

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioner,

v.

NEURAL AI, LLC,
Patent Owner.

Case No. IPR2025-00608
U.S. Patent No. RE49,461

Petition for *Inter Partes* Review of U.S. Patent No. RE49,461

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

TABLE OF CONTENTS

	<u>Page</u>
TABLE OF AUTHORITIES	ix
PETITIONER’S EXHIBIT LIST	x
TABLE OF ABBREVIATIONS	xii
MANDATORY NOTICES.....	xiii
A. Real Parties-in-Interest.....	xiii
B. Related Matters [37 C.F.R. § 42.8(b)(2)].....	xiii
C. Lead and Back-Up Counsel [37 C.F.R. § 42.8(b)(3)].....	xiii
D. Service Information [37 C.F.R. § 42.8(b)(4)]	xiv
I. INTRODUCTION	1
II. REQUIREMENTS OF <i>INTER PARTES</i> REVIEW	2
A. Standing.....	2
B. Identification of Challenge and Relief Requested	2
1. U.S. Patent Application Publication No. 2005/0197977 (“Buck”) (Ex1004).....	3
2. U.S. Patent Application Publication No. 2003/0140179 (“Wilt”) (Ex1005)	4
3. Neural Network Toolbox User’s Guide Version 4 (“nnet”) (Ex1006).....	4
4. Z. Luo, Artificial Neural Network Computation on Graphic Process Unit (“ANN”) (Ex1007)	5
5. GPU Gems 2: Programming Techniques for High- Performance Graphics and General-Purpose Computation (“GPU Gems”) (Ex1008).....	6

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
C. How the Challenged Claims Are To Be Construed Under 37 C.F.R. § 42.104(b)(3).....	6
D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.104(b)(4).....	7
E. Supporting Evidence Under 37 C.F.R. § 42.104(b)(5)	7
F. Payment of Fees	7
III. TECHNICAL BACKGROUND	7
A. State of the Art Before the '461 Patent	7
1. GPU Acceleration of Non-Graphics Computations and its Applications	7
2. Resource and Task Management	9
3. APIs and Programming GPUs	10
B. Alleged Invention of the '461 Patent	11
C. Prosecution History of the '461 Patent	14
D. Person of Ordinary Skill in the Art	14
IV. GROUNDS OF UNPATENTABILITY.....	15
A. Grounds 1–6	15
1. Overview of Prior Art	15
a. Buck.....	15
b. Wilt	16
c. nnet.....	18
d. ANN.....	19
e. GPU Gems	20

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
2. Combinations	21
a. Combination of Buck and Wilt (All Grounds).....	21
b. Combination of nnet with Buck and Wilt (Ground 2)	24
c. Combination of ANN (Ground 3) or GPU Gems (Ground 4) with Buck and Wilt.....	25
i. ANN (Ground 3)	26
ii. GPU Gems (Ground 4).....	27
iii. Implementing the Computer Vision Method of ANN or GPU Gems in Buck/Wilt’s System	28
d. Combination of nnet with Buck, Wilt, and ANN (Ground 5) or Buck, Wilt, and GPU Gems (Ground 6)	29
3. Detailed Analysis of Grounds.....	30
a. Claim 21.....	31
i. 21[pre]: “A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:”	31
ii. 21[a]: “executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;”	34

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
iii. 21[b]: “executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;”.....	38
iv. 21[c]: “shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network;”	42
v. 21[d]: “shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network;”	43
vi. 21[e]: “queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network; and”	45
vii. 21[f]: “executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream.”.....	46
b. Claim 22: “The method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network.”	48

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
c. Claim 23: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network.”	49
d. Claim 24: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network.”	51
e. Claim 25: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network.”	52
f. Claim 26: “The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen.”	55
g. Claim 27: “The method of claim 21, wherein executing the user interaction stream comprises: parsing elements to be used in the computations representing the artificial neural network.”	60
h. Claim 28: “The method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises: converting the elements into GPU programs.”	61
i. Claim 29: “The method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs.”	63
j. Claim 30: “The method of claim 29, wherein executing the user interaction stream comprises:	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
transferring the GPU programs to the second memory partition.”.....	63
k. Claim 31: “The method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of the computations representing the artificial neural network to disk.”	64
l. Claim 32: “The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations.”	64
m. Claim 38: “The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network.”	67
n. Claim 39: “The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network.”	69
o. Claim 40: “The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network.”	70
p. Claim 41: “The method of claim 21, wherein, during execution of the computations representing the artificial neural network, the computational stream controls the data exchange between the user interaction stream and the computational stream by requesting the inputs to the artificial neural network.”	72
q. Claim 33.....	73

