, convolutional heads, cross-entropy loss functions, and gradient-based optimization and uses a common training paradigm of joint loss minimization through backpropagation. Integrating He's pose-estimation head into

Zhengyang's multi-task framework would have required no alteration to core architecture or training logic, only the addition of a new task head with its own labeled keypoints and loss.

380. A POSITA would have found it obvious to combine the teachings of Zhengyang and He to arrive at the claimed invention because each reference addresses the same class of problems (*e.g.*, multi-task visual recognition and efficient inference within a shared-backbone neural-network architecture) and provides complementary, well-understood techniques that a skilled practitioner would naturally integrate to enhance system performance and deployability.

381. Zhengyang provides the foundation: a unified, end-to-end neural-network model for driver-state analysis that replaces multiple single-task classifiers with a single multi-head framework. It discloses a shared backbone coupled to multiple prediction heads, each trained via its own task loss (mask, eyeglasses, smoking, and phone use) and optimized jointly through forward and backward propagation until convergence. The goal is to "simultaneously recognize multiple features of the driver" and thereby "improve the accuracy of driver state recognition." EX1007, ¶¶[0003], [0005]. A POSITA would recognize this as a standard multi-task learning configuration, directly amenable to additional task heads.

382. He teaches precisely such an extension, adding a pose-estimation prediction head within the same backbone-plus-head framework. He explicitly discloses a "keypoint head" comprising convolutional and deconvolutional layers that predict one-hot keypoint masks for each joint type (*e.g.*, left shoulder, right elbow), trained jointly with detection and segmentation heads. EX1008, 2986-87. The model "simultaneously predict[s] boxes, segments, and keypoints," achieving real-time inference at 5 fps while sharing a common backbone. EX1008, 2987. A POSITA would view this as a directly analogous system: both references use shared feature maps to feed task-specific heads trained through multi-loss optimization.

383. The motivation to combine Zhengyang with He would have been straightforward and strong. Zhengyang already teaches multi-task distraction classification using spatial attention and localized regions of interest, and He demonstrates that adding a pose-estimation head within a shared backbone architecture yields complementary benefits, capturing human body geometry and keypoints that directly inform attention and distraction cues. By the priority date of the '243 patent, incorporating pose estimation into driver-monitoring systems was routine and expected to improve robustness, as pose features (*e.g.*, head tilt, shoulder rotation, hand position) provide fine-grained indicators of distraction. A POSITA, seeing Zhengyang's stated objective of "comprehensively reflect[ing] the

driver's driving state" and "improving the accuracy of driver state recognition" would have recognized that adding He's keypoint-based pose-estimation head was an obvious and compatible enhancement, merely another task branch drawing on shared spatial features, trained via its own loss, and integrated using the same backpropagation framework. EX1007, ¶[0046]. Both systems are directed to visual-recognition tasks involving human detection and posture interpretation, and thus constitute analogous art addressing the same problem space.

384. The combination would also have carried a reasonable expectation of success. Both references rely on standard deep-learning components (ResNet backbones, convolutional heads, cross-entropy losses, and gradient-based optimization). Integrating He's pose-estimation head into Zhengyang's backbone-plus-head network would have required no change to underlying training or data-flow mechanisms, only the addition of another task head with its own labeled keypoints and loss. Because Zhengyang already manages multiple attribute-specific heads through its OSME module, a POSITA would have expected such an addition to operate predictably and improve performance by providing richer spatial context for driver-state recognition.

385. In sum, combining Zhengyang's end-to-end multi-task driver-monitoring network with He's pose-estimation head would have been a predictable, complementary, and a conventional adaptation of known components.

Each reference operates within the same technological field of computer-vision-based human-state recognition, employs compatible architectures, and uses standard training and deployment workflows. A POSITA would have been motivated to integrate these elements to enhance recognition accuracy, leverage spatial pose features, and ensure deployability, thereby arriving at the claimed system through ordinary skill rather than any inventive leap.

### B. Independent Claim 1

#### 1. [1Pre] "A method comprising"

386. Zhengyang discloses this element.

387. Zhengyang describes a "multi-task attribute recognition method… for driving scenarios." EX1007, Summary. The method includes a technical solution that "acquires image data of a driver in a driving scenario; inputs the image data into a pre-trained multi-task network model for attribute recognition; the attribute recognition task of the multi-task network model comprises: mask recognition, eyeglasses recognition, smoking recognition, and phone call recognition; determining the driver's multi-task attribute recognition result based on the attribute recognition output result of the multi-task network model," which frames the steps of claim 1. *Id*., ¶[0066]. Using this method, the "multi-task network model can be used to simultaneously recognize multiple features of the driver and output an output result that can comprehensively reflect the driver's driving state,

thereby improving the accuracy and real-time performance of driver state recognition." *Id*.

**2. [1A] "inputting labeled examples into a neural network,"**

388. Zhengyang discloses this element.

389. A POSITA would understand from Zhengyang that the system performs supervised learning in which driver images, labeled with multi-task attributes, are fed into a neural network and used to optimize it. Zhengyang first lays out the end-to-end operational workflow, detailing both the inference pipeline and the corresponding training process: "Inputting the preprocessed image data into a pre-trained multi-task network model for attribute recognition and obtaining attribute recognition results of the multi-task network model for the target tasks of whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0012]. Zhenyang then describes how those inputs are received by the core network and routed to multiple task heads: "Inputting the preprocessed image data into the multi-task main network to obtain a feature map, using an OSEM module to extract the feature maps for different attribute recognition tasks, and inputting them into respective attribute recognition modules, wherein the attribute recognition modules are a mask wearing attribute recognition module, an eyeglasses wearing attribute

recognition module, a smoking attribute recognition module, and a phone call attribute recognition module;" EX1007, ¶[0020].

390. Zhengyang is explicit that the image examples provided to its network are labeled before training: "Labeling the sample image data with multi-task attributes;" EX1007, ¶[0025]. Zhenyang elaborates the same point in the detailed description: "It can be understood that before training the multi-task network model, it is necessary to label the driver based on whether he or she is wearing a mask or eyeglasses, smoking, or making a phone call… In this way, the labeling information and driver images will be used as input to train the model." EX1007, ¶[0063]. Together, these passages confirm that the examples used during training are annotated with ground-truth labels, and those labeled examples (*e.g.*, the driver images paired with attribute labels) are input into Zhengyang's neural network.

391. Zhengyang also provides architectural context showing how these labeled inputs are processed through a backbone-plus-heads multi-task network. Zhengyang states: "The collected and cropped RGB image is input into a backbone network to extract the encoded features of the image. Specifically, the backbone network uses the SE-ResNet18 network… These 7×7 feature maps are then processed through a one-squeeze multi-excitation (OSME) module to extract 4 different types of feature maps…" EX1007, ¶[0069]. This aligns the input pathway

(preprocessed, labeled images) with the network components (backbone feature extractor, OSME, and task-specific heads) that consume those inputs.

392. Zhengyang reiterates the same core step elsewhere to describe the training/inference loop: "[I]nputting the preprocessed image data into a pre-trained multi-task network model for attribute recognition and obtaining attribute recognition results of the multi-task network model for the target tasks…" EX1007, ¶[0076], and again underscores the training set preparation: "Labeling the sample image data with multi-task attributes;" EX1007, ¶[0080]. Importantly. for supervised learning, Zhengyang explains the one-to-one coupling between each input image and its label during optimization: "The preprocessed sample image data is matched one-to-one with the labeled multi-task attributes and arranged in order; the preprocessed sample image is input into the network, the final output is compared one-to-one with the labeling result according to the loss function, and the difference is reduced by repeated forward and backward propagation." EX1007, ¶[0090]. Zhengyang then restates the training pipeline step in the "training process" section: "Inputting the preprocessed image data into the multi-task main network to obtain a feature map, using an OSEM module to extract the feature maps for different attribute recognition tasks, and inputting them into respective attribute recognition modules…" EX1007, ¶[0094].

393. Finally, Zhengyang identifies the supervision signal used to train the network's outputs against the labels: "Specifically, the labeling information of driver attributes (mask state, eyeglasses state, smoking state, phone call state) is used to train the multi-task driver attribute recognition task using focal loss." EX1007, ¶[0121]. In sum, Zhengyang repeatedly and expressly discloses that labeled examples (driver images with attribute labels) are input into a neural network (backbone with task-specific heads), and that training proceeds by comparing network outputs to those labels with a loss (*e.g.*, focal loss) and updating the model via forward/backpropagation, which is analogous to the claimed "inputting labeled examples into a neural network."

**3. [1B] "the neural network comprising: a backbone network,"**

394. Zhengyang discloses this element.

395. A POSITA would understand that Zhengyang identifies a backbone as the entry point that first receives driver images and extracts generic, shared features for downstream tasks. Zhengyang states: "In this solution, the collected RGB image can be input into a backbone network to extract the encoded features of the image; a facial recognition neural network for driving scenarios is used to analyze the encoded features. If above a set threshold, the face frame coordinate value closest to the previous result is given; if below a set threshold, it directly outputs: no driver, and ends; the face frame coordinates are proportionally

enlarged, the input image is cropped, and the cropped image is normalized and resized." EX1007, ¶[0060]. A POSITA would recognize this as the classical role of a backbone feature extractor in modern CNN pipelines: shared feature encoding followed by subsequent decision logic. Zhengyang makes clear that the same class of images, cropped driver face images captured in the vehicle cabin, is used both during model training and during runtime inference. Paragraph [0060] explains that the collected RGB image is "input into a backbone network to extract encoded features," establishing the role of the backbone as the input interface for driver images. Later, paragraph [0083] reiterates that "the initial architecture of the multi-task network model uses the SE-ResNet18 network" and that "[t]hese 7×7 feature maps are then processed through an OSME module" to train the attribute-specific heads. A POSITA would understand that this training process involves inputting training data, *i.e.*, labeled images of drivers with annotated attributes such as mask, eyeglasses, smoking, and phone use, into the backbone network, as confirmed by paragraphs [0045] and [0080]-[0090], which describe labeling, forward and backward propagation, and loss optimization. Thus, the same "collected RGB images" introduced in [0060] are the training data fed into the backbone during model training.

396. Zhengyang then expressly names the backbone architecture and connects it to the rest of the multi-task network. Zhengyang explains: "The

collected and cropped RGB image is input into a backbone network to extract the encoded features of the image. Specifically, the backbone network uses the SE-ResNet18 network, and the output scale feature map is 7×7. These 7×7 feature maps are then processed through a one-squeeze multi-excitation (OSME) module to extract 4 different types of feature maps, namely, mask, eyeglasses, smoking, and phone call feature maps, which are used to determine the features of different attribute recognition tasks." EX1007, ¶[0069]. Technically, this establishes that (i) the backbone = SE-ResNet18, (ii) its output is a 7×7 feature map, and (iii) those shared features are distributed to task-specific branches via OSME, which is exactly how a "backbone-plus-heads" multi-task network is architected.

397. Zhengyang reiterates this system-level architecture when introducing the initial model design used during training: "Accordingly, the initial architecture of the multi-task network model uses the SE-ResNet18 network, and the output scale feature map is a 7×7 feature map. These 7×7 feature maps are then processed through a one-squeeze multi-excitation (OSME) module to extract 4 different types of feature maps, namely, mask, eyeglasses, smoking, and phone call feature maps, which are used to determine the features of different attribute recognition tasks." EX1007, ¶[0083]. A POSITA would understand that this paragraph ties the backbone choice (SE-ResNet18) to the training configuration, confirming that the backbone is a core, shared component of the claimed multi-task network.

398. Zhengyang also provides implementation-level details for the backbone, demonstrating that SE-ResNet18 is not an abstract label, but the concrete network used to produce the shared features. It explains: "It is well known that there are many different network structures, and the required network structure can be selected. The multi-task network structure used here is SE-ResNet18, and the initial architecture is adjusted. For example, the input size is set to 112×112, the 7×7 convolution kernel (stride=2) of the first convolution layer is replaced with a 3×3 convolution kernel (stride=1), the first maximum pooling layer is deleted, the stride of the second convolution layer of the first resnet block stage is changed to 2, and in each residual block, a batch normalization layer and dropout are added. The structure is BN layer, Prelu, convolution, BN layer, Prelu layer, convolution, and dropout layer, and the dropout ratio of the dropout layer in the residual block is 0%-20%." EX1007, ¶[0100]. These implementation-level modifications, such as adjusting kernel sizes, removing the pooling layer, changing strides, and adding batch normalization, Prelu, and dropout layers, demonstrate that SE-ResNet18 is not merely referenced in theory but is the concrete backbone architecture actually configured and used to extract shared features within Zhengyang's multi-task network.

399. Finally, Zhengyang shows the operational flow in its training schematic, again placing the backbone at the point where cropped driver images

enter the shared encoder before task recognition. It states: "FIG. 3 is a schematic diagram of training process of a multi-task network model … When the value is higher than a set threshold, the face frame closest to the previous result is cropped. After the cropped image is input into the backbone network, the mask attributes, eyeglasses attributes, smoking attributes, and phone call attributes can be recognized." EX1007, ¶[0109]. To a POSITA, this confirms that, even at runtime, the system first routes images into the backbone and then branches into task-specific heads.

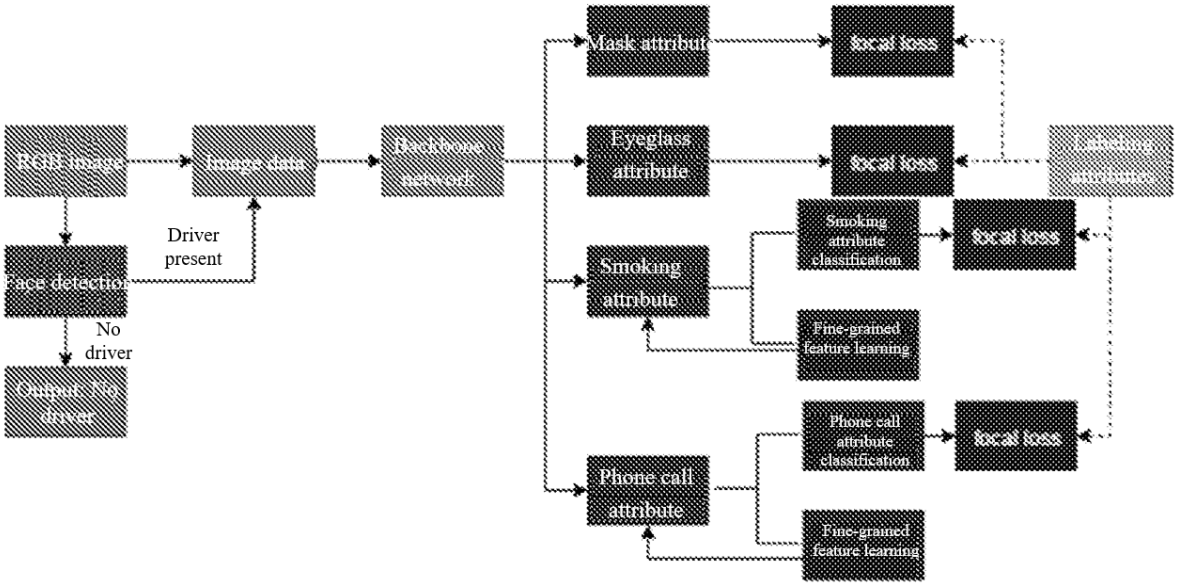



FIG. 3

**EX1007, FIG. 3**

400. Once again, while Zhengyang's paragraph [0060] illustrates the operational flow when in use, Zhengyang elsewhere explains that the same

collected driver images form the training data fed into the backbone during model training. *See* EX1007, ¶¶[0045], [0080]-[0083], [0089]-[0090]. A POSITA would understand that the "input image" in paragraph [0060] is representative of the same input modality and data type used in training, the labeled, preprocessed driver images. Thus, Zhengyang's disclosure of inputting the cropped driver image into the backbone network satisfies the claimed step of "inputting the training data into the neural network comprising a backbone network," because the backbone's input stage performs that function both during supervised training and later inference.

#### 4. [1C] "one or more prediction heads coupled to the backbone network,"

401. Zhengyang discloses this element.

402. In my view, this limitation is satisfied by any single qualifying head. The phrase "one or more prediction heads" sets the count, and the phrase "comprising one or more of" the three listed categories means that the set can be any subset of those three. A POSITA would therefore understand that the claim does not require all three head types. One head of one type is enough.

403. A POSITA would recognize that Zhengyang discloses a multi-task neural network with multiple prediction heads coupled to a shared backbone, each responsible for a distinct driver-state recognition task. The disclosure makes clear that, after shared feature extraction by the backbone, the network routes task-

specific feature maps to dedicated attribute recognition modules. Zhengyang teaches: "Inputting the preprocessed image data into the multi-task main network to obtain a feature map, using an OSEM module to extract the feature maps for different attribute recognition tasks, and inputting them into respective attribute recognition modules, wherein the attribute recognition modules are a mask wearing attribute recognition module, an eyeglasses wearing attribute recognition module, a smoking attribute recognition module, and a phone call attribute recognition module." EX1007, ¶[0020].

404. These modules constitute prediction heads, each receiving backbone-generated features and producing a task-specific output. Zhengyang immediately confirms this by explaining that "the output of the entire network is a feature map of each task and an attribute recognition result of each task… whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]. A POSITA would recognize these as discrete classification heads. Zhengyang next describes the functional structure of these heads and explicitly discloses a distraction classification head and an object detection prediction head.

**5. [1D] Distraction classification head**

405. A POSITA would read Zhengyang and see that it squarely includes distraction-focused prediction heads, most plainly for smoking and phone use,

within a single multi-task network. Zhengyang sets the stage by enumerating the tasks handled by the model: "the attribute recognition task of the multi-task network model comprises: mask recognition, eyeglasses recognition, smoking recognition, and phone call recognition." EX1007, ¶[0008]. Those tasks are not abstract labels; they are implemented as distinct downstream modules that receive shared features from the backbone and produce per-task decisions. As Zhengyang explains, the network "us[es] an OSEM module to extract the feature maps for different attribute recognition tasks, and input[s] them into respective attribute recognition modules, wherein the attribute recognition modules are a mask wearing attribute recognition module, an eyeglasses wearing attribute recognition module, a smoking attribute recognition module, and a phone call attribute recognition module." EX1007, ¶[0020]; *see also Id.*, ¶[0094].

406. Zhengyang further confirms the nature of these modules as classification heads by describing their explicit outputs. The network's head-level predictions are "a feature map of each task and an attribute recognition result of each task, and the output results of the attribute recognition of each task are whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]; and it specifies the label categories: "the number of output categories of the attribute recognition tasks of whether a mask is worn, whether there is smoking, and whether a phone call is

being made is two categories." *Id.*, ¶[0028]; *see also Id.*, ¶[0095], ¶[0098]. In other words, smoking and phone-call recognition are implemented as binary classification heads, *i.e.*, distraction classification heads attached to the shared representation.

407. For the more challenging distraction attributes, Zhengyang discloses the internal structure of these heads and how they refine their decisions. Each of the smoking and phone call heads "comprises a subject recognition module and an attention module; the subject recognition module is used for attribute recognition to output the result of whether there is smoking and whether a phone call is being made; the attention module comprises an attention proposal sub-network and an attention network, which are used for attribute recognition and optimization of the attribute recognition results of the main network." EX1007, ¶¶[0031]-[0033]. The attention proposal sub-network functions as a detection-style module that produces localization parameters identifying regions of interest: it "output[s] the center point coordinates of the attention area and the edge lengths of the attention area to perform cropping and enlargement processing on the original input image." EX1007, ¶[0034]; *see also* ¶[0101]. The refined crop is then re-analyzed in successive stages: "The enlarged image will be input into the first-level attention model to re-judge the attributes, and the APN generated by the first-level attention model will further crop and enlarge the image and input it into the second-level

attention model to re-judge the attributes." *Id.*, ¶[0102]; "The first-level attention neural network model and the second-level attention neural network model here have the same network structure as the main network model and will output attribute recognition results as described above."*Id.*, ¶[0103].

408. Finally, Zhengyang's description of how the model is trained confirms that the different modules (like mask, eyeglasses, smoking, and phone-call recognition) are true prediction heads. Zhengyang teaches "comparing the attribute recognition results to the labeling results, reducing the difference through the corresponding loss function to optimize network parameters" and "adjusting the weights of the loss function to optimize the training results of the multi-task network model." *Id.*, ¶¶[0022]- [0023]; *see also* ¶[0096]-¶[0097]. It further identifies "the attribute classification loss generated by the four attribute recognition tasks," weighted per task difficulty. *Id.*, ¶[0084]. Taken together, these disclosures show that Zhengyang implements smoking and phone-use modules as task-specific classification heads fed by the backbone, trained from their own outputs, and strengthened with attention-based refinement, which is analogous to the claimed "distraction classification prediction head."

### 6. [1E] Object Detection Prediction Head

409. Zhengyang also discloses an object detection head within the same multi-task network architecture, integrated alongside the distraction classification

heads. A POSITA would recognize that the attention proposal sub-network (APN) described in Zhengyang functions as an object detection prediction head because it localizes and delineates salient regions of interest, specifically, areas corresponding to objects such as cigarettes or phones associated with driver distraction, using boundingbox coordinates rather than categorical labels alone. Rather than producing only categorical class labels, the APN predicts spatial parameters defining the position and size of detected objects, consistent with standard object detection heads used in deep learning architectures.

410. Zhengyang explains that "the attention module comprises an attention proposal sub-network and an attention network, which are used for attribute recognition and optimization of the attribute recognition results of the main network." EX1007, ¶[0033]. Zhengyang then details the internal design of this sub-network: "[T]he attention proposal sub-network mainly comprises two fully connected layers in series, which are used to output the center point coordinates of the attention area and the edge lengths of the attention area to perform cropping and enlargement processing on the original input image." *Id.*, ¶[0034]. These outputs (center coordinates and edge lengths) are precisely the standard parameters used in object detection systems to define bounding boxes around detected entities, as would be understood by a POSITA.

411. Further confirming its detection-oriented role, Zhengyang describes that the APN generates localized regions to be re-analyzed by higher-level attention models, stating that "the attention proposal sub-network comprises two fully connected layers in series… to perform cropping and enlargement processing on the feature map." EX1007, ¶[0101]. This operation mirrors the region proposal stage of conventional object detection networks (*e.g.*, Faster R-CNN), in which the network identifies candidate regions that likely contain relevant objects before performing fine-grained classification. The cropped region is then passed to additional attention models that refine recognition results through hierarchical spatial focus. "The enlarged image will be input into the first-level attention model to re-judge the attributes… and the APN generated by the first-level attention model will further crop and enlarge the image and input it into the second-level attention model to re-judge the attributes." *Id.*, ¶[0102].

412. From the standpoint of a POSITA, this iterative attention and cropping process is unmistakably object detection functionality: Zhengyang first detects and isolates objects within an image (the attention proposal sub-network), then passes these localized regions to secondary modules for refined classification. The APN's structure and output type (predicting coordinates and boundingbox extents) demonstrate that Zhengyang's network includes an object detection prediction head integrated within the broader multi-task framework. This detection

head cooperates with the distraction classification heads to jointly identify both the presence and location of distraction-related objects in real time, enabling a unified end-to-end perception model.

413. Accordingly, Zhengyang discloses not only classification heads that determine whether a driver is smoking or using a phone, but also an explicit object detection head that outputs spatial localization data for those same distraction cues, further confirming that its architecture encompasses multiple specialized prediction heads operating in parallel within a shared backbone framework.

### 7. [1F] Pose Estimation Prediction Head

414. While Zhengyang expressly discloses multiple prediction heads, specifically, the distraction classification heads for smoking and phone use and the object detection prediction head embodied by the attention proposal sub-network, it does not expressly disclose a pose estimation prediction head. However, it would have been obvious to add a pose estimation prediction head to Zhengyang's architecture.

415. Although Zhengyang focuses its implementation on attribute recognition and localized attention mechanisms, Zhengyang's design readily accommodates more spatially expressive tasks. The disclosed prediction heads (covering mask wearing, eyeglasses, smoking, and phone use) operate as classification and localization modules trained to identify specific driver-related

features. As Zhengyang explains, "the output results of the attribute recognition of each task are whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]. These outputs demonstrate a well-structured multi-task network optimized for discrete driver attributes, with attention-based refinement modules that "input the enlarged image into the first-level attention model to re-judge the attributes."*Id.*, ¶[0102].

416. While these modules are described in the context of attribute verification, the underlying architecture, featuring dense spatial feature maps, attention-guided cropping, and iterative reanalysis, provides the same representational foundation necessary for estimating human pose. In other words, Zhengyang's multi-task framework already captures spatial cues from the driver's body and face, and extending those cues to predict pose-related quantities such as keypoints or joint orientations would have been a logical evolution of the same design principles. Indeed, a POSITA would have readily understood how to extend Zhengyang's system to include a pose-estimation head as explicitly taught in He.

**(a) He**

417. He et al. (Mask R-CNN) explicitly discloses a pose-estimation prediction head implemented as a distinct neural branch for keypoint detection within its unified, multi-head convolutional architecture. Section 5 of He's paper, titled "Mask R-CNN for Human Pose Estimation," describes the framework's

direct extension to pose estimation: "Our framework can easily be extended to human pose estimation. We model a keypoint's location as a one-hot mask, and adopt Mask R-CNN to predict *K* masks, one for each of *K* keypoint types (e.g., left shoulder, right elbow)." EX1008, 2986. He further explains that "the keypoint head architecture is similar to that in Figure 3 (right). The keypoint head consists of a stack of eight 3×3 512-d conv layers, followed by a deconv layer and 2× bilinear upscaling, producing an output resolution of 56×56." *Id*. This architecture is trained using a dedicated loss function, "minimiz[ing] cross-entropy loss over an $m^2$-way softmax output" for each visible keypoint and evaluated by its person keypoint AP ($AP_{kp}$) score. *Id*.

418. He et al. further confirm that this keypoint head operates as a parallel prediction branch, trained alongside detection and segmentation heads, within the same backbone network: "We have a unified model that can simultaneously predict boxes, segments, and keypoints while running at 5 fps," and "adding a segment branch … improves $AP_{kp}$," while adding the keypoint branch slightly reduces box/mask AP, suggesting that while keypoint detection benefits from multi-task training, it does not in turn help the other tasks. EX1008, 2987. He also presents a dedicated ablation study demonstrating that the keypoint branch improves overall performance when trained jointly with other heads (Table 5).

419. A POSITA would readily recognize from these disclosures that He teaches a pose-estimation prediction head: a specialized convolutional sub-network configured to predict spatial heatmaps for human body joints from shared backbone features. This head is trained with its own loss term ($L_{keypoints}$) and outputs one heatmap per anatomical keypoint. He thus provides an express disclosure of a pose-estimation prediction head integrated within a multi-branch, shared-backbone framework. The architecture demonstrates that pose estimation was an interchangeable, well-understood extension of Mask R-CNN's prediction heads, confirming that multi-task frameworks of this type supported keypoint-based pose-estimation heads.

**(b) MTC**

420. While Zhengyang does not expressly disclose a pose estimation prediction head, it provides a multi-task neural architecture that is structurally and functionally compatible with one. Zhengyang's system already incorporates a shared convolutional backbone (SE-ResNet18) and multiple specialized prediction heads for driver attributes such as mask wearing, eyeglasses, smoking, and phone use. Each of these heads receives shared feature maps and produces task-specific outputs (categorical attributes or localized bounding boxes) optimized through a multi-task loss. As Zhengyang explains, "the output results of the attribute recognition of each task are whether a mask is worn, whether eyeglasses are worn,

whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]. It further teaches that its attention proposal sub-network generates region crops that are reanalyzed iteratively: "the enlarged image will be input into the first-level attention model to re-judge the attributes." *Id.*, ¶[0102]. This combination of shared features, task-specific heads, and attention-guided refinement provides precisely the architectural framework on which a pose-estimation head could operate.

421. He et al. provides a concrete, enabling example of such an additional head. He expressly discloses a pose-estimation prediction head implemented as a distinct neural branch for keypoint detection. As He explains, "Our framework can easily be extended to human pose estimation. We model a keypoint's location as a one-hot mask, and adopt Mask R-CNN to predict *K* masks, one for each of *K* keypoint types (*e.g.*, left shoulder, right elbow)." EX1008, 2986. The paper further describes the internal design of this keypoint head: "The keypoint head consists of a stack of eight 3×3 512-d conv layers, followed by a deconv layer and 2× bilinear upscaling, producing an output resolution of 56×56." *Id*. He also confirms that this keypoint head operates as a parallel prediction branch trained jointly with other heads: "We have a unified model that can simultaneously predict boxes, segments, and keypoints while running at 5 fps." *Id.*, 2987. He's unified framework shows

that pose estimation, segmentation, and detection can be trained from a shared backbone under separate task-specific losses without architectural interference.

422. A POSITA would have been motivated to combine He's pose-estimation head with Zhengyang's multi-task network because the combination would predictably enhance Zhengyang's ability to infer the driver's physical state and attentiveness, which are core objectives of Zhengyang's system. Zhengyang already localizes and classifies driver attributes based on facial and hand regions; adding He's pose-estimation head would allow the same system to infer the driver's body orientation, arm position, or head posture, thereby enabling a more complete assessment of distraction and readiness. The addition would have been technically straightforward: both architectures employ a shared backbone feeding multiple prediction heads, and He's keypoint head is expressly designed to attach as a modular branch that consumes shared feature maps. Because Zhengyang's backbone produces dense spatial representations of the driver's face and upper body, a POSITA would have recognized that the same feature maps could serve as input to a keypoint-regression head without altering the network's fundamental design.

423. Further, the motivation to combine would have been reinforced by well-established design principles in multi-task learning. He demonstrates that the Mask R-CNN architecture can "simultaneously predict boxes, segments, and

keypoints," thereby improving efficiency through shared feature extraction. *Id.* A POSITA would have understood that integrating a pose-estimation head into Zhengyang's existing backbone would yield similar efficiency and functional benefits, leveraging shared convolutional features to produce additional behavioral cues (pose) that complement distraction-related classifications. The combination would therefore have been an obvious, predictable extension yielding an improved, more comprehensive driver-monitoring network consistent with Zhengyang's stated goal "to comprehensively reflect the driver's driving state and improve the accuracy of driver state recognition." EX1007, ¶[0046].

424. Accordingly, it would have been obvious to a POSITA to extend Zhengyang's backbone-plus-heads framework by incorporating He's keypoint-based pose-estimation prediction head. Both references teach modular, multi-task networks trained under separate loss functions; both rely on dense convolutional feature maps shared across tasks. He provides an explicit blueprint for implementing a pose-estimation head using nearly identical architectural conventions, and Zhengyang provides the motivating application context (*i.e.*, driver distraction detection) where body-pose inference offers clear functional benefit. The combination would have been a predictable and routine design choice within the knowledge of one of ordinary skill in the art at the time.

425. A POSITA would have considered Zhengyang and He to be analogous references addressing closely related problems within the same field of human-centered computer vision. Zhengyang focuses on recognizing driver attributes and behaviors, such as smoking, phone use, and gaze direction, by analyzing body and facial regions in cabin imagery. He addresses the structurally similar problem of recognizing and localizing human keypoints (*e.g.*, elbows, shoulders, joints) from image data, using the same type of convolutional feature representations and detection-head architecture. Both systems are directed to neural-network-based multi-task perception of human pose or behavior from visual input, rely on shared backbones feeding task-specific heads, and optimize under multi-branch loss functions. Because both references lie squarely within the domain of deep-learning-based human detection and activity recognition, a POSITA would have naturally looked to He when seeking to improve Zhengyang's system with additional spatial or behavioral inference capabilities.

426. Furthermore, He's disclosure is not only analogous but directly complementary to Zhengyang's stated design goals. Zhengyang's architecture seeks "to comprehensively reflect the driver's driving state and improve the accuracy of driver state recognition." EX1007, ¶[0046]. He's pose-estimation head provides exactly that type of information (*e.g.*, fine-grained spatial cues describing body posture and limb orientation) which would enhance Zhengyang's ability to

infer driver distraction, drowsiness, or readiness. Thus, a POSITA would have found it entirely natural to apply He's human-pose head within Zhengyang's existing multi-head backbone to achieve a more holistic model of driver state.

427. A POSITA also would have had a reasonable expectation of success in making this combination. Both references use convolutional backbones that output dense spatial feature maps suitable for keypoint regression. He explicitly demonstrates that its pose-estimation head can be added to an existing detection network without disrupting other branches, stating that "we have a unified model that can simultaneously predict boxes, segments, and keypoints while running at 5 fps." EX1008, 2987. Similarly, Zhengyang's network already includes multiple heads drawing from shared features, proving its compatibility with additional branches. The modification would therefore have been straightforward: the keypoint head from He could be connected to Zhengyang's shared feature maps and trained with labeled keypoints using a dedicated loss term, in the same manner as He's implementation.

428. No technical obstacles would have discouraged a POSITA from making this combination/modification. He's pose-estimation head is modular, uses standard convolutional and deconvolutional layers, and is trained via common loss functions (cross-entropy or mean-squared error) already employed in Zhengyang. Because both systems use comparable feature resolutions, convolutional block

depths, and loss formulations, integration would not require redesign of the backbone or attention modules. A POSITA would have reasonably expected that combining He's pose-estimation head with Zhengyang's existing distraction-classification and object-detection heads would yield predictable success, producing a unified model capable of detecting both what the driver is doing and how the driver's body is oriented, thereby improving recognition accuracy consistent with Zhengyang's objectives.

#### 8. [1G] "minimizing a loss based on outputs of the one or more prediction heads causing a change in parameters of the backbone network"

429. Zhengyang alone or in combination with He renders this element obvious.

430. At the outset, as I stated earlier, to the extent any claim-construction argument is raised on this point, I am not opining that claim element [1G] requires a "joint loss" or that multiple head-loss terms must be combined into a single scalar objective. In my opinion, the ordinary reading of [1G] requires only that the loss being minimized is "based on outputs of the one or more prediction heads" and that this loss minimization causes a change in the backbone parameters. That language is broad enough to cover (a) loss computed from a single head, (b) loss computed from multiple heads, or (c) loss computed by any aggregation function

of the outputs (for example, a sum, weighted sum, or simple selection of one head's loss).

**(a) Zhengyang**

431. Zhengyang expressly discloses minimizing a loss based on the outputs of its prediction heads in a manner that causes corresponding changes to the parameters of the shared backbone network. A POSITA would understand that Zhengyang teaches standard end-to-end supervised training, in which task-specific losses computed from head outputs are propagated backward through the network, via gradient descent, thereby updating the shared backbone parameters.

432. Zhengyang first explains that each prediction head produces attribute recognition results that are compared to labeled ground truth, and that the resulting loss is used to optimize the entire network: "Comparing the attribute recognition results to the labeling results, reducing the difference through the corresponding loss function, and inputting the feature maps of each task into the corresponding loss function to optimize network parameters; Adjusting the weights of the loss function to optimize the training results of the multi-task network model and reduce the training time." EX1007, ¶¶[0022]-[0023]. A POSITA would recognize that "optimize network parameters" refers to gradient-based backpropagation, through which losses computed at the prediction heads produce gradient signals that flow through the intermediate layers and into the shared backbone. Because

each task's "feature maps" originate from the backbone, minimizing these losses necessarily alters the backbone's parameters.

433. Zhengyang next describes this process at the multi-task level, explicitly linking each head's classification loss to global parameter updates: "Determining the attribute classification loss generated by the four attribute recognition tasks … and reduc[ing] these differences through forward and back propagation to the multi-task network model. These attribute classification losses are multiplied by different weights, depending on the difficulty of calculating the attribute features." EX1007, ¶[0084]. A POSITA would understand that this describes the standard training loop, in which head-level losses (forward pass) propagate backward to update shared parameters (backward pass), including those of the backbone.

434. Beyond classification loss, Zhengyang applies a metric loss to intermediate attention feature maps derived from the backbone: "Four sets of 77 attention feature maps … are input into the metric loss function together … to make the attention feature maps of whether there is smoking, whether a phone call is being made, whether a mask is worn, and whether eyeglasses are worn mutually exclusive features." EX1007, ¶¶[0085]-[0089]. Because these attention feature maps are derived from the backbone's features, minimizing this loss sends

gradients back through the attention modules into the backbone, improving how the backbone separates features for each task.

435. Zhengyang repeatedly emphasizes that the entire network (*e.g.*, heads, attention modules, and backbone) is optimized through iterative forward and backward propagation until convergence. It explains: "The preprocessed sample image … is compared one-to-one with the labeling result according to the loss function, and the difference is reduced by repeated forward and backward propagation. Finally, the network converges … and the parameters in the network are the required final results." EX1007, ¶[0090]. This confirms that all network parameters, including those of the backbone, are adjusted during loss minimization. The same sequence is reiterated at ¶¶[0096]-[0097], underscoring that optimization based on head-derived losses is a core part of the disclosed training method.

436. Zhengyang further specifies auxiliary loss components (focal loss, Lrank loss, and a total weighted loss) each computed from prediction-head outputs and refined attention regions. Zhengyang states: "Loss weights are adjusted according to the attribute recognition results of the at least two tasks … Fine-grained classification loss … can be softmax or softmax cross entropy, etc. In this case, the focal loss is used. … The Lrank loss function … make[s] the classification effect of the second-level attention image better than… the first-level

attention image." EX1007, ¶¶[0117]-[0120]. Zhengyang then defines a composite objective: "loss = $W_1$loss_$f_1$ + $W_2$loss_$f_1$ + $W_3$(loss_$f_1$ + Lrank) + $W_4$(loss_$f_1$ + Lrank) + $W_5$Lnp." *Id.*, ¶[0128]. A POSITA would understand that these losses are minimized jointly during training and that their gradients propagate through the shared feature maps, thereby updating the backbone.

437. Taken together, Zhengyang clearly teaches minimizing losses computed from the outputs of the prediction heads (*e.g.*, classification, metric, focal, and ranking losses) through forward and backward propagation that optimizes all network parameters, including those of the shared backbone. Zhegnyang's repeated references to "optimizing network parameters," "forward and back propagation," and "network convergence" collectively demonstrate the element [1D].

438. Further, even if not express in Zhengyang, because He's pose-estimation head operates under the same end-to-end learning framework disclosed in Zhengyang, a POSITA would have found it obvious to apply the same loss-minimization process of Zhengyang to that additional head of He.

**(b) He**

439. He explicitly teaches that its keypoint head is trained using a dedicated cross-entropy loss on predicted keypoint heatmaps, with gradients propagated through the shared ResNet-FPN backbone to refine spatial

representations ("the training target is a one-hot m×m binary mask… we minimize the cross-entropy loss over an $m^2$-way softmax output"). EX1008, 2986.

**(c) MTC**

440. A POSITA would have recognized that integrating this He head into Zhengyang's multi-task framework would simply extend the existing loss pipeline, where each head's losses are jointly minimized and backpropagated, to include an additional loss term for pose estimation. The same optimization logic and gradient flow would naturally update the shared backbone parameters, yielding improved spatial awareness without altering Zhengyang's training method or network structure.

441. The motivation to combine and the reasoning for how and why He's pose estimation head would be incorporated into Zhengyang's framework are the same as discussed for element [1C]. Both Zhengyang and He operate within analogous multi-task convolutional architectures that use a shared backbone and multiple prediction heads trained through joint loss minimization. Zhengyang already teaches computing and minimizing task-specific losses from multiple heads to optimize shared parameters through forward and backward propagation, while He demonstrates that a pose estimation head can be trained in precisely the same manner, by minimizing a keypoint-specific loss function and propagating its gradients through the shared backbone.

442. A POSITA would therefore have recognized that adding He's pose estimation head to Zhengyang's architecture would not require any structural modification or novel training method. The new head would simply contribute an additional loss term (for keypoint prediction) to Zhengyang's existing multi-task objective. Because both systems rely on standard backpropagation across shared feature maps, the combined architecture would train end-to-end, with all head-specific losses, including that of the pose estimation head, jointly minimized to update the backbone. The motivation, as before, would be to enhance the system's ability to infer driver state and spatial awareness while leveraging the same proven optimization framework already taught by Zhengyang.

**9. [1H] "storing a distraction classification model after minimizing the loss, the distraction classification model comprising the parameters of the backbone network and parameters of at least one of the one or more prediction heads"**

443. Zhengyang discloses this claim element.

444. Zhengyang describes a supervised multi-task network trained end-to-end, where each task head contributes a separate loss term, and those loss signals propagate backward to refine the model parameters. Zhengyang explains that training proceeds by "comparing the attribute recognition results to the labeling results, reducing the difference through the corresponding loss function, and inputting the feature maps of each task into the corresponding loss function to

optimize network parameters; [and by] adjusting the weights of the loss function to optimize the training results of the multi-task network model and reduce the training time." EX1007, ¶¶[0022]-[0023].

445. Zhengyang confirms the same iterative update process later, stating that differences are reduced "through forward and back propagation to the multi-task network model." *Id*., ¶[0084]. Zhengyang further describes convergence: "the preprocessed sample image is input into the network, the final output is compared one-to-one with the labeling result according to the loss function, and the difference is reduced by repeated forward and backward propagation. Finally, the network converges … and the parameters in the network are the required final results." *Id.*, ¶[0090]. Zhengyang summarizes again: "Comparing the attribute recognition results to the labeling results, reducing the difference through the corresponding loss function … to optimize network parameters." *Id*., ¶¶[0096]-[0097].

446. A POSITA would understand that these passages describe standard multi-task optimization where per-head losses generate gradients that update all trainable layers along the forward graph, including the shared backbone network and the task-specific prediction heads. Zhengyang therefore describes minimizing loss functions until a set of final backbone parameters and final prediction-head parameters are produced.

447. Zhengyang also discusses post-training model preparation for runtime use. Zhengyang explains: "Selecting the network architecture and model with the best attribute analysis results based on test accuracy; Deleting the network that does not participate in the final attribute recognition output and its parameters to reduce the network model size and forward transmission time." *Id*., ¶¶[0038]-[0039]. These passages describe selection of a best-performing converged model and preparation of a reduced inference-ready architecture.

448. Zhengyang proceeds to describe execution from a memory device: "a computer-readable storage medium having stored thereon a computer program…" and "a device comprising a memory … the processor implementing the multi-task attribute recognition method…" *Id.*, ¶¶[0044]-[0045]. Zhengyang describes the non-volatile memory medium in detail, including flash memory. *Id*., ¶[0141].

449. A POSITA would understand that this set of disclosures collectively describes the conventional workflow where final trained parameters of the backbone and the deployed prediction head or heads are stored in non-volatile memory for runtime inference.

### C. Claim 2: The method of claim 1, wherein the distraction classification prediction head includes a convolutional layer, pooling layer, and fully connected layer.

450. Zhengyang discloses or renders obvious claim 2.

451. Zhengyang discloses, or at minimum renders obvious, that each distraction-classification prediction head, including the smoking-recognition and phone-call-recognition heads, includes convolutional, pooling, and fully connected layers. This conclusion is supported by (1) explicit identification of SE-ResNet18 as the backbone network (EX1007, ¶[0100]), (2) repeated statements that the attention networks "have the same network structure as the main network model" (*Id.*, ¶[0103]), and (3) detailed layer-type descriptions of both the backbone and attention proposal sub-networks. *Id.*, ¶¶[0100]-[0101], [0103], [0109].

**1. Convolutional-Layer Element.**

452. A POSITA would understand that Zhengyang expressly, or at minimum renders obvious, that each distraction-classification prediction head, including the smoking-recognition and phone-call-recognition heads, includes at least one convolutional layer. This conclusion follows from three mutually reinforcing disclosures in Zhengyang:

453. (1) Zhengyang defines its backbone network as SE-ResNet18, which is a convolutional network; (2) Zhengyang states that the attention sub-networks within the distraction heads "have the same network structure as the main network model"; and (3) Zhengyang describes that main network's convolutional configuration in detail.

### 2. Zhengyang Defines the Main Network as SE-ResNet18, a Convolutional Architecture.

454. Zhengyang identifies SE-ResNet18 as the backbone network. SE-ResNet18 is a well-known deep convolutional neural network that uses stacked convolutional blocks with batch-normalization and rectified linear unit activations. Zhengyang confirms this understanding stating:

455. "It is well known that there are many different network structures, and the required network structure can be selected. The multi-task network structure used here is SE-Resnet 18, and the initial architecture is adjusted. For example, the input size is set to 112*112, the 7*7 convolution kernel (stride=2) of the first convolution layer is replaced with a 3*3 convolution kernel (stride=1), the first maximum pooling layer is deleted, the stride of the second convolution layer of the first resent block stage is changed to 2, and in each residual block, a batch normalization layer and dropout are added. The structure is BN layer, Prelu, convolution, BN layer, Prelu layer, convolution, and dropout layer, and the dropout ratio of the dropout layer in the residual block is 0%-20%." EX1007, ¶[0100].

456. A POSITA would understand that Zhengyang explicitly discloses the presence of convolutional layers based on its identification and detailed configuration of an SE-ResNet18 backbone. By stating that "the multi-task network structure used here is SE-ResNet18," Zhengyang identifies a well-known

convolutional neural network architecture that performs image analysis through successive convolutional operations. Zhengyang goes further to describe specific modifications to those convolutional layers, noting that "the 7*7 convolution kernel (stride=2) of the first convolution layer is replaced with a 3*3 convolution kernel (stride=1)" and that "the stride of the second convolution layer of the first ResNet block stage is changed to 2." *Id*. These direct references to convolution kernels and stride adjustments confirm that convolutional layers are active computational elements within the network. Even though Zhengyang deletes the first maximum-pooling layer and introduces batch-normalization and dropout in each residual block, these adjustments preserve the convolutional framework rather than replace it. The disclosure reinforces that the convolutional layers form the fundamental feature-extraction mechanism of the architecture, responsible for detecting and encoding spatial patterns within the image before classification occurs. Accordingly, Zhengyang's definition of SE-ResNet18, its modification of the convolution kernel and stride parameters, and its reliance on residual blocks built from convolutional units collectively demonstrate that the described model includes convolutional layers.

457. Thus, Zhengyang makes clear that the backbone consists of successive convolutional layers that perform the core feature-extraction computations of the network. Further, a POSITA would understand that any component described as

having "the same network structure as the main network model" would also include convolutional layers.

### 3. The Attention Sub-Networks in the Distraction Heads Have the Same Structure.

458. Zhengyang explains that the smoking-and phone-use-recognition heads include attention sub-networks that refine the initial predictions by re-analyzing cropped regions of interest. Each head "comprises a subject-recognition module and an attention module." Zhengyang then states that "the first-level attention neural-network model and the second-level attention neural-network model here have the **same network structure** as the main network model and will output attribute-recognition results." EX1007, ¶¶[0031]-[0033], [0102]-[0103] (emphasis added).

459. Because the main network is SE-ResNet18, which is a convolutional neural network, the attention sub-networks that have the same structure also use convolutional layers. A POSITA would read this as a direct indication that each attention model within the distraction head employs the same convolutional block arrangement (for example, convolution, batch-normalization, activation, dropout) as the ResNet backbone.

#### 4. Functional Role of the Convolutional Layers within the Distraction Heads.

460. Zhengyang further states that the attention models "perform network analysis on the cropped and enlarged image and output a new result of whether there is smoking or whether a phone call is being made." EX1007, ¶[0035]. The phrase "network analysis on the cropped image," coupled with the disclosure that these models have the same structure as the main network, indicates that the attention models apply convolutional filters to extract spatial features from each region of interest. Convolutional layers are the standard components used to identify spatially localized patterns in image data and are particularly suited for attention-based refinement tasks in computer vision.

461. A POSITA would therefore understand that each distraction head includes one or more convolutional layers that process the cropped image region to detect visual cues such as a handheld phone or cigarette. This functional role corresponds to the convolutional layers in the distraction-classification prediction head of the '243 patent. Further, because the attention sub-networks are identified as "neural network models" that perform image analysis on pixel data, a POSITA would view convolutional layers as a fundamental design feature that enables such spatial analysis. Fully connected layers alone could not achieve localized spatial

filtering or maintain the two-dimensional image structure preserved by cropping and enlargement. EX1007, ¶[0103].

462. Finally, even if Zhengyang does not explicitly list every layer type within the attention models, a POSITA would recognize that a network described as having the same structure as SE-ResNet18 includes convolutional layers. SE-ResNet18 is universally recognized as an 18-layer convolutional residual network that alternates convolution, batch-normalization, activation, and pooling layers. Zhengyang's repeated statement that the attention models have the same structure amounts to an explicit incorporation of the convolutional architecture into each head. EX1007, ¶¶[0100]-[0103].

463. *In summary,* Zhengyang discloses the convolutional-layer component of the Claim 2 limitation through three complementary teachings:

The main network model is SE-ResNet18, a convolutional network ¶[0100];

The attention models within the distraction heads "have the same network structure as the main network model" ¶[0103]; and

The attention models perform "network analysis on the cropped and enlarged image" ¶[0035].

464. Accordingly, a POSITA would conclude that the smoking-and phone-use-recognition heads disclosed in Zhengyang include convolutional layers, satisfying the "convolutional layer" element of Claim 2 of the '243 patent.

**5. Disclosure of the Pooling-Layer Element.**

465. A POSITA would understand that Zhengyang expressly discloses and at least renders obvious pooling operations within its task-specific heads, including the smoking-and phone-use-recognition heads. Although Zhengyang states that "the first maximum-pooling layer is deleted, a POSITA would understand this only removes the initial pooling step to preserve higher feature resolution, not all pooling operations. EX1007, ¶[0100]. SE-ResNet18, as used in Zhengyang, still includes later down-sampling through stride-2 convolutions and a final global-average-pooling layer before the fully connected classifier, both standard pooling operations in CNNs. Because the attention networks "have the same network structure as the main network model", these pooling operations also appear in the distraction-classification heads. Thus, deleting the first max-pool layer refines early processing but does not eliminate pooling, and Zhengyang still discloses or renders obvious pooling layers within the claimed prediction head. *Id.*, ¶[0103]. Zhengyang confirms this understanding in additional parts of its disclosure. For example, Zhengyang identifies an explicit pooling step applied to the per-task feature maps produced by the OSME module and also indicates the presence of internal pooling operations within the attention sub-networks that share the SE-ResNet18 structure.

**6. Explicit Per-Task Pooling Prior to Head Input.**

466. Zhengyang states that after feature extraction and routing through the OSME module, "the 4 output results of this module are then applied to the previous feature map to obtain feature maps for the four tasks. After the four feature maps are pooled, the feature vectors are input into four attribute-recognition modules." EX1007, ¶[0101]. A POSITA would recognize this as a task-specific pooling step that converts each two-dimensional feature map into a fixed-length vector representing spatially aggregated information for that attribute. Because this pooled vector serves as the direct input to each classification head, the pooling layer is part of the head's computational architecture rather than a global preprocessing operation.

**7. Functional and Architectural Placement within the Head.**

467. In convolutional neural networks, the boundary between the backbone and the prediction head is flexible in implementation but well understood conceptually. Practitioners routinely define the head as the set of task-specific layers trained or fine-tuned on top of a pre-trained feature-extraction backbone. The precise transition point may vary across models, but this division of labor is standardized in deep-learning practice. . As would have been understood by a POSITA, when pooling is performed separately for each task, the operation is functionally within the head branch. Each distraction head in Zhengyang begins

with its own pooled feature vector, which is then processed by the attention and classification modules. Furthermore, because the attention models "have the same network structure as the main network model", they also include pooling layers. The SE-ResNet18 backbone includes pooling between its convolutional stages, and the same would apply to the attention sub-networks that replicate its structure. Therefore, even under a narrow interpretation of the head boundary, each attention model within the head includes its own convolution-and-pooling hierarchy. EX1007, ¶¶[0100]-[0103].

**8. Functional Role and Equivalence to Claim 2 Pooling Layer.**

468. The pooling operation in Zhengyang serves the same purpose as the pooling layer recited in Claim 2: it reduces spatial resolution and aggregates salient features before classification. By collapsing spatial dimensions and emphasizing key activations such as hand or object position, pooling facilitates the fully connected and soft-max stages that follow. The description "after the four feature maps are pooled, the feature vectors are input into four attribute-recognition modules" captures this functional role precisely. EX1007, ¶[0101]. A POSITA would understand that Zhengyang's pooling performs the same function, in the same manner, and achieves the same result as the claimed pooling layer, aggregating spatial features within each task branch for distraction classification. Whether described as part of the "interface" or "head" in prose, it remains within

the head's computational pathway. Even if an opposing interpretation asserted that this pooling occurs just before the head, it still meets the open-ended "includes" requirement of Claim 2, because the pooled features directly feed the head's classification computation.

469. *In summary,* Zhengyang discloses the pooling-layer portion of the limitation through three related teachings:

An explicit pooling operation produces task-specific feature vectors immediately before each head EX1007, ¶[0101];

The attention sub-networks share the SE-ResNet18 architecture, which uses convolution-and-pooling hierarchies *Id*., ¶¶[0100], [0103]; and

The pooling operation performs the same dimensional-reduction and feature-aggregation function as the claimed pooling layer. *Id*., ¶¶[0100], [0101].

470. A POSITA would therefore conclude that Zhengyang discloses the pooling-layer portion of Claim 2 of the '243 patent.

**9. Fully Connected-Layer Element.**

471. A POSITA would understand that Zhengyang expressly discloses fully connected layers within the distraction-classification heads, including the smoking-recognition and phone-use-recognition heads. These fully connected layers appear in both the attention-proposal sub-network and the subsequent classification stages.

472. Zhengyang explains that each distraction head "comprises a subject-recognition module and an attention module." EX1007, ¶[0031]. Within the attention module, the attention-proposal sub-network performs region localization and cropping and "mainly comprises two fully connected layers in series, which are used to output the centerpoint coordinates of the attention area and the edge lengths of the attention area to perform cropping and enlargement processing on the original input image." *Id.*, ¶¶[0034], [0101]. This statement identifies two fully connected layers and describes their function. A POSITA would understand that these layers belong to the same network branch used for distraction classification because their outputs control the image region subsequently analyzed by the attention models of the same head.

473. Zhengyang restates this architecture when describing end-to-end training: "The attention proposal sub-network comprises two fully connected layers in series … to perform cropping and enlargement processing." EX1007, ¶[0101]. This confirms that the fully connected layers are trained along with the rest of the network. A POSITA would further recognize that the attention models that follow the APN perform final binary classification and end with fully connected plus soft-max layers, as indicated by the computation of probabilities $p_2$ and $p_3$ through the soft-max function. *Id.*, ¶¶[0035], [0113]-[0116]. These fully

connected layers map the pooled features to decision values, performing the same function as the fully connected layers in the ’243 patent.

474. By 2020, it was standard practice in multi-task driver-monitoring networks for each head to terminate in one or more fully connected layers converting pooled features into output logits. Given Zhengyang’s explicit mention of “fully connected layers in series” and its description of categorical outputs, a POSITA would conclude that each distraction-classification head includes fully connected layers that implement the final decision stage.

475. Taken together, the evidence establishes:

The attention-proposal sub-network includes two fully connected layers; EX1007, ¶¶[0034], [0101];

The attention models end in FC + soft-max layers that produce binary classification outputs; *Id.*, ¶¶[0035], [0113]-[0116]; and

The training discussion confirms that these layers are part of the end-to-end architecture. *Id*., ¶[0101]

476. A POSITA would therefore conclude that Zhengyang discloses the fully connected-layer portion of Claim 2 of the ’243 patent.

**10. MTC**

477. Even if Zhengyang were read as not expressly listing all three layer types within each head (which I disagree with as laid out above), a POSITA would

find their inclusion obvious based on Zhengyang's own disclosure and because attention sub-networks that process 2-D images universally employ convolutional → pooling → fully-connected sequences for spatial analysis and classification. If it is argued that Zhengyang does not expressly enumerate each layer type within its distraction-classification heads, a POSITA would recognize that the described architecture incorporates convolutional, pooling, and fully connected operations consistent with the SE-ResNet18 framework it adopts. Zhengyang identifies SE-ResNet18 as the backbone network and states that the attention networks used in the smoking- and phone-use-recognition modules "have the same network structure as the main network model" EX1007, ¶¶[0101], [0103]. Because these attention networks process cropped 2-D image regions, a POSITA would have understood them to employ the standard ResNet-style sequence of convolutional layers for feature extraction, pooling (including stride-2 convolution and global-average-pooling) for spatial aggregation, and fully connected layers for binary classification. This inference follows directly from Zhengyang's reliance on an SE-ResNet-based multi-task design and its explicit cross-reference between the backbone and attention sub-networks.

478. **How:** Zhengyang defines its multi-task network using an SE-ResNet18 backbone composed of batch-normalization, Prelu activation, convolution, and residual blocks. EX1007, ¶[0100]. SE-ResNet18 is a conventional

convolutional network that extracts spatial features through successive convolutional and pooling operations and outputs classifications through fully connected layers. Zhengyang modifies this architecture to support multiple driver-state tasks, then directs that the attention models for smoking and phone-use "have the same network structure." A POSITA would understand this to mean that each head mirrors the SE-ResNet block pattern: using convolutional layers to analyze cropped driver images, pooling or stride-2 operations to reduce dimensionality, and fully connected layers with soft-max outputs to generate binary classifications. This conv-pool-fully connected composition was the natural and technically consistent implementation of Zhengyang's instruction to replicate the main network structure.

479. **Why:** By 2020, CNN-based driver-monitoring and attribute-recognition systems routinely used convolutional, pooling, and fully connected layers in exactly this progression. These layer types served distinct, complementary functions: convolution for localized feature extraction, pooling for spatial summarization and translational robustness, and fully connected layers for categorical decision-making. Corroborating evidence establishes the standard roles of these layer types in convolutional neural networks. Convolutional layers extract hierarchical spatial features from image data by applying learned filters across local receptive fields. EX1031; EX1025. Pooling layers summarize these spatial

activations, providing translation invariance and improving robustness and generalization. EX1040; EX1027. Fully connected layers aggregate the pooled features and generate classification outputs through learned linear transformations and softmax operations. EX1038; EX1030. Using this combination provided predictable accuracy, training stability, and computational efficiency in embedded or in-vehicle environments. Because Zhengyang already employs SE-ResNet18 as the backbone and emphasizes that all sub-networks share its architecture, a POSITA would have recognized that implementing the same convolution-pooling-fullyconnected composition in the attention heads was both technically compatible and practically advantageous. Implementing the same layer sequence in the attention heads (as already contemplated by Zhengyang) would have ensured architectural consistency with the SE-ResNet backbone, simplified gradient flow during joint training, and enabled unified weight sharing across all task branches. A POSITA would therefore have viewed this design as a predictable application of Zhengyang's own framework rather than an inventive modification.

480. **Reasonable Expectation of Success:** A POSITA would have expected success in using convolutional, pooling, and fully connected layers within Zhengyang's distraction-classification heads. Zhengyang defines a shared SE-ResNet18 backbone and states that the attention networks "have the same network structure as the main model", making them naturally compatible. Since the

attention heads process cropped driver images based on the same feature maps as the backbone, using the same convolutional-pooling-fullyconnected layout would maintain stable training and consistent performance, as seen in standard ResNet-based multi-task systems. EX1007, ¶[0103]. These layer types were standard in common CNN frameworks, so implementing them in Zhengyang's attention sub-networks would have been straightforward and predictably effective.

481. Accordingly, even if Zhengyang were found not to expressly identify each of these layer types within its distraction-classification heads, it would have been obvious to a POSITA to implement convolutional, pooling, and fully connected layers consistent with the SE-ResNet-based architecture to achieve the desired feature extraction, dimensionality reduction, and classification behavior. A POSITA would therefore conclude that Zhengyang either expressly or at least renders obvious the full limitation of Claim 2 of the '243 patent.

482. Accordingly, Zhengyang either expressly or at least renders obvious the limitation of Claim 2 of the '243 patent.

### D. Claim 3: The method of claim 1, wherein the object detection prediction head comprises a bounding box regression network and an object class prediction network, wherein the bounding box regression network outputs coordinates of a bounding box enclosing a detected object and the object class prediction network outputs a class corresponding to the detected object.

483. Zhengyang discloses claim 3, which recites an object-detection prediction head comprising a bounding-box regression network and an object-class prediction network. A POSITA would recognize that Zhengyang's multi-task driver-state recognition architecture already implements these same two functional components through its attention proposal sub-network ("APN") and attention networks. The APN performs spatial localization by predicting numerical coordinates defining attention regions, while the attention networks perform categorical classification of the cropped image regions. Together, these modules carry out the same localization-and-classification workflow characteristic of a bounding-box regression head paired with an object-class prediction head.

#### 1. Object Detection Prediction Head.

484. A POSITA would understand that Zhengyang discloses, within its multi-task driver-state recognition framework, an object detection prediction head that includes two distinct sub-networks: (1) a bounding box regression network that produces coordinates defining the spatial extent of a detected object, and (2) an object class prediction network that produces a categorical label corresponding

to that detected object. These components are embodied in Zhengyang's attention proposal sub-network (APN) and attention networks, respectively. Together, these modules cooperate to localize, crop, and classify distraction-related objects such as a cigarette or a mobile phone.

**2. Bounding Box Regression Network Outputs Object Coordinates.**

485. Zhengyang discloses that its APN "comprises two fully connected layers in series, which are used to output the center-point coordinates of the attention area and the edge lengths of the attention area to perform cropping and enlargement processing on the original input image." EX1007, ¶[0034]; *see also* ¶[0101]. A POSITA would understand these outputs (center coordinates and edge lengths) as the same four parameters (x-center, y-center, width, height) used in standard bounding-box regression networks such as Faster R-CNN or SSD. Although Zhengyang describes these outputs as attention areas rather than explicit detection boxes, a POSITA would recognize that they perform the same localization role: predicting the position and size of a region likely containing the object of interest (*e.g.*, a cigarette or mobile phone). The APN thus provides the bounding-box regression functionality within Zhengyang's multi-task framework. Zhengyang further states that these coordinates are used to guide cropping and enlargement, which are standard post-regression steps in object detection systems.

486. From the perspective of a POSITA, the attention proposal sub-network disclosed in Zhengyang performs the function of spatial localization rather than classification. Spatial localization refers to the process of determining *where* an object appears within an image by predicting numerical coordinates that define its position and size. In contrast, classification refers to determining *what* the object is by assigning it to a discrete category. The outputs identified in Zhengyang (*e.g.*, the "center-point coordinates of the attention area and the edge lengths of the attention area") are continuous numerical values that specify the exact position and dimensions of a rectangular region surrounding the detected object. These values correspond to the four parameters used in bounding-box regression networks: x-center, y-center, width, and height. Modern object detection architectures such as Faster R-CNN and SSD use the same type of coordinate regression to predict bounding box locations. The use of these continuous coordinate outputs is characteristic of regression-based localization rather than classification, which outputs categorical labels or probabilities.

487. In practical terms, the attention proposal sub-network in Zhengyang learns a mapping between image features and these spatial coordinates through a regression loss function that minimizes the difference between predicted and ground-truth box coordinates during training. This is the same computational mechanism employed by bounding-box regression networks in standard detection

pipelines. The resulting output provides precise positional data for each detected region in the image. A POSITA would therefore recognize that the attention proposal sub-network performs the same function as a bounding box regression network. Its purpose is to identify and localize the region of interest in the image that likely contains a distraction-related object, such as a cigarette or mobile phone, by predicting bounding-box coordinates. The attention network that follows then analyzes this localized region to determine the object's class.

488. Accordingly, the attention proposal sub-network disclosed in Zhengyang meets the structural and functional requirements of a "bounding box regression network" as recited in Claim 3. It provides the coordinate-based localization component of the object detection prediction head, which works in conjunction with the downstream classification component to perform full object detection within Zhengyang's multi-task driver-state recognition framework.

#### 3. Object Class Prediction Network Outputs Object Class Label.

489. After the bounding box regression stage, Zhengyang explains that the localized image region is processed by a downstream attention network that classifies the content of the cropped image. The reference states that "the enlarged image will be input into the first-level attention model to re-judge the attributes… and the APN generated by the first-level attention model will further crop and

enlarge the image and input it into the second-level attention model to re-judge the attributes." EX1007, ¶[0102]; *see also* ¶[0033]. A POSITA would understand that re-judging the attributes refers to evaluating what object or condition is present in the cropped image, which is a classification task. Each attention model takes the localized region identified by the APN and predicts which category that region belongs to, such as whether the image contains a cigarette or a phone.

490. Further, each of these attention models "has the same network structure as the main network model," which Zhengyang defines as SE-ResNet18, a convolutional network that produces a soft-max classification output. EX1007, ¶[0103]; *see also* ¶[0100]. SE-ResNet18 is a well-known convolutional neural network architecture designed for image classification. It performs hierarchical feature extraction through convolutional and pooling layers and concludes with fully connected layers and a soft-max activation that outputs class probabilities. Because Zhengyang states that its attention models share this structure, a POSITA would understand that they also perform categorical classification of image content.

491. This interpretation is supported by Zhengyang's explicit description of the binary outputs for its distraction-recognition tasks. The reference specifies that "the number of output categories … whether there is smoking, and whether a phone call is being made is two categories." EX1007, ¶[0028]. It also describes the

computation of "probability $p_2$ through the soft-max function … probability $p_3$ …" EX1007, ¶¶[0113]-[0116], confirming that the network produces per-class probabilities through soft-max activation. These are standard outputs of an object class prediction head in convolutional detection systems.

492. Therefore, a POSITA would understand that Zhengyang discloses an object detection prediction head composed of two cooperating sub-networks: (1) an attention proposal sub-network (APN) that performs bounding box regression to generate coordinates defining a region of interest, and (2) attention networks that function as object class prediction networks to classify the contents of that region. The APN outputs the center-point coordinates and edge lengths of each attention area, corresponding to standard bounding box parameters used in object detection systems. EX1007, ¶¶[0034], [0101]. The localized image regions are then input to the attention models, which share the same structure as the SE-ResNet18 backbone and produce categorical predictions such as "smoking" or "phone use" through soft-max classification. EX1007, ¶¶[0033], [0102]-[0103], [0028].

493. Collectively, these modules implement the conventional two-branch design of an object detection prediction head, in which one branch performs spatial localization and the other performs categorical classification. The APN determines *where* the object appears by predicting bounding box coordinates, and the attention

networks determine *what* the object is by assigning a class label to the detected region.

494. In summary, Zhengyang teaches an integrated detection framework in which the APN and attention networks together perform the respective localization and classification functions required by Claim 3. A POSITA would therefore conclude that Zhengyang satisfies the limitation of Claim 3 of the ’243 patent, which recites an “object detection prediction head compris[ing] a bounding box regression network and an object class prediction network.”

**E. Claim 4: The method of claim 3, wherein each of the bounding box regression network and the object class prediction network comprise deep neural networks having a plurality of hidden layers, each hidden layer in the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and a batch activation layer.**

495. Zhengyang discloses or renders obvious claim 4.

496. A POSITA would understand that Zhengyang describes both (1) the bounding-box regression network and (2) the object-class prediction network as deep neural networks composed of multiple hidden layers consistent with its SE-ResNet18 architecture. Each hidden layer includes convolutional, batch normalization (BN), and activation operations, components expressly identified in Zhengyang’s description of SE-ResNet18, and a POSITA would understand that these elements are incorporated into the feature-processing portions of the

attention-proposal and attention-classification sub-networks used for object detection and classification.

### 1. Zhengyang Defines the SE-ResNet18 Architecture and Its Layer Composition.

497. Zhengyang explains that its multi-task driver-state model is built upon SE-ResNet18, a deep residual convolutional neural network. The reference teaches that "the multi-task network structure used here is SE-ResNet18 … the 7*7 convolution kernel (stride=2) of the first convolution layer is replaced with a 3*3 convolution kernel (stride=1), the first maximum pooling layer is deleted, … [and t]he structure is BN layer, Prelu, convolution," and so on. EX1007, ¶[0100]. A POSITA would understand this as describing the residual-block pattern typical of SE-ResNet architectures with each block containing convolution, batch normalization, and activation operations repeated throughout the network.

498. A POSITA would understand that Zhengyang's reference to "BN layer, Prelu, convolution …" (*id.*) describes the standard residual-block structure used in SE-ResNet architectures, where each functional hidden layer is implemented as a sequence of convolution, batch normalization, and activation operations. In practice, these operations are tightly coupled and treated as a single hidden-layer unit in modern CNN design, meaning that each hidden layer in

Zhengyang's SE-ResNet18 performs the claimed convolution, normalization, and activation functions in combination.

**2. Convolutional Layer.**

499. The convolutional layer is the part of the network that looks at small areas of an image and searches for visual patterns. It uses filters, which act like digital lenses that slide over the image and detect features such as edges, lines, and textures. Each filter learns to recognize a different type of pattern as the model trains. Zhengyang's reference to a "3×3 convolution kernel (stride=1)" (EX1007, ¶[0100]) identifies this operation explicitly. The SE-ResNet18 model applies many of these convolutional layers in sequence, allowing the network to gradually build up an understanding of more complex shapes and objects, from simple edges to hands, faces, and phones. *See also* corroborating evidence EX1025.

**3. Batch Normalization Layer.**

500. Zhengyang also identifies the "BN layer" (EX1007, ¶[0100]), which a POSITA would understand as a batch normalization operation that stabilizes the distribution of feature activations across the mini-batch, improving convergence and preventing gradient explosion or vanishing. The batch normalization layer helps keep the network's calculations stable as it learns. After each convolutional layer detects patterns, the BN layer adjusts the results so that the values stay within a consistent range. This "normalizing" step prevents certain layers from producing

values that are too large or too small, which could slow down or destabilize training. By keeping the data balanced, the BN layer allows the network to learn faster and more reliably, even when it has many stacked layers. Zhengyang specifically mentions the "BN layer" in ¶[0100], showing that each convolutional block in the SE-ResNet18 model includes this normalization step to maintain stability and speed up learning. *See also* corroborating evidence EX1026.

**4. Activation Layer (Prelu).**

501. Zhengyang further specifies that its structure includes a "Prelu" activation layer (EX1007, ¶[0100]). The activation layer helps the network make flexible decisions about what patterns are important in an image. After the data is normalized, this layer adds a non-linear transformation, which allows the network to recognize complex relationships rather than just simple, straight-line patterns. In Zhengyang, the activation function used is called Prelu (Parametric Rectified Linear Unit). Prelu gives the model the ability to adjust how it reacts to negative input values, helping it learn more efficiently and adapt to different visual conditions. The phrase "BN layer, Prelu, convolution" in ¶[0100] shows that each stage of the SE-ResNet18 model includes this activation step after normalization and before the next convolution, forming part of the model's standard processing cycle. *See also* corroborating evidence EX1027; EX1028.

502. A POSITA would thus recognize that each "hidden layer" in Zhengyang's SE-ResNet18 network refers to such a residual block composed of convolution, batch normalization, and activation operations. These three steps together form the building unit of the network, repeated many times to create the deep layered architecture characteristic of SE-ResNet18.

503. Functionally, this composition is what allows Zhengyang's model to extract and refine hierarchical visual representations from driver images: (1) The convolutional layer detects low-level spatial features; (2) The batch normalization layer ensures consistent scaling and improves training stability; and (3) The activation layer (Prelu) introduces nonlinearity so that deeper layers can represent complex driver postures, hand positions, and object shapes (e.g., a phone or cigarette).

504. Thus, the reference to "BN layer, Prelu, convolution" in Zhengyang (EX1007, ¶[0100]) identifies exactly the type of hidden layer structure recited in Claim 4: a sequence of convolutional, normalization, and activation operations repeated across multiple hidden layers of the network. Because Zhengyang further states that its attention networks "have the same network structure as the main network model" (*id.*, ¶[0103]), a POSITA would conclude that both the bounding-box regression and classification heads likewise include hidden layers having the same convolution-BN-activation pattern.

**5. Bounding-Box Regression Network (Attention Proposal Sub-Network).**

505. As discussed in the Claim 3 analysis, Zhengyang's attention-proposal sub-network (APN) outputs bounding-box coordinates for regions of interest. While Zhengyang states that the APN "mainly comprises two fully connected layers in series" (EX1007, ¶[0034]), a POSITA would understand this refers only to the final regression stage that produces coordinate values. Before that stage, the APN would process backbone feature maps using convolutional blocks consistent with SE-ResNet18 to extract spatial features. This design follows the same pattern used in region-proposal networks (RPNs) from Faster R-CNN and Mask R-CNN, which apply convolutional layers before fully connected regression heads to maintain spatial accuracy and stable training.

506. Zhengyang explains that the APN is part of an "attention module" that performs "attribute recognition and optimization of the attribute recognition results of the main network" (*id.*, ¶[0033]). A POSITA would understand this to mean that the APN operates on feature maps from the SE-ResNet18 backbone. A POSITA would understand that fully connected layers alone cannot effectively process the spatial feature maps from the SE-ResNet18 backbone. Passing these maps directly into fully connected layers would collapse their two-dimensional structure, removing spatial relationships between features and preventing accurate

localization of regions of interest. It would also add unnecessary parameters and reduce computational efficiency. Following standard detection architectures like Faster R-CNN and Mask R-CNN, a POSITA would therefore include convolution-batch-normalization-activation blocks before the final fully connected regression layers to preserve spatial information and enable accurate, stable bounding-box prediction.

507. In modern CNN architectures, these hidden layers are constructed using a standard sequence of convolution, batch normalization, and activation (Prelu or ReLU) to maintain stable training and consistent feature scaling. A POSITA would therefore implement the APN's intermediate layers in this same manner, consistent with the SE-ResNet18 pattern described in Zhengyang (*id.*, ¶[0100]). Even though the reference highlights only the two final fully connected layers, a POSITA would understand that those operate on the output of preceding convolutional blocks, exactly as in standard detection architectures.

**6. Object-Class Prediction Network (Attention Networks).**

508. For the object-class prediction network, Zhengyang discloses one or more attention models that classify cropped image regions. Zhengyang states that "the first-level attention neural network model and the second-level attention neural network model here have the same network structure as the main network model and will output attribute recognition results as described above" (*id.*,

¶[0103]). A POSITA would understand that "same network structure" refers to the reuse of the SE-ResNet18 residual-block pattern within each attention model, i.e., multiple stacked convolution-BN-activation blocks forming a deep CNN for classification of the localized image regions.

509. Because the main network is SE-ResNet18, defined by the "BN layer, Prelu, convolution" block pattern (*id.*, ¶[0100]), each attention network would also include multiple hidden layers using the same sequence. A POSITA would recognize that reusing this structure allows shared feature representation, stable gradient flow, and consistent training behavior, which are well-established advantages in SE-ResNet models. See corroborating evidence EX1029. These blocks enable each attention network to extract hierarchical features and produce soft-max classification outputs, consistent with standard multi-task CNN design.

**7. POSITA Understanding of Shared Architecture.**

510. Zhengyang emphasizes that the attention networks and main network "have the same network structure" (EX1007, ¶[0103]) and that the APN operates on feature maps produced by the backbone and is trained jointly with the classification heads (*id.*, ¶¶[0033], [0101]). A POSITA would understand this to mean that all sub-networks in Zhengyang use the same deep-layer template composed of repeated convolution-BN-activation blocks, ensuring compatibility and stable joint training. This interpretation aligns with established practice in

multi-task ResNet architectures such as Faster R-CNN, Mask R-CNN, and other SE-ResNet derivatives, which maintain uniform layer composition across detection and classification branches to ensure consistent feature scaling and optimization behavior.

511. In summary, based on Zhengyang's explicit architectural description and its stated use of SE-ResNet18 as the shared backbone, a POSITA would conclude that both the bounding-box regression network (attention-proposal sub-network) and the object-class prediction network (attention networks) are implemented as deep neural networks composed of multiple hidden layers, each following the SE-ResNet residual-block pattern of convolution, batch normalization, and activation.

512. However, to the extent it is determined that Zhengyang does not expressly enumerate each component, these elements would have been obvious to a POSITA implementing Zhengyang's SE-ResNet-based design to achieve standard feature extraction and classification functionality, as described next.

**8. MTC**

513. To the extent Zhengyang is viewed as not explicitly reciting that each hidden layer includes convolution, BN, and activation layers, a POSITA would have found it obvious to implement these components consistent with the SE-ResNet18 design.

514. **How:** A POSITA would have found it obvious to implement each hidden layer in Zhengyang's bounding-box regression and object-classification networks using the same convolution-batch-normalization-activation sequence defined in its SE-ResNet18 backbone. Zhengyang describes this structure as "BN layer, Prelu, convolution …" (EX1007, ¶[0100]) and notes that the attention networks "have the same network structure as the main network model" (*id.*, ¶[0103]). A POSITA would understand this as the standard residual-block pattern used in SE-ResNet, where convolution, normalization, and activation are repeated across layers. Keeping this same block structure in the detection and classification heads would ensure architectural compatibility and allow joint training with the shared backbone. Although Zhengyang states that the APN "mainly comprises two fully connected layers" (*id.*, ¶[0034]), a POSITA would view those as final output layers, with earlier layers necessarily including convolutional blocks for spatial feature extraction, just as in region-proposal networks from Faster R-CNN and Mask R-CNN, which use convolutional stacks followed by fully connected regression heads to maintain spatial and gradient consistency.

515. **Why:** Using the same convolution-BN-activation block structure across all sub-networks would have been a straightforward and well-understood choice. It keeps data scaling consistent, stabilizes gradient flow, and ensures feature representations match between the backbone and task-specific heads. These

benefits were well known from SE-ResNet research (EX1029) and detection models like Faster R-CNN and Mask R-CNN (EX1030; EX1008). By the priority date, it was standard practice to use identical block designs in detection and classification branches to simplify training and improve convergence, as confirmed by Zhengyang's own teachings. A POSITA would therefore view the claimed layer sequence: convolution, batch normalization, and activation, as the conventional way to implement Zhengyang's SE-ResNet-based attention and proposal networks.

516. **Reasonable Expectation of Success:** A POSITA would have expected this implementation to succeed with little or no experimentation. SE-ResNet-based detection and classification models had already demonstrated stable training and high accuracy using the same convolution-BN-activation blocks in every residual unit. Major deep-learning frameworks provided these layers as standard modules, allowing straightforward replication of the backbone's structure in all sub-networks. Because Zhengyang already specifies that its attention and proposal networks share the same architecture as the backbone, extending that structure to include convolution, batch normalization, and activation in each hidden layer would have been a routine engineering step yielding predictable results: maintaining stability, improving convergence, and producing the same performance characteristics observed in other ResNet-based detection systems. *See* corroborating evidence EX1030 *generally*; EX1029; EX1019; EX1008.

517. Accordingly, even if it is determined that Zhengyang does not expressly enumerate every layer type within the bounding-box regression and classification networks, a POSITA would have found it obvious to implement each hidden layer using convolution, batch normalization, and activation layers consistent with the SE-ResNet18 architecture. Zhengyang therefore either expressly discloses or, at minimum, renders obvious the limitation of Claim 4 of the ’243 patent.

**F. Claim 5: The method of claim 1, wherein the pose estimation prediction head includes a plurality of hidden layers and an output layer, each of the plurality of hidden layers comprising a convolutional layer, a batch normalization layer, and an activation layer, and the output layer comprising a convolutional layer.**

518. Zhengyang in view of He renders obvious claim 5.

519. As discussed above for claim 1, a POSITA would understand that, to the extent Zhengyang does not *expressly* disclose a pose-estimation prediction head, it would have been obvious to incorporate one, specifically, a convolutional pose-estimation head of the type explicitly described in He 2017 (EX1008). Zhengyang already provides the multi-task backbone and head framework; He provides the detailed architecture of a pose-estimation head that fits seamlessly into such a framework. See element 1[C].

**1. Zhengyang's Structural Correspondence and Complementarity.**

520. Zhengyang's SE-ResNet18 backbone produces spatial feature maps that feed several task-specific heads for mask, eyeglass, smoking, and phone-use detection. Zhengyang states that "the output results of the attribute recognition of each task are whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]. This structure mirrors a multi-head CNN design, where shared features are routed to specialized branches. Zhengyang further states that these attention networks "have the same network structure as the main network model." EX1007, ¶[0103]. A POSITA would understand this to mean that all heads, including any added head, such as a pose-estimation head, would follow the same internal layer sequence defined for the backbone ("BN layer, Prelu, convolution …"). EX1007, ¶[0100].

**2. He**

521. He describes a pose-estimation (keypoint) head as an additional convolutional sub-network attached to a shared backbone:

522. "Our framework can easily be extended to human pose estimation. We model a keypoint's location as a one-hot mask, and adopt Mask R-CNN to predict K masks, one for each of K keypoint types." EX1008, 2986.

523. "The keypoint head consists of a stack of eight 3×3 512-d conv layers, followed by a deconv layer and 2× bilinear upscaling, producing an output resolution of 56×56." EX1008, 2986.

524. A POSITA would recognize this as a deep convolutional neural network branch with multiple hidden layers and a final convolutional output layer. The hidden layers perform feature extraction, each composed of a convolutional layer followed by batch normalization and an activation layer, as was standard in ResNet-derived models during the relevant time. The final output convolutional layer produces the per-keypoint heatmaps used for pose estimation.

525. Claim 5 requires that the pose-estimation prediction head include (1) a plurality of hidden layers, each with convolutional, batch-normalization, and activation layers, and (2) an output layer comprising a convolutional layer. He discloses this configuration directly. A POSITA would understand this description as explicitly teaching the following layer structure:

**(a) Hidden Layers (Eight 3×3 Convolutional Layers).**

526. He. clearly states that the keypoint head "consists of a stack of eight 3×3 512-d conv layers." EX1007, 2986. These eight layers are the hidden layers of the pose-estimation head. Each layer processes the shared feature maps to extract detailed spatial information about keypoint locations. A POSITA would understand that, in models like Mask R-CNN, every convolutional layer is

followed by a batch normalization layer and an activation function such as ReLU. This combination, convolution, batch normalization, and activation, is the standard structure of a hidden layer in a ResNet-based network and matches the configuration described in Claim 5. Further, even if it were argued that He does not explicitly disclose these hidden layers, it would have been obvious when combined with Zhengyang. Because Zhengyang states that all heads share the same network structure as its SE-ResNet18 backbone, when combining Zhengyang with He, a POSITA would have implemented He's pose-estimation head using the same convolution-BN-activation sequence defined in Zhengyang (EX1008, ¶[0100]) and applied consistently across all other heads. This would result in a pose-estimation head having the exact configuration recited in Claim 5.

**(b) Output Layer (Final Convolutional Layer).**

527. The final step in He's keypoint head is another convolutional layer that produces the model's output. This layer generates the "K masks," where each mask represents a heatmap showing the predicted position of one keypoint type, such as the head, shoulders, or elbows. A POSITA would recognize this as the output layer described in Claim 5, which specifies that the output layer "comprises a convolutional layer."

**(c) Supporting Operations (Deconvolution and Upsampling).**

528. After the hidden convolutional layers, He adds a deconvolution layer and bilinear upsampling to make the output heatmaps larger and easier to interpret. These steps help refine the pose predictions but do not change the basic structure of the network, which still follows the same sequence of convolution, batch normalization, and activation in the hidden layers, ending with a final convolutional layer that produces the keypoint masks.

529. In summary, He explicitly identifies eight convolutional layers (the hidden layers) and a final output convolutional layer that generates the pose predictions. A POSITA would understand that each convolutional layer is followed by batch normalization and an activation function, as this was the standard and explicitly referenced design in ResNet and Mask R-CNN architectures. When considered together with Zhengyang, which expressly defines this same hidden-layer structure ("BN layer, Prelu, convolution …" (EX1007, ¶[0100])), it would have been clear and obvious to implement He's pose-estimation head using that same configuration.

**3. MTC**

530. **How:** A POSITA would have added He's pose-estimation head as another branch connected to the shared feature maps produced by Zhengyang's

SE-ResNet18 backbone. The input to this new branch would be the same feature data already used by Zhengyang's attention heads, and the output would be heatmaps showing the locations of keypoints such as the driver's head, shoulders, and arms. Each hidden layer would follow the same sequence of convolution, batch normalization, and activation described in both He and Zhengyang, and the final layer would be a convolutional layer that generates the keypoint heatmaps. This addition would fit into Zhengyang's existing framework without modifying the backbone or other heads, requiring only an additional output and corresponding training objective to guide the pose-estimation predictions.

531. **Why:** A POSITA would have been motivated to add He's pose-estimation head to Zhengyang's driver-monitoring network to enhance its ability to determine the driver's posture and detect distraction. Zhengyang's stated goal is to "comprehensively reflect the driver's driving state, thereby improving the accuracy of driver state recognition." EX1007, ¶[0046]. Integrating a pose-estimation head of the type described in He would directly serve this goal by providing detailed spatial information about the driver's body position and movement. The added head would analyze keypoints such as the driver's head, shoulders, and arms, data essential for detecting conditions like looking away from the road, reaching for an object, or removing hands from the wheel. The keypoint head would supply fine-grained spatial data complementary to Zhengyang's categorical recognition heads

(e.g., smoking or phone use). Because the hidden layer convolution-BN-activation pattern used in He and Zhengyang was routine, and Zhengyang specifies that all its task heads share the same SE-ResNet18 structure (EX1008, ¶[0100] ("BN layer, Prelu, convolution"), ¶[0103] ("same network structure as the main network model")), a POSITA would have naturally implemented He's pose-estimation head using that same convolution-BN-activation pattern. This would maintain architectural consistency across all prediction heads, as contemplated by Zhengyang, thereby improving accuracy and training stability while extending the system's capabilities to include pose analysis.

532. He further shows that adding a pose-estimation head increases the model's capability without interfering with other branches of the network: "We have a unified model that can simultaneously predict boxes, segments, and keypoints while running at 5 fps." EX1008, 2987. A POSITA would understand that this demonstrates the compatibility of a pose-estimation head with existing detection and classification heads. Incorporating such a head into Zhengyang's multi-task framework would therefore be a straightforward, predictable enhancement that strengthens the system's ability to identify driver posture and distraction. Because Zhengyang specifies that all heads share the same SE-ResNet18 structure, a POSITA would have used the same convolution-BN-

activation pattern for the new head, ensuring consistent training behavior and unified network performance.

533. **Reasonable Expectation of Success:** A POSITA would have expected this combination to succeed. Both Zhengyang and He use convolutional backbones that output dense spatial feature maps, multiple task-specific heads trained under separate losses, and identical layer types (convolution, batch normalization, activation). Implementing He's keypoint head within Zhengyang's existing framework would simply involve attaching a new branch and training it with labeled keypoint data. He confirms that the keypoint branch can be added without disrupting other tasks as the framework "can be readily extended to more complex tasks." EX1008, 2981. The combination would therefore operate reliably and predictably using established multi-task CNN techniques.

534. Accordingly, a POSITA would conclude that the combination of Zhengyang and He renders obvious, a pose-estimation prediction head that meets the requirements of Claim 5. He provides the explicit convolutional, batch-normalized, and activated hidden layers (or at least would have been obvious in view of Zhengyang) and the final convolutional output layer producing pose heatmaps, while Zhengyang provides the shared-backbone, multi-head context and motivation. The integration would have been straightforward, well-motivated, and predictably successful.

### G. Claim 6: The method of claim 1, further comprising inputting image data into the distraction classification model and receiving a distraction classification tag from the distraction classification model based on the image data.

535. Zhengyang alone or in combination with He discloses or renders obvious claim 6.

#### 1. Zhengyang

536. A POSITA would understand that Zhengyang discloses or a minimum strongly suggests this process in full. Zhengyang describes a workflow in which image data of the driver is provided as input to a trained multi-task neural network that analyzes the data and outputs a classification result, a tag, indicating whether the driver is distracted by smoking or phone use.

##### (a) Inputting Image Data into the Distraction Classification Model.

537. Zhengyang explicitly teaches that image data captured from an in-vehicle driver-facing camera is preprocessed, labeled with driver-state attributes, and input into a multi-task neural network model for both training and inference. As described in ¶¶[0020-0023], Zhengyang's training process begins by "inputting the preprocessed image data into the multi-task main network to obtain a feature map …," after which "the output results of the attribute recognition of each task are … whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made," and those outputs are compared

"to the labeling results … through the corresponding loss function" to optimize network parameters.

538. Prior to training, the system "label[s] the sample image data with multi-task attributes" such as the presence or absence of a mask, eyeglasses, smoking, and phone use. EX1007, ¶¶[0025]. Thus, the image data input into the model during training constitutes labeled, preprocessed driver images that pair the visual data with attribute annotations for supervised learning.

539. During deployment, Zhengyang applies the same model, now pre-trained, to incoming driver images captured in real time, inputting those image data of the driver into the pre-trained multi-task network model to predict the same attributes. EX1007, ¶¶[0031], [0046], [0061]-[0062], [0139], [0145]. A POSITA would therefore understand that the term "inputting image data" in Claim 6 is satisfied by Zhengyang's (i) the supervised training stage, where labeled, preprocessed driver images are input to optimize model parameters, and (ii) the inference stage, where newly captured driver images are input to generate distraction-classification outputs. In both contexts, Zhengyang's disclosure satisfies the claim's requirement of inputting labeled image data into the distraction-classification model.

**(b) Receiving a Distraction Classification Tag from the Model.**

540. Zhengyang further describes that the model outputs classification results for each of the monitored driver-state tasks: "The output of the entire network is a feature map of each task and an attribute recognition result of each task … whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]. A POSITA would recognize that these "attribute recognition result[s]" are categorical classification outputs generated by the model. For the smoking and phone-use modules, Zhengyang specifies that the model produces binary outputs corresponding to two classes: "The number of output categories … whether there is smoking, and whether a phone call is being made is two categories." EX1007, ¶[0028]; *see also* EX1007, ¶¶[0098]-[0099]. Zhengyang explains that these classification outputs are computed using a soft-max function at the final stage of the forward pass, producing probabilities that the driver is or is not engaged in the activity. EX1007, ¶¶[0113]-[0116]. A POSITA would interpret these binary classification results as the "distraction classification tag" recited in Claim 6, corresponding to predictions such as "smoking detected" or "phone use detected."

541. A POSITA would have understood that Zhengyang's "attribute recognition result" represents a classification tag because (1) it is a binary

categorical prediction identifying whether the driver is or is not performing a distracting activity, and (2) in CNN practice, these results were presented as classification labels or tags for downstream decision modules. In standard CNN implementations, the class with the highest probability is selected to yield a discrete prediction (e.g., "smoking" or "not smoking"), which a POSITA would interpret as a classification tag. Zhengyang's binary, per-task categorical outputs function as tags indicating the driver's detected distraction state. A POSITA would further understand that these outputs are received by the vehicle's driver-state monitoring controller for subsequent alert generation or data logging, thereby satisfying the claimed step of "receiving a distraction-classification tag from the model.

542. Thus, Zhengyang describes an integrated, end-to-end inference pipeline in which the driver's image data is received, processed through the SE-ResNet18 backbone, and analyzed by the task-specific heads to produce classification results. The system is designed to "comprehensively reflect the driver's driving state and improve the accuracy of driver state recognition." EX1007, ¶[0046].

543. A POSITA would understand that this represents a real-time classification process: inputting driver image data into the model, executing a forward pass through the backbone and heads, and receiving categorical outputs

that identify distraction-related behaviors. This process is directly analogous to the method recited in Claim 6. The "distraction classification model" in Claim 6 corresponds to Zhengyang's trained multi-task SE-ResNet18 network, and the "distraction classification tag" corresponds to the binary output identifying whether the driver is engaged in smoking or phone use.

544. To the extent Zhengyang is viewed as not expressly disclosing the step of receiving a distraction classification tag, a POSITA would have found it obvious to combine Zhengyang's multi-task driver-state recognition model with the explicit class-label output taught by He. Both references describe deep convolutional networks that process image data through a shared backbone and produce per-task classification outputs. The combination would represent a routine and predictable adaptation within established CNN design practice. Indeed, it would have been a matter of common sense for a POSITA to adapt Zhengyang's output from attribute-level predictions (e.g., gaze direction, hand position, or posture) to categorical outputs such as a 'distracted' or 'not distracted' tag. Given that both types of outputs are derived from the same shared feature representations in a multi-task CNN, substituting or augmenting the attribute label with an overall classification tag represents a routine design choice consistent with standard deep-learning practice and consistent with He's teachings.

**2. He**

545. He describes how Mask R-CNN produces interpretable class-label outputs from its classification branch. In extending Faster R-CNN, He explains that the detector "has two outputs for each candidate object, a class label and a bounding-box offset; to this we add a third branch that outputs the object mask." EX1008, 2982. Later, He clarifies that the classification branch predicts the object's category and that this predicted class index (k) determines which mask is used: "we only use the k-th mask, where k is the predicted class by the classification branch." EX1008, 2983. A POSITA would recognize that these passages together describe the conventional CNN classification process: feature maps produced by convolutional layers are passed through a fully-connected head, followed by a soft-max that yields per-class probabilities, and the highest-probability class is selected as the label output. Although He applies this mechanism to detected object categories (e.g., "person," "car"), the output itself is a discrete categorical label produced from image features by a soft-max decision function. Thus, in functional terms, He's class-label output corresponds to the claimed distraction-classification tag: a categorical indicator derived from image data that identifies which of multiple possible classes applies.

546. A POSITA would further understand that He's class-label branch was designed to be modular and easily adapted to new classification tasks. He explicitly

notes that the classification head operates in parallel with other branches using the same convolution-BN-activation building blocks, making it straightforward to substitute task-specific categories. Because Zhengyang's multi-task SE-ResNet18 network employs the same primitives, convolutional layers feeding into a soft-max classifier, a POSITA would find it obvious to apply He's explicit class-label output mechanism within Zhengyang's driver-state recognition framework. The adapted system would process driver images through the SE-ResNet18 backbone and its task-specific heads, each producing a soft-max-based categorical output (e.g., "smoking," "phone use," or "no distraction") analogous to He's per-object class label, thereby satisfying the claimed limitation of receiving a discrete distraction-classification tag.

**3. MTC**

547. **How:** Zhengyang already describes a multi-task driver-state recognition system that captures driver images, inputs them into a shared SE-ResNet18 backbone, and produces "attribute recognition result[s]" for multiple tasks, including mask-wearing, smoking, and phone use. EX1007, ¶[0021]. The backbone generates shared feature maps that feed into task-specific attention heads. He discloses a nearly identical backbone-to-head architecture for object detection and segmentation: accepting image data, generating shared convolutional features, and routing them to parallel heads, including a classification branch that

outputs a class label for each detected instance ("two outputs for each candidate object, a class label and a bounding-box offset" (EX1008, 2982)). A POSITA would recognize that both systems follow the same modular CNN paradigm: shared feature extraction followed by specialized prediction branches.

548. Zhengyang explicitly states that each of its attention models has "the same network structure as the main network model" (EX1007, ¶[0103]), defined as SE-ResNet18 (EX1007, ¶[0100]). Although this "same structure" language refers primarily to layer composition (convolution, batch normalization, activation), a POSITA would understand that the same soft-max-based classification behavior would also be replicated to maintain consistent output formatting and training flow across tasks. Because SE-ResNet18 itself is a classification model, its downstream attention heads are naturally classification-oriented. Incorporating He's explicit class-label output branch into Zhengyang's attention heads would therefore formalize what Zhengyang already implies: the generation of categorical outputs for each driver-state task (e.g., "smoking," "phone use," or "no distraction"). The categorical output would then be received by Zhengyang's downstream driver-monitoring or alerting module, performing the claimed "receiving a distraction classification tag" step. Implementing He's classification branch within Zhengyang's framework would not require architectural redesign. Zhengyang's heads already use the SE-ResNet18 pattern, convolution, batch normalization, and

activation followed by fully connected output layers, and He's branch uses the same structure. Adding a final soft-max classification layer, as in He, would be a predictable and design-consistent extension, ensuring that each head produces a clear categorical label in a format compatible with standard multi-task CNN outputs.

549. **Why**: Zhengyang's stated objective is to "comprehensively reflect the driver's driving state" and "improv[e] the accuracy of driver state recognition." EX1007, ¶[0046]. Producing explicit categorical tags for each task directly supports that goal by making the model's predictions interpretable and actionable for real-time driver monitoring. He demonstrates that adding a class-prediction branch to a multi-head CNN is a standard, modular practice that allows other network heads (e.g., mask or keypoint branches) to share features while maintaining distinct soft-max classification outputs, EX1008, 2982, 2987.

550. By the time the '243 patent was filed, systems that used convolutional neural networks (CNNs) to monitor drivers were already producing simple, easy-to-read results, like tags saying "distracted" or "hands off wheel." *See* corroborating evidence EX1032; EX1033. A POSITA would therefore have been motivated to include He's class-label output feature in Zhengyang's model. Doing so would make the system's predictions clearer, ensure all tasks use the same type of output, and make it easier to connect with other car systems that rely on those

simple tags. Since Zhengyang's task heads already follow the SE-ResNet18 classification setup, adding that output step would be a straightforward design improvement, not a major change or new concept.

551. **Reasonable Expectation of Success.** A POSITA would have been confident that combining He and Zhengyang would work smoothly. Both systems use the same kind of deep-learning setup, a shared convolutional backbone that sends feature maps to different task heads built from the same basic layers: convolution, batch normalization, activation, and a soft-max output. He also shows that its classification branch can run alongside other heads for segmentation and pose estimation without causing problems or reducing accuracy ("[our] model … can simultaneously predict boxes, segments, and keypoints" (EX1008, 2987)). This would reassure a skilled engineer that adding a similar classification branch to Zhengyang's model and distraction classification head would be stable and reliable, with no risk of disrupting how the network trains or makes predictions.

552. Finally, the classification process described in He is not limited to any one task. While He's network outputs object categories, the same setup could easily be used to label driver behaviors in Zhengyang's distraction-detection system. Adding He's class-label branch to Zhengyang's model would be a straightforward and expected step, using the same well-known CNN design methods. The result would be exactly what both systems are designed to achieve:

clear, categorical outputs such as “smoking,” “on phone,” or “no distraction,” produced from driver images and passed to the vehicle’s monitoring system for interpretation. Stated differently, He’s classification process is not limited to object categories; it is a task-agnostic framework for mapping feature maps to categorical outputs through soft-max inference. Adapting it to Zhengyang’s behavioral categories (e.g., “smoking,” “phone use,” “no distraction”) would require only substituting the label set, not altering the architecture.

553. Accordingly, a POSITA would conclude that the combination of Zhengyang and He renders Claim 6 of the ’243 patent obvious. Zhengyang provides the full framework of a driver-monitoring system that captures image data, processes it through a shared SE-ResNet18 backbone, and produces per-task attribute-recognition results, while He contributes the explicit class-label output branch and soft-max classification implementation used in analogous multi-head CNNs. Because Zhengyang already defines each head as sharing the same SE-ResNet18 classification structure, adopting He’s explicit class-label output would have been a routine, design-consistent step that yields predictable, interpretable results. The resulting system would input driver image data, execute a forward pass through the network, and output categorical distraction tags, such as “smoking,” “phone use,” or “no distraction”, that are received by the driver-monitoring module, thereby meeting the functional and structural requirements of Claim 6.

### H. Claim 9: The method of claim 1, wherein the distraction classification prediction head classifies an input vector with a label, the label comprising one or more of an undistracted, mobile usage, or other distracted label.

554. Zhengyang in view of He discloses or renders obvious claim 9.

#### 1. Zhengyang

555. A POSITA would understand that Zhengyang expressly discloses the underlying functionality required by this claim. Zhengyang describes a multi-task driver-state-recognition network trained to classify images of a driver into discrete behavioral categories, including "whether there is smoking" and "whether a phone call is being made." EX1007, ¶[0021]. Both are identified as driver-distraction behaviors that the invention seeks to recognize to "improve[e] the accuracy of driver state recognition." EX1007, ¶[0005]. The complementary outcomes ("no phone call," "no smoking") represent a non-distracted or "undistracted" condition. Thus, Zhengyang's attribute-recognition tasks naturally map to the same categorical structure recited in Claim 9. Further, a POSITA would understand that each input image is internally represented as a feature vector after preprocessing and convolution, so Zhengyang's image input data is analogous to the claimed 'input vector' in CNN terminology.

##### (a) Express Disclosure of Classification Labels.

556. Zhengyang defines four attribute-recognition tasks, mask wearing, eyeglasses wearing, smoking, and phone-call recognition, and states that the

network outputs an "attribute-recognition result" for each task describing the driver's current state: "The output of the entire network is a feature map of each task and an attribute recognition result of each task … whether a mask is worn, whether eyeglasses are worn, whether there is smoking, and whether a phone call is being made." EX1007, ¶[0021]. While Zhengyang labels only the distracted conditions, a POSITA would understand that standard CNN classification frameworks includes the complementary non-event label (e.g., "no phone," "no smoke") as the default or baseline category representing the undistracted state.

557. For the distraction-related tasks, Zhengyang explains that each produces a binary output: "The number of output categories of the attribute recognition tasks of whether a mask is worn, whether there is smoking, and whether a phone call is being made is two categories." EX1007, ¶[0028]. A POSITA would understand each head to produce a categorical decision ("yes/no") for its task, consistent with standard CNN classification behavior. In contemporary CNN terminology, a POSITA would understand "attribute-recognition result" to denote the categorical label obtained by applying a soft-max or arg-max operation to the output probabilities of each prediction head. This is the standard way a classification head "labels" an input vector with one of its predefined categories, such as "phone use detected" or "no phone use." Accordingly, Zhengyang's binary

outputs constitute the generation of explicit class labels within the meaning of the claim, as would have be understood by a POSITA.

558. Further, Zhengyang specifies that the multi-task network includes "smoking recognition" and "phone call recognition" heads. EX1007, ¶[0008]. From the context of improving driving safety by "improving the accuracy of driver state recognition" (EX1007, ¶[0005]), a POSITA would interpret the phone-call-recognition head as corresponding to *mobile-usage* distraction and the smoking-recognition head as corresponding to *other distracted* behavior. The negative outcomes of those tasks (e.g., no phone call and no smoking) represent the *undistracted* condition. This direct correspondence between Zhengyang's binary outputs and the claimed labels is summarized as follows:

| **Claimed Label** | **Zhengyang Equivalent** | **Support** |
|---|---|---|
| Undistracted | "No smoking" / "No phone call" | ¶[0028] |
| Mobile Usage | "Phone-call recognition" | ¶¶[0008], [0020] |
| Other Distracted | "Smoking recognition" | ¶¶[0008], [0020] |

559. Further, Zhengyang explains that its recognition module processes each captured image to obtain the "attribute recognition result of each task." EX1007, ¶[0021]. A POSITA would understand this as the standard CNN workflow in which an input feature vector is passed through fully connected and

soft-max layers that output the most probable label. Thus, Zhengyang's operation of mapping an input image to categorical results constitutes the claimed step of "classifying an input vector with a label." This is consistent with the structure of multi-head CNNs used in the field, such as SE-ResNet-based models performing per-task classification. To the extent it was determined that Zhengyang does not explicitly output a single consolidated "undistracted / mobile-usage / other-distracted" label, a POSITA would have found it obvious to include one, specifically in view of He's disclosure.

**2. He**

560. He's Mask R-CNN architecture supplies the explicit disclosure regarding how convolutional features are converted into categorical labels. He describes a network that outputs "two outputs for each candidate object, a class label and a bounding-box offset" and further explains that "we only use the k-th mask, where k is the predicted class by the classification branch." EX1008, 2982-2983. These passages illustrate the standard deep-learning implementation in which convolutional features are passed through fully-connected and soft-max layers to produce a categorical class label controlling downstream behavior. A POSITA would recognize this as the same labeling logic implied in Zhengyang's "attribute-recognition result." Incorporating He's explicit class-label branch into Zhengyang merely formalizes how the smoking-and phone-use-recognition heads

output categorical results. It does not alter Zhengyang's structure, only clarifies that each task head outputs a class label identifying the driver's state.

**3. MTC**

561. A POSITA would have found it straightforward to combine He's explicit class-label output design with Zhengyang's driver-monitoring network. Zhengyang already describes the core components: driver images as inputs, an SE-ResNet18 feature-extraction backbone, and task-specific prediction heads for detecting smoking and phone use. He adds the standard classification head structure: convolutional layers followed by fully connected and soft-max layers that converts prediction scores into clear categorical labels. Using He's output method in Zhengyang would simply formalize the binary predictions as labeled results. In this combined system, the smoking and phone-call heads would output labels such as "mobile usage" or "other distracted," and the absence of those behaviors would represent "undistracted." The overall architecture would follow the same pattern He describes: shared feature extraction, specialized heads, and a classification branch that outputs labeled decisions.

562. **How:** A POSITA would implement He's class-label branch in Zhengyang by adding a final soft-max and arg-max stage to each distraction head to output a categorical label rather than a raw probability. Each distraction head in Zhengyang already gives binary results, so merging them into one three-class

output would be a simple software step. A developer could do this by mapping the existing outputs into a single tag without changing how the network is built or trained. This integration would not alter Zhengyang's architecture or training objective, as each head already performs binary classification. The modification merely formalizes the output as a labeled class, following He's standard practice for classification heads in multi-task CNNs.

563. **Why:** The motivation to provide a unified distraction label arises directly from Zhengyang's stated objective of "improving the accuracy of driver state recognition." EX1007, ¶[0005]. To achieve that goal, a driver-monitoring system must distinguish between normal and distracted conditions to trigger feedback and the system must produce interpretable categorical outputs suitable for triggering alerts or reports. Unifying the binary heads into one categorical output simplifies the downstream decision logic and reduces the number of I/O channels and alert conditions that must be handled by the vehicle HMI controller, improving system reliability and processing efficiency. He demonstrates that CNN heads routinely output class labels that drive downstream actions, making it natural to adapt the same mechanism in Zhengyang for driver-state reporting. By 2018-2020, it was well-known in the field that CNNs used class-label outputs to categorize driver states such as "attentive," "phone use," or "other distraction." *See* corroborating evidence EX1034; EX1035.

564. **Reasonable Expectation of Success:** A POSITA would have expected this combination to work without issue. Both Zhengyang and He use the same type of architecture, convolutional networks with multiple prediction heads. Adding He's class-label output to Zhengyang's model would only require a simple software adjustment and follows standard computer-vision practices. A POSITA would have expected this change to work easily. It only requires simple logic to merge the existing binary results, for example: if phone = 1, then mobile usage; if smoking = 1, then other distracted; otherwise, undistracted. Since Zhengyang already generates these binary outputs, no change to the network's design would be needed. By the priority date of the'234 patent, driver-monitoring systems were already using similar CNN-based pipelines to produce labeled distraction states for each driver frame, showing that this approach was proven and reliable. *See* corroborating evidence EX1032; EX1034; EX1035.

565. Accordingly, the combination of Zhengyang and He renders Claim 9 obvious. Zhengyang provides the driver-state recognition framework and distraction-related outputs; He adds the explicit class-label prediction mechanism that transforms those outputs into labeled categories. A POSITA would readily combine these teachings to produce a distraction-classification head that classifies each input image with a label corresponding to "undistracted," "mobile-usage," or "other-distracted." The combination involves only well-understood, routine steps

in CNN design, achieves the precise functionality recited in the claim, and would have been expected to succeed without difficulty.

### I. Claim 10: The method of claim 1, wherein the object detection prediction head detects a location of an object within an input vector.

566. Zhengyang discloses this claim.

567. A POSITA would understand that Zhengyang teaches the claimed limitation through its description of the attention proposal sub-network (APN). Within Zhengyang's multi-task driver-state-recognition system, the APN is part of the attention module used in each attribute-recognition head, including the smoking-recognition and phone-call-recognition heads, and functions to determine where the relevant distraction-related object (for example, a cigarette or a mobile phone) appears within the driver image. Zhengyang's APN therefore serves as the localization stage that identifies the spatial region most likely to contain the object associated with the recognized attribute.

568. Zhengyang explains that each attribute-recognition head includes an attention module composed of an attention proposal sub-network and an attention network. The APN is described as outputting spatial coordinates that define the region of interest: "The attention proposal sub-network mainly comprises two fully connected layers in series, which are used to output the center point coordinates of the attention area and the edge lengths of the attention area to perform cropping

and enlargement processing on the original input image." EX1007, ¶[0034]. A POSITA would recognize Zhengyang thus describes the prediction of spatial parameters (e.g., center coordinates and edge lengths) that together define a bounding box around a region of visual interest. In the context of Zhengyang's smoking-and phone-recognition tasks, those bounding boxes correspond to the physical locations of the driver's hands, mouth, or the handheld object itself, which are the salient indicators of distraction behavior. Although Zhengyang does not explicitly use the term "object," a POSITA would appreciate that predicting a discrete region corresponding to a phone, cigarette, or hand functions identically to detecting the location of an object in standard CNN practice.

569. Zhengyang further states that "the attention module comprises an attention proposal sub-network and an attention network, which are used for attribute recognition and optimization of the attribute recognition results of the main network." EX1007, ¶[0033]. A POSITA would understand this to mean that the APN performs the localization stage of the recognition process (i.e., identifying where the target object relevant to the attribute (e.g., phone use or smoking) appears in the image) so that the downstream attention network can perform refined classification within that localized region. This division of labor between localization and classification mirrors the structure of two-stage detection frameworks that were widely used in computer-vision systems at the time.

570. Later portions of Zhengyang confirm that the APN's coordinate outputs are used for spatial localization and iterative refinement: "The attention proposal sub-network comprises two fully connected layers in series … to perform cropping and enlargement processing on the feature map." EX1007, ¶[0101]. "The enlarged image will be input into the first-level attention model to re-judge the attributes … and the APN generated by the first-level attention model will further crop and enlarge the image and input it into the second-level attention model to re-judge the attributes." EX1007, ¶[0102]. A POSITA would understand that this process forms a detection-refinement loop: first, the APN predicts bounding-box coordinates; next, those coordinates are used to crop the relevant region; and finally, the cropped region is analyzed again for more accurate recognition. This approach is conceptually similar to the region-proposal and classification steps used in architectures like Faster R-CNN.

571. In Zhengyang, each input vector represents an image or feature map of a single driver frame. A POSITA would understand that in a convolutional network, each image is converted into a numerical feature vector. When the APN outputs the center coordinates and edge lengths of the attention area, it's identifying where the distraction-related object appears in that feature vector. These coordinates come directly from the image's feature data, satisfying the claim requirement of detecting an object's location within an input vector.

572. Accordingly, Zhengyang's attention proposal sub-network performs the claimed object-detection operation. Because the APN operates as the first sub-component of each prediction head and outputs spatial coordinates that define a region of interest corresponding to the object of distraction, a POSITA would view it as part of the "object-detection prediction head" within the meaning of the claim. Its coordinate outputs predict the spatial location of the relevant distraction-related object within the driver image, and its integration with downstream attention networks provides the subsequent classification refinement. Taken together, these features mirror the region-proposal and localization functions in standard object-detection pipelines, thereby satisfying the limitation of detecting a location of an object within an input vector. A POSITA would therefore conclude that Zhengyang teaches a component that "detects a location of an object within an input vector," satisfying the limitation of Claim 10.

### J. Claim 11: The method of claim 1, wherein the pose estimation prediction head detects an orientation of one or more body parts of a user depicted in an input vector.

573. Zhengyang in view of He renders this claim obvious.

574. Zhengyang provides the multi-task driver-monitoring architecture with shared convolutional features and multiple prediction heads, while He contributes an explicit, person-centric pose-estimation head that detects human body orientation from those same shared feature maps.

575. As established in connection with Element [1F] (*supra* ¶¶414-428), Zhengyang in view of He renders obvious the inclusion of a pose-estimation prediction head within Zhengyang's multi-task architecture. Zhengyang provides the foundational framework, comprising multiple prediction heads, including classification heads for smoking and phone use and an object-detection head embodied by the attention proposal sub-network (APN), that is structurally compatible with additional branches. He explicitly teaches a pose-estimation prediction head designed to detect human body orientations and operating on the same type of backbone, producing per-joint orientation outputs. Accordingly, the same reasoning that supports the obviousness of Element [1C] applies equally here: a POSITA would have found it an obvious and predictable extension to incorporate He's pose-estimation head into Zhengyang's driver-monitoring system, thereby producing a network capable of detecting the orientation of one or more body parts from the input vector as recited in Claim 11.

### 1. Zhengyang

#### (a) Zhengyang's Multi-Task Architecture Provides the Structural Foundation.

576. Zhengyang teaches acquiring an image of a driver via an in-vehicle camera and inputting that image into a multi-task neural network for feature extraction and recognition: "The method comprising: [a]cquiring image data of a

driver in a driving scenario camera video; [and] [i]nputting the image data into a pre-trained multi-task network model for attribute recognition." Zhengyang ¶[0006]-[0008]. In convolutional-network terminology, each image is represented internally as a numerical feature vector; thus, any head operating on these shared feature maps detects information within an "input vector" as claimed. As discussed in Element [1F] (*supra* ¶¶414-428), Zhengyang's network processes driver images to generate detailed spatial feature maps and attention regions centered on the head and hands. Although these modules are used for detecting driver attributes such as mask wearing or phone use, the same processing captures geometric and spatial cues that reflect the driver's physical configuration. A POSITA would recognize that this same backbone and attention structure could readily support an additional prediction head designed to estimate the orientation or position of these body parts without changing the core network design. Incorporating such a head would have been an obvious and predictable modification in light of He et al. (Mask R-CNN), which teaches a modular pose-estimation head that attaches to shared convolutional feature maps.

**(b) Spatial Outputs in Zhengyang Capture Orientation Cues.**

577. As detailed previously (*supra* ¶¶409-413), Zhengyang's attention-proposal sub-network outputs spatial parameters that define the center coordinates

and edge lengths of attention areas, effectively generating bounding boxes for driver regions of interest: "The attention proposal sub-network mainly comprises two fully connected layers in series, which are used to output the center point coordinates of the attention area and the edge lengths of the attention area to perform cropping and enlargement processing on the original input image." EX1007, ¶[0034]. These outputs specify where regions such as the driver's face, hands, or arms appear within the image and at what scale. Through iterative cropping and refinement (EX1007, ¶[0102]), the model tracks changes in these regions, producing spatial data that capture body alignment and relative positioning. A POSITA would understand that these spatial features provide the same type of localized input signals that He's pose-estimation head uses to determine orientation or keypoint direction. Thus, Zhengyang provides the spatial framework, and He supplies the explicit architectural mechanism for extending those spatial cues into pose-orientation predictions, together rendering the claimed pose-estimation prediction head obvious.

**2. He**

**(a) He (Mask R-CNN) Provides an Explicit Pose-Estimation Prediction Head.**

578. He describes a pose-estimation, or keypoint-detection, head that receives an input vector representing image data and detects the position and

orientation of body parts from shared convolutional features. In Section 5, He explains that the Mask R-CNN network "can easily be extended to human pose estimation," where each anatomical keypoint, such as the left shoulder or right elbow, is represented as a one-hot spatial mask and predicted by a dedicated output head: "We model a keypoint's location as a one-hot mask, and adopt Mask R-CNN to predict K masks, one for each of K keypoint types … For each of the K keypoints of an instance, the training target is a one-hot m × m binary mask …." EX1008, 2986. Each keypoint corresponds to a human anatomical feature such as the head, shoulders, or elbowswhich are directly relevant to driver monitoring.

579. He further explains that this keypoint head is a separate branch composed of eight 3 × 3 convolutional layers, followed by a deconvolution layer and 2× bilinear upscaling, producing a 56 × 56 heatmap for each keypoint: "The keypoint head consists of a stack of eight 3 × 3 512-d conv layers, followed by a deconv layer and 2× bilinear upscaling, producing an output resolution of 56×56." EX1008, 2986. Each heatmap shows how the network responds to the input vector (the driver's image data) for a specific body part. The brightest point on the heatmap marks that part's location, and the pattern of these peaks across all K heatmaps represents the overall body orientation. The head is trained with a pixel-wise loss that drives the model to produce one strong activation for each joint, capturing both its position and direction within the input vector.

580. A POSITA would recognize that this architecture corresponds directly to the claimed limitation, namely, a pose-estimation prediction head that detects the orientation of one or more body parts of a user depicted in an input vector. Because He's keypoint branch operates on shared convolutional features derived from the input image vector and is integrated alongside detection and segmentation heads within the same ResNet-FPN backbone, it would fit naturally within Zhengyang's multi-head network framework, enabling orientation detection to be added with minimal modification.

**3. MTC**

581. As discussed in Element [1F] (*supra* ¶¶414-428), a POSITA would have been motivated to combine the teachings of Zhengyang and He. Zhengyang discloses a robust multi-task CNN architecture with a shared backbone (SE-ResNet18) and multiple prediction heads, including classification heads for smoking and phone-use detection and an attention-based localization head (APN). He teaches a modular, keypoint-based pose-estimation head designed to attach to the same type of shared feature maps. Both systems address human-centric vision tasks using comparable multi-branch architectures, and their combination would predictably improve Zhengyang's driver-monitoring network by adding orientation awareness. Zhengyang seeks "to comprehensively reflect the driver's driving state, thereby improving the accuracy of driver state recognition." EX1007, ¶[0046].

He's keypoint head provides exactly the missing orientation component needed to achieve that comprehensive understanding. A POSITA would therefore have been motivated to add He's branch to enrich Zhengyang's recognition outputs with orientation data which is precisely aligned with Zhengyang's stated goal.

582. **How:** A POSITA would have implemented He's keypoint-detection (pose-estimation) head as an additional branch within Zhengyang's existing multi-task framework. Zhengyang already produces region-focused feature maps for the driver's head, hands, and face using its attention proposal subnetwork. He's keypoint branch operates on such feature maps and outputs per-joint heatmaps that represent the location and direction of each body part. Integration would involve connecting He's head to Zhengyang's shared backbone at the same stage used for other heads and training the new branch with labeled driver-pose data using standard pixel-wise regression or classification losses. Because both systems rely on shared convolutional backbones, multi-head training, and supervised optimization, this addition would require minimal structural modification.

583. **Why:** POSITA would have been motivated to make this modification to enhance Zhengyang's ability to detect and interpret distraction behaviors more accurately. Zhengyang's existing heads identify what the driver is doing (e.g., holding a phone or smoking) but not how the driver's body is positioned. Incorporating He's pose-estimation head enables detection of spatial orientation,

such as whether the driver's head is turned toward a phone or the arm is extended away from the steering wheel, which provides richer behavioral context for determining distraction or drowsiness. *See* corroborating evidence EX1036 (showing that head-pose tracking improves drowsiness detection). Adding He's head would therefore provide a complementary and well-understood enhancement that improves the granularity and reliability of Zhengyang's driver-monitoring outputs, consistent with multi-task-learning principles favoring shared-feature architectures for efficiency and accuracy.

584. **Reasonable Expectation of Success:** A POSITA would have had a clear and reasonable expectation of success in making this combination: (1) **Architectural Compatibility.** Both Zhengyang and He use deep convolutional backbones that produce spatially detailed feature maps shared across multiple prediction heads. Zhengyang's SE-ResNet18 backbone generates feature representations of the driver's head, hands, and face for use by its classification and localization modules. He's Mask R-CNN framework relies on the same type of shared convolutional maps to support its detection, segmentation, and pose-estimation branches. Because both systems are built around shared-feature multi-task learning, a POSITA would have recognized that He's keypoint head could be added to Zhengyang's network without redesigning the backbone or changing how features are generated. (2) **Operationally Compatible Systems.** He explicitly

demonstrates that its pose-estimation head can run in parallel with other prediction tasks, noting that "a unified model [] can simultaneously predict boxes, segments, and keypoints while running at 5 fps." EX1008, 2987. This shows that multi-branch training with shared features was routine and reliable. Zhengyang likewise trains multiple heads. classification and localization, using a common backbone and multi-loss optimization. A POSITA would therefore view the two systems as operationally compatible, where adding one more branch for pose estimation would simply extend an already proven pattern. (3) **Available Data**. By the priority date, a POSITA would have understood that the same data and training methods used in Zhengyang would support a pose-estimation branch. Zhengyang already trains multiple heads using supervised learning, where each head outputs spatial or categorical results derived from labeled driver images. He's pose-estimation head is trained in an identical fashion, using image inputs and pixel-level labels to predict spatial outputs. The same loss functions applied in Zhengyang's classification and localization tasks (e.g., categorical or pixel-wise loss) would apply naturally to training a pose-estimation head on corresponding regions of the driver image. A POSITA would therefore recognize that no new data structures or learning techniques were needed; the existing labeled image data and training process would directly accommodate the added pose-estimation task; and (4) **Predictable Outcome**. Combining Zhengyang's multi-task framework with

He's keypoint head would have predictably produced a unified model that detects both what the driver is doing and how the driver's body is oriented. Zhengyang's existing classification heads would continue to recognize distraction behaviors such as phone use or smoking, while He's pose-estimation head would add directional cues like head pose or arm position. The output of the combined model would therefore be richer and more interpretable, providing a more accurate assessment of driver distraction and attentiveness without introducing any technical uncertainty.

585. Accordingly, Zhengyang in view of He renders Claim 11 obvious. Zhengyang supplies the shared-backbone multi-task structure and spatial attention framework; He provides an explicit, modular pose-estimation head that detects orientation from the same type of feature maps. Their combination would have been straightforward, well-motivated, and expected to succeed, producing a system that detects both what the driver is doing and how the driver's body is oriented, thereby meeting the limitation of Claim 11.

### K. Claim 12: The method of claim 1, wherein storing a distraction classification model comprises storing the distraction classification model in a dash-mounted camera device.

586. Zhengyang discloses claim 12.

587. Zhengyang provides the physical, dash-mounted camera architecture with onboard processing and memory but does not expressly disclose storing the trained model within that device.

588. As established in connection with Element [1H] (*supra* ¶¶443-449), Zhengyang discloses storing a trained distraction-classification model comprising backbone and head parameters.

**Zhengyang: In-Vehicle, Camera-Based Device with Local Processing and Memory**

589. **Camera-Integrated Hardware.** Zhengyang describes a system that "acquires a face image of a driver by means of a camera and inputs the image into the multi-task main network for feature extraction and recognition." EX1007, ¶[0014]. Zhengyang further states that the device comprises a memory and a processor implementing the multi-task attribute-recognition method. *Id.*, ¶[0045]. A POSITA would recognize this as describing the typical dash-or windshield-mounted driver-monitoring camera unit used in vehicles at the time; a single housing that includes a camera sensor, processor, and local flash or eMMC memory for real-time inference. The description maps directly to the embedded implementation pattern that was standard in commercial IVMS modules, namely a single housing mounted at the windshield or dash region containing: (1) an

imaging sensor, (2) a processor, and (3) local non-volatile memory used for persistent model storage.

590. **Pruning and Model Selection Imply Local Deployment.** Zhengyang explains that the final trained model is selected based on test accuracy and that the network that does not participate in the final attribute-recognition output to reduce model size and forward-transmission time. EX1007, ¶¶[0038]-[0039]. This indicates that inference is intended to occur *within* the same hardware unit that captures the driver image, minimizing data transmission delays. A POSITA would therefore understand that Zhengyang discoses that the finalized, converged model parameters are stored locally in the device's non-volatile memory (e.g., flash or eMMC) allowing the embedded processor to perform real-time driver-state recognition directly on captured images.

591. Based on the above technical disclosure, it is my opinion that Zhengyang itself teaches storing the converged distraction-classification model within the memory of the dash-mounted camera device so that the local processor within that same unit can execute the inference pipeline directly on incoming in-cabin images.

### L. Claims 13-18 and 20-24

592. Claims 13-18 and 20-24 correspond to the method claims 1-12, but are presented in alternative statutory forms (computer-readable medium and

device). Specifically, claims 13-18 recite a *non-transitory computer-readable storage medium* storing program instructions that, when executed, cause a processor to perform the same steps as the corresponding method claims 1-7; and claims 20-24 recite a *device* including at least one processor and storage configured to perform those same steps.

**1. Claim 13 (CRM version of claim 1).**

593. Claim 13 mirrors the method of claim 1 except that it recites a non-transitory medium storing program instructions for performing the same multi-task training process:inputting labeled driver images into a neural network with a backbone and multiple prediction heads, minimizing loss across those heads, and storing the trained model. Thus, for the same reasons explained above with respect to claim 1, a POSITA would have recognized that the steps of training a multi-task neural network as recited in claim 13 would have been obvious in view of Zhengyang and He. *See* Section X.B.

**2. Claim 14 (CRM version of claim 2).**

594. Claim 14 depends from 13 and corresponds to dependent claim 2, specifying that the distraction-classification head includes convolutional, pooling, and fully-connected layers. For the same reasons explained above with respect to claim 2, a POSITA would have recognized that the features of the distraction-

classification prediction head recited in claim 14 would have been obvious in view of Zhengyang and He. *See* Section X.C.

**3. Claims 15-17 (CRM versions of claims 3-5).**

595. Claim 15 parallels claim 3, specifying an object-detection head comprising bounding-box-regression and class-prediction networks. Claim 16 parallels claim 4, describing those networks as including convolutional and fully-connected hidden layers. Claim 17 parallels claim 5, reciting a pose-estimation head comprising convolutional and fully-connected stacks. For the same reasons explained above with respect to claims 3-5, a POSITA would have recognized that the object-detection and pose-estimation heads recited in claims 15-17 would have been obvious in view of Zhengyang and He. *See* Section X.D.

**4. Claim 18 (CRM versions of claim 6).**

596. Claim 18 recites inputting image data into the trained distraction-classification model and receiving a classification tag; claim 19 further recites performing an action based on that tag. For the same reasons explained above with respect to claim 6, a POSITA would have recognized that inputting image data into the trained model, receiving a distraction-classification tag, and performing an action based on that tag, as recited in claims 18 and 19, would have been obvious in view of Zhengyang and He. *See* Section X.G.

**5. Claim 20 (Device version of claim 1).**

597. Claim 20 recites a device comprising a processor and storage medium configured to perform the same functional steps as the method of claim 1 and the CRM of claim 13. For the same reasons explained above with respect to claim 1, a POSITA would have recognized that implementing the same method steps within a processor-based device as recited in claim 20 would have been obvious in view of Zhengyang and He. *See* Section X.B.

**6. Claim 21-23 (Device version of claims 3-5).**

598. Claim 21 parallels claim 3 and claim 15, reciting a device whose processor executes bounding-box-regression and class-prediction networks. Claims 22 and 23 recite that the object-detection and pose-estimation heads comprise convolutional and fully-connected hidden layers, as in claims 4-5 and 16-17. For the same reasons explained above with respect to claims 3-5, a POSITA would have recognized that implementing the object-detection and pose-estimation heads within a processor-based device, as recited in claims 21-23, would have been obvious in view of Zhengyang amd He. *See* Sections X.D-F.

**7. Claim 24 (Device versions of claim 6).**

599. Claim 24 recites inputting image data into the model and receiving a classification tag. These correspond directly to claims 6 and 18. For the same reasons explained above with respect to claim 6, a POSITA would have recognized

that inputting image data, receiving a distraction-classification tag, and performing an action based on that tag within a processor-based device, as recited in claims 24, would have been obvious in view of Zhengyang and He. *See* Section X.G.

600. Accordingly, claims 13-18 and 20-24 add no substantive limitations beyond restating the method claims in computer-readable-medium or device form. Thus, for the same reasons discussed with respect to claims 1-12, claims 13-18 and 20-24 are rendered obvious by Zhengyang in view of He.

## XI. GROUND 4: ZHENGYANG IN VIEW OF HE AND FURTHER IN VIEW OF CHAO RENDERS OBVIOUS CLAIMS 7-8, 19, AND 25-33

### A. Claims 7, 8, 19, and 25

601. Zhengyang in view of He and further in view of Chao renders obvious claims 7, 8, 19 and 25. Claims 19 and 25 merely restate the same functional recitations of claims 7 and 8, but in CRM and device form.

602. Claim 7 adds the step of performing at least one downstream action based on the distraction-classification tag. Once the model outputs the categorical label (for example smoking or phone use), the system then performs some action that uses that tag. That action does not need to be specified yet. The requirement is simply that at least one action is taken because of the tag.

603. Claim 8 then gives specific examples of such actions by identifying three categories: (1) updating a display of a computing device, (2) playing an

audible alarm, and (3) transmitting a prediction to a remote server. A POSITA would read claim 8 as confirming that the downstream "action" from claim 7 includes these typical HMI/IVMS response channels. Claim 8 is satisfied by demonstrating a system that performs a single one of those actions in response to the distraction-classification tag.

**1. Zhengyang/He**

604. As discussed for claim 6 (Ground 3), a POSITA would understand that Zhengyang in view of He, discloses or renders obvious the generation of a distraction classification tag that identifies driver states such as smoking or phone use. Building upon that foundation, it would have been obvious to a POSITA that the system performs or enables a downstream action based on that tag, particularly when considered in light of Chao, which expressly teaches using model-prediction outputs to trigger vehicle control or safety actions in an autonomous-driving context.

605. Zhengyang explicitly teaches that its invention "relates to the field of intelligent vehicles" and discloses "a multi-task attribute recognition method, apparatus, medium, and device for driving scenarios … to simultaneously recognize multiple features of the driver and output an output result that can comprehensively reflect the driver's driving state, thereby improving the accuracy of driver state recognition." EX1007, Summary, ¶[0002]. A POSITA would

understand or found it obvious that a system designed "to improve driving safety" would act on the information it produces and generate actionable feedback. Zhengyang's model identifies specific distraction states, "whether there is smoking, and whether a phone call is being made" (EX1007, ¶¶[0021], [0028]), and outputs these as binary recognition results. Such results function as actionable signals that would naturally feed into the vehicle's driver-monitoring or alerting subsystems, as would have been recognized by a POSITA.

606. Zhengyang further emphasizes that its network output "reflects the driver's driving state and improves the accuracy of driver-state recognition." EX1007, ¶[0046]. A POSITA would recognize that "reflecting the state" in an intelligent-vehicle environment is a functional step in a feedback loop that enables responsive safety actions, such as triggering a dashboard alert, activating an audible warning, or logging an event in the vehicle's ECU.

607. Additionally, Zhengyang explicitly discloses an architecture that includes the very components required by Claim 8. Zhengyang includes an output apparatus with a display and speaker (EX1007, ¶[0152]), illustrating that its system environment already includes the hardware capable of communicating model results. While Zhengyang does not expressly describe the exact output operation, these components would have made it obvious for a POSITA to present or transmit the distraction-classification results using those same interfaces. A POSITA would

recognize these elements as standard I/O subsystems in intelligent-vehicle or driver-monitoring applications. Zhengyang explains that the model determines "whether there is smoking, and whether a phone call is being made" (EX1007, ¶[0021]) and its express goal is "to improve the accuracy of driver state recognition." (EX1007, ¶[0002]). These disclosures indicate and suggest that the model's classification results, i.e., the distraction-classification tags, are intended to be communicated to the driver or vehicle systems through available output channels. Zhengyang's computing-device embodiment (EX1007, ¶¶[0149]-[0153]) explicitly associates these input and output components with the human-machine interface of the in-vehicle monitoring system. This configuration suggests the conclusion that the classification outputs are intended for functional use, such as updating a driver-facing display, generating an audio alert, or transmitting data for remote analysis, rather than remaining as internal model signals.

608. In practice, a POSITA would understand that Zhengyang's system naturally supports at least three response types: (1) Visual feedback through the display (e.g., a dashboard icon or status change to indicate "phone use detected"); (2) Audible feedback via the speaker (e.g., a chime or verbal warning); and (3) Remote communication through the network interface (e.g., transmitting a driver-state record to a cloud server or fleet log). These functions directly correspond to

the claimed "updating a display," "playing an audible alarm," and "transmitting a prediction to a remote server.

609. In my opinion, a POSITA would understand that Zhengyang discloses an end-to-end driver-state recognition pipeline in which a shared SE-ResNet18 backbone produces spatial feature maps that fan-out into multiple attention heads, including heads that output categorical distraction results such as smoking and phone-use. *See* EX1007, ¶¶[0021], [0028], [0046], [0100]-[0103]. Zhengyang frames these categorical outputs as the direct mechanism to "comprehensively reflect the driver's driving state, thereby improving the accuracy of driver state recognition." EX1007, ¶[0046]. In my view, a POSITA would not interpret a safety-motivated IVMS architecture as producing these class labels merely to discard them. The logical reading is that they exist to be used downstream.

610. He reinforces the standard engineering interpretation of such labels. He discloses a multi-head CNN in which the classification branch determines the control path taken downstream: "two outputs for each candidate object, a class label and a bounding-box offset" and "we only use the k-th mask where k is the predicted class." EX1008, 2982-2983. In my view, that is exactly the conventional multi-branch gating pattern: a classification output is the control signal that determines which next action executes. A POSITA would directly analogize this to the gating variable that conditions HMI or IVMS behaviors in a vehicle system.

611. He discloses a comparable multi-task convolutional architecture (Mask R-CNN) that processes image data through a shared backbone and produces multiple outputs, including a classification branch that assigns a class label and other branches that perform further analysis (e.g., segmentation, keypoints). EX1008, 2982 ("two outputs for each candidate object, a class label and a bounding-box offset").

612. He shows that CNN classification outputs routinely control later processing stages within a multi-head framework, reinforcing the general engineering principle that model outputs can determine subsequent system behavior. The classification branch in He functions as the decision point for subsequent processing/action: the system "only use[s] the k-th mask, where k is the predicted class by the classification branch." EX1008, 2983. He describes not merely inference but utilization of its CNN outputs to drive downstream control or display actions, e.g., using predicted keypoints or regions to localize and annotate subjects in real time. These constitute functional "actions" within the meaning of the claim, which encompasses system behaviors responsive to classification results rather than abstract data generation alone. While this describes an internal processing step rather than an external vehicle action, it demonstrates the general principle that CNN outputs routinely control later operations within multi-head frameworks. A POSITA would therefore view He's classification-driven mask

selection as analogous engineering logic to trigger downstream processes (e.g., alerts or control signals) once a class decision is made. A POSITA would analogize this internal gating behavior to external signal routing in vehicle systems, where the same logic principle, triggering subsequent processes based on classification results, can be applied to visual or audible alerts. A POSITA would understand that this principle applies directly to Zhengyang's architecture: once the model identifies a distraction category (e.g., "smoking," "phone use"), that classification output serves as a decision signal governing the next operation, whether to update a display, sound a warning, or send data to a remote endpoint. He therefore reinforces that post-classification actions like those recited in Claim 8 are a natural and expected part of multi-head CNN systems, both for internal computation and for external HMI responses in automotive context

613. Accordingly, a POSITA would interpret Zhengyang's attention-head outputs (e.g., "smoking," "phone use") as categorical decisions that function like He's predicted class labels; signals that determine what action the system performs next. Together, Zhengyang and He suggest the structure and functional logic of producing classification outputs that govern subsequent operations within the vehicle system. A POSITA would find this consistent with Zhengyang's framework, where each attention head outputs a categorical tag (e.g., smoking, phone use). The same logic applies: once the model determines that a specific

distraction is present, that tag becomes a control signal that informs what the vehicle or driver-monitoring module does next.

614. In my view, the breadth of claim 7 encompasses both external actions and internal control-flow actions. Claim 7 only requires "performing at least one action based on the distraction classification tag," without limiting that action to a specific output channel or to a user-facing notification. He expressly discloses that the predicted class label is used as the decision variable that drives which subsequent operation executes, stating that the model outputs "a class label and a bounding-box offset" and that the system "only use[s] the k-th mask where k is the predicted class." EX1008, 2983-2984. A POSITA would recognize that this mask-selection behavior is an internal system action triggered by the predicted class. Under the broad language of claim 7, that class-driven control selection is itself an "action based on the distraction classification tag." Therefore, even without resort to Chao, Zhengyang in view of He already satisfies the core requirement of claim 7.

615. Together, Zhengyang and He (although not expressly described), at a minimum, strongly suggest the structure and functional logic of producing classification outputs that drive further system actions. However, to the extent it is argued to the contrary, producing classification outputs that drive further system actions would have been obvious in view of Chao.

**2. Chao**

616. Chao discloses a comparable multi-task convolutional architecture that processes image data through a shared backbone and produces multiple outputs, including a classification branch that assigns a class label of a driver's detected state. Chao then utilizes the determined state for "triggering the vehicular action based on the determined state." EX1006, ¶[0129]. The determined state (e.g., "texting," "drinking," "full attention") serves as the decision variable for subsequent control responses.

617. Chao provides a methods and devices that "use in-vehicle passenger monitoring data" to "provide accurate and quick mechanisms to recognize a driver's state" using "one or more in-vehicle data acquisition devices (e.g. a camera providing RGB, depth, and/or infrared data)." EX1006, ¶[0022]. The methods and devices then use the driver's state determined by the "spatio-temporal analysis modeling scheme to trigger vehicular actions," including "providing an appropriate distance from a vehicle located in front or providing a notification to the driver of the vehicle." *Id*., ¶[0022]. Thus, a vehicle "may be capable of sensing its environment and/or sense changes inside the vehicle and navigate without direct human input and/or provide notifications to occupants of the vehicle." *Id*., ¶[0068]. In some instances, the determined state may trigger actions including "modifying one or more of an acceleration, braking, and/or steering of the vehicle or

communicating one or more notifications via the UIs." *Id*., ¶[0095]. The notification may include "a visual notification, a sound notification, or a vibration notification." *Id*., ¶[0168]. These disclosures correspond exactly to the claim's requirement of performing at least one action based on the distraction-classification tag.

618. In regards to updating a display of a computing device or playing an audible alarm, the determined state may trigger actions including "communicating one or more notifications via the UIs." *Id*., ¶[0095]. As described, the vehicular action "may be sent to other parts of the vehicle, such as vehicular controller 320 of mobility system 120 and/or the application processor 216 to send a notification to one or more passengers via one or more of the UIs 206." *Id*., ¶[0117]. The notification may include "a visual notification, a sound notification, or a vibration notification." *Id*., ¶[0168]. A POSITA would understand that displaying a visual notification communicated to a UI would be the same as updating a display of a computing device and a sound notification is an audible alarm.

619. In regards to transmitting a prediction to a remote server, Chao describes that a perception system 302 "may determine state data for each object/feature" to determine the classification of a driver. *Id*., ¶[0075]. Using the output of perception system 302, a planning system provides "a plan to a vehicle controller 320 of the mobility system 120 that controls one or more vehicle

controls such as Engine Control 322, Brake Control 324, and/or Steer Control 326 to execute the plan," in response to a driver's state detection and "may additionally or alternatively provide a notification to the Application Processor 216 to communicate via one or more UIs 206." *Id*., ¶[0078]. A POSITA would understand that mobility system 120 or Application Processor 216 may be a remote entity separate from perception system 302, as "any single processor, controller, or logic circuit detailed herein may be realized as two (or more) separate entities with equivalent functionality." *Id*., ¶[0032].

620. A POSITA would recognize that these notifications and transmissions are all triggered based on the distraction-classification tag output by the model's SoftMax layer. Accordingly, Chao supplies the explicit mechanism by which the predictions from Chun or Chao can be routed to a speaker, display, or communication module for a safety-related action in real time.

### 3. *Motivation to Combine*

621. In my opinion, a POSITA would have viewed the integration of Zhengyang, He, and Chao as a direct continuation of the same technical pipeline. Zhengyang already generates categorical distraction outputs in real time from driver-facing image streams. He shows the standard control-gating use of class IDs inside a multi-head CNN, where the predicted class is explicitly used to decide which next operation executes. Chao shows the final stage of that same pipeline,

where the determined driver state triggers HMI or vehicle-level actions. All three are describing different pieces of the same flow: perception → class decision → actuation.

622. Technically, the "how" is straightforward. The output tensor produced by the Zhengyang attention head (e.g., softmax logits or class indices) would simply be passed to the same UI / notification / control interface module described in Chao. No new network blocks are required. No changes to the feature extractor are required. The data formats already match. Passing the class ID into Chao's notification/actuation callback is a routine one-line integration at the API boundary where the perception module hands off to the vehicle interface module.

623. The "why" is inherent in the design intent of each reference. Zhengyang is framed as an intelligent vehicle safety system. He demonstrates that class outputs are meant to drive subsequent decisions. Chao's architecture already consumes a class-based state and maps it to screen output, audible output, or control-path actuation. A POSITA would view these three components together as three consecutive layers of the same design stack: classification exists so that something downstream can respond.

624. The expected success of this integration would have been extremely high. All three references assume the same modality (driver-facing image input), the same model family (CNN with multiple heads), the same deployment target

(in-vehicle runtime), and the same interface surface (display, speaker, network output). There is no domain shift, no incompatible data type, no architectural mismatch, and no novel training requirement. This is simply routing a known class decision into an existing vehicle notification or control channel.

625. In my professional view, this is not a speculative or novel connection. This is the way these systems are actually deployed: the visual model produces a categorical state, and that state is forwarded to the HMI or control layer to produce an immediate user-visible or vehicle-visible response. A POSITA would have viewed the Zhengyang → He → Chao chain as a single coherent pipeline and would have expected the integration to succeed on first implementation without special tuning or modification.

### B. Claims 26-33

#### 1. Independent Claim 26

626. Claim 26 recites the same core subject matter as claim 1, with additional limitations corresponding to those already addressed for claims 6-8 and 12, but expressed in the context of a dash-mounted in-vehicle inference device. Each of those component limitations has already been mapped to Zhengyang, He, and Chao. Accordingly, for the same technical reasons, the same combination renders claim 26 obvious.

627. In particular, claim 26[A] which recites storing a distraction-classification model in a dash-mounted camera device corresponds to claim 12. Zhengyang already discloses in-vehicle deployment hardware including display, speaker, and network interfaces where the distraction classification model is stored in the dash-mounted camera device.

628. Claim 26[B] which recites a backbone network coupled to task-specific prediction heads is the same architecture already addressed for claim 1. Zhengyang discloses SE-ResNet18 feature extraction feeding multiple heads including smoking and phone-use; He confirms classification-driven downstream gating; and Chao confirms state-conditioned actuation.

629. Claim 26[C] which recites capturing an interior-vehicle image using a sensor in the dash-mounted camera is disclosed in Zhengyang which employs cabin-mounted cameras for driver-monitoring inference.

630. Claim 26[D] which recites inputting the captured image into the model and receiving a distraction-classification tag is the same inference step already addressed for claim 6; Zhengyang discloses receiving categorical outputs (smoking / phone use), and He confirms class-driven branching.

631. Claim 26[E] which recites performing an action based on that tag corresponds directly to claims 7 and 8; Chao expressly teaches generating display, audio, or remote notifications based on the determined driver state.

### 2. Claims 27-29

632. Claims 27-29 add no new substance beyond claim 26 and correspond to the same refinements already addressed in Ground 4 (classification label, object-location, pose orientation). Each is disclosed for the same reasons in view of Zhengyang, He, and Chao.

### 3. Independent Claim 30

633. Claim 30 recasts claim 26 in device form, requiring a dash-mounted camera device having processor and memory. Zhengyang discloses in-vehicle modules storing executable CNN weights and performing inference with onboard compute. Claims 31-33 then correspond directly to claims 27-29.

634. The only difference is statutory form. The same distraction-classification functionality is performed in the same way, but expressed as logic embodied in a processor-memory device. A POSITA would have understood that the same inference steps can be implemented in software stored in memory and executed by processors in-vehicle.

635. Accordingly, claims 26-33 merely restate the same substantive content in CRM/device form. A POSITA would have found each obvious for the same reasons already articulated in view of Zhengyang, He, and Chao.

## XII. OBJECTIVE INDICIA OF NONOBVIOUSNESS

636. In reaching my opinions regarding obviousness in this declaration, I also considered whether there were any objective indicia of nonobviousness. The basis for my understanding of obviousness and objective indicia of nonobviousness is provided above in Section III.

637. I understand that any objective indicia of nonobviousness must have a nexus, or a close connection, to the novel aspects of the claims. I understand that objective indicia of nonobviousness include the following: commercial success, long-felt but unmet need, failure of others, copying, praise in the industry, and unexpectedly superior results as compared with the closest prior art.

638. Considering the information I have reviewed, I am not aware of any objective evidence that might support the patentability of the claims of the ’243 patent. Further, from my review of the file history (EX1002) of the ’243 patent, I did not see that the applicant raised any objective indicia of nonobviousness during prosecution. I am not otherwise aware of any publicly available evidence of objective indicia of nonobviousness.

639. Therefore, it is my opinion that no objective indicia of nonobviousness support the patentability of the claims of the ’243 patent.

## XIII. CONCLUSION

In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the allotted time.

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Respectfully submitted,



Trevor J. Darrell, Ph.D.

Date: November 14, 2025

# Appendix A

# LIST OF MATERIALS CONSIDERED

| Exhibit | Description |
|---|---|
| 1001 | U.S. Patent No. 12,062,243 to Hassan *et al.* ("Hassan") |
| 1002 | Prosecution History of U.S. Patent No. 12,062,243 |
| 1004 | *Curriculum Vitae* of Dr. Trevor Darrell |
| 1005 | Chun, Sehyun, et al. "NADS-Net: A nimble architecture for driver and seat belt detection via convolutional neural networks." Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. 2019. |
| 1006 | U.S. Publication No. 2020/0242381 to Chao *et al.* ("Chao") |
| 1007 | Chinese Publication No. CN111860253 to Dongguan Zhengyang Electronic Mechanical Co ltd ("DZE") |
| 1008 | He, K., Gkioxari, G., Dollar, P., and Girschick, R. "Mask R-CNN" IEEE International Conference on Computer Vision (2017) |
| 1009 | Caruana, Rich. "Multitask Learning." Machine Learning 28 (1997): 41-75. |
| 1010 | Shanmugamani, R. Deep Learning for Computer Vision (2018) |
| 1011 | P. Smith, M. Shah and N. da Vitoria Lobo, "Determining driver visual attention with one camera," in IEEE Transactions on Intelligent Transportation Systems, vol. 4, no. 4, pp. 205-218, Dec. 2003, doi: 10.1109/TITS.2003.821342. |
| 1012 | Baheti, Bhakti & Gajre, Suhas & Talbar, Sanjay. (2018). Detection of Distracted Driver Using Convolutional Neural Network. 1145-11456. 10.1109/CVPRW.2018.00150. |
| 1013 | M. S. Majdi, S. Ram, J. T. Gill and J. J. Rodríguez, "Drive-Net: Convolutional Network for Driver Distraction Detection," 2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), Las Vegas, NV, USA, 2018, pp. 1-4, doi: 10.1109/SSIAI.2018.8470309. |
| 1014 | U.S. Patent No. 11,176,384 B1 to Yang et al. ("Yang") |

| Exhibit | Description |
|---|---|
| 1015 | U.S. Patent No. 12,347,177 B2 to Li et al. (“Li”) |
| 1016 | R. Ranjan, V. M. Patel and R. Chellappa, "HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 1, pp. 121-135, 1 Jan. 2019, doi: 10.1109/TPAMI.2017.2781233 |
| 1017 | Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587). |
| 1018 | Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448). |
| 1019 | Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137-1149. |
| 1020 | Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., & Tian, Q. (2019). Actional-structural graph convolutional networks for skeleton-based action recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 3595-3603). |
| 1021 | Christoph Maurer, Integration and Deployment of Machine Learning Models (Master’s Thesis, Graz University of Technology, Feb. 2021) |
| 1022 | Umar Zakir Abdul, Hamid; et al. (2016). "Current Collision Mitigation Technologies for Advanced Driver Assistance Systems-A Survey". PERINTIS eJournal. 6 (2). |
| 1023 | Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., & Muller, U. (2017). Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv preprint arXiv:1704.07911. |
| 1024 | G. Gkioxari, B. Hariharan, R. B. Girshick, and J. Malik. R-cnns for pose estimation and action detection. CoRR, abs/1406.5212, 2014 |
| 1025 | LeCun, Y., Bengio, Y., & Hinton, G. (2015). “Deep learning.” Nature, Vol. 521 (pp. 436-444). |

| Exhibit | Description |
|---|---|
| 1026 | Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. |
| 1027 | Goodfellow, I., Bengio, Y. & Courville, A. (2016). Deep Learning. |
| 1028 | He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deeping into Rectifiers: Surpassingb Human-Level Performance on ImageNet Classification. |
| 1029 | Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7132-7141). |
| 1030 | He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. |
| 1031 | Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. |
| 1032 | Abouelnaga, Y., Eraqi, H., & Moustafa, M. (2017). Real-time Distracted Driver Posture Classification. In 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montreal, Canada. |
| 1033 | Kose, N., Kopuklu, O., Unnervik, A., & Rigoll, G. (2019). Real-Time Driver State Monitoring Using a CNN Based Spatio-Temporal Approach. |
| 1034 | Braunagel, C., Stolzmann, W., Kasneci, E., & Rosensteil, W. (2017). Driver Activity Recognition in the Context of Conditionally Autonomous Driving. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems (pp. 1652-1657). |
| 1035 | Engström, J., et al. 2019. A conceptual framework and taxonomy for understanding and categorizing driver inattention. |
| 1036 | Choi, I., & Kim, Y. (2015). Head Pose and Gaze Direction Tracking for Detecting a Drowsy Driver. In Applied Mathematics & Information Sciences 9, No. 2L (pp. 505-512). |
| 1038 | Simonyan, K. & Zisserman A. 2015. Very Deep Convolutional Networks For Large-Scale Image Recognition. |

| Exhibit | Description |
|---|---|
| 1039 | Maurer, C., Integration and Deployment of Machine Learning Models. Graz University of Technology Master's Thesis. (2021). |
| 1040 | Boureau et al. 2010. A Theoretical Analysis of Feature Pooling in Visual Recognition, Proceedings of the 27th International Conference on Machine Learning. |