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
i. 33[pre]: “A system for executing computations representing an artificial neural network, the system comprising:”	73
ii. 33[a]: “a camera to acquire input data for the artificial neural network;”	73
iii. 33[b]: “a first memory partition;”	74
iv. 33[c]: “a second memory partition;”	74
v. 33[d]: “at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network;”	74
vi. 33[e]: “a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and”	76
vii. 33[f]: “a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
computations representing the artificial neural network for performance at times selected to avoid data corruption.”	78
r. Claim 34: “The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network.”	80
s. Claim 35: “The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network.”	81
t. Claim 36: “The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network.”	81
u. Claim 37: “The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network.”	82
V. ENABLEMENT AND SECONDARY CONSIDERATIONS	82
VI. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION.....	83
A. Discretionary Denial Under Section 314(a) Is Not Warranted.....	83
B. Discretionary Denial Under Section 325(d) Is Not Warranted.....	84
VII. CONCLUSION.....	86
APPENDIX: CHALLENGED CLAIM LISTING	

TABLE OF AUTHORITIES

	<u>Page(s)</u>
<i>Adobe Inc. v. RAH Color Techs.</i> , IPR2019–00628, (P.T.A.B. Aug. 20, 2019)	85
<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, (P.T.A.B. Mar. 20, 2020)	83
<i>CrowdStrike, Inc., v Webroot Inc.</i> , IPR2023-00126, (P.T.A.B. May 5, 2023)	84
<i>Fusion Orthopedics, LLC v. Extremity Medical, LLC</i> , IPR2023-00894, (P.T.A.B. Nov. 17, 2023)	84
<i>Ohio Willow Wood Co. v. Alps South, LLC</i> , 735 F.3d 1333 (Fed. Cir. 2013)	83
<i>Samsung Elecs. Co. v. Staton Techiya, LLC</i> , IPR2022-00302, (P.T.A.B. Jul. 11, 2022)	83
<i>TSMC Ltd. v. Alidouble, Inc.</i> , IPR2023-00372, (P.T.A.B. Jul. 13, 2023)	83

PETITIONER’S EXHIBIT LIST

Exhibit	Description
1001	U.S. Patent No. RE49,461
1002	Excerpt of File History for U.S. Patent No. RE49,461 (App. No. 17/136,343)
1003	Declaration of Prof. Tajana Rosing, Ph.D.
1004	U.S. Patent Application Publication No. 2005/0197977 (“Buck”)
1005	U.S. Patent Application Publication No. 2003/0140179 (“Wilt”)
1006	Neural Network Toolbox User’s Guide Version 4 (“nnet”)
1007	Artificial Neural Network Computation on Graphic Process Unit (“ANN”)
1008	GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (2005) (“GPU Gems”)
1009	Plaintiff’s Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court’s Standing Order Governing Patent Proceedings (Jan. 14, 2025)
1010	Declaration of Dr. Mary Bolin
1011	Numerical Recipes in C (2d ed. 2002)
1012	Jeanne Martin, <i>Fortran 90 Pointers vs. “Cray” Pointers</i> , 11 ACM SIGPLAN Fortran Forum (1992)
1013	Arthur Veen, <i>Dataflow Machine Architecture</i> (1986)
1014	Michael Flynn, <i>Some Computer Organizations and Their Effectiveness</i> (1972)
1015	Press Release – NVIDIA Launches the World’s First Graphics Processing Unit; GeForce 256 (Aug. 31, 1999)
1016	Excerpt of OpenGL Shading Language (2004)
1017	Excerpt of The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics (2003)
1018	OpenGL 2.1 Reference Pages
1019	Advanced Image Processing with DirectX 9 Pixel Shaders (2004)

Exhibit	Description
1020	Dominik G�ddecke -- GPGPU::Basic Math Tutorial
1021	Ian Buck, <i>Data Parallel Computation on Graphics Hardware</i> (2003)
1022	Youquan Liu, <i>Real-Time 3D Fluid Simulation on GPU with Complex Obstacles</i> (2004)
1023	Kyoung-Su Oh, <i>GPU implementation of neural networks</i> (2004)
1024	U.S. Patent No. 7,673,304 (“Gosalia”)
1025	Thomas Rolfes, <i>Artificial Neural Networks on Programmable Graphics Hardware</i> in Game Programming Gems 4 (2004)
1026	P.J.G. Lisboa, <i>A review of evidence of health benefits from artificial neural networks in medical intervention</i> (2002)
1027	U.S. Patent No. 7,418,576 (“Lindholm”)
1028	Bertil Svensson, <i>SIMD Processor Array Architectures</i> in PARALLEL PROCESSING IN INDUSTRIAL REAL-TIME APPLICATIONS (1992)
1029	Michael Glover, <i>A Massively-Parallel SIMD Processor for Neural Network and Machine Vision Applications</i> (1993)
1030	Francisco Mesa-Martinez, <i>The UCSC Kestrel High Performance SIMD Processor: Present and Future</i> (2003)
1031	Getting Started with MATLAB, Version 7 (2005)
1032	Declaration of Gordon MacPherson
1033	<i>The C programming Language</i> (1988)
1034	U.S. Patent No. 7,861,060 (“Nickolls”)
1035	U.S. Patent No. 7,139,003 (“Kirk”)
1036	<i>Sotera Stipulation</i> (Mar. 5, 2025)

TABLE OF ABBREVIATIONS

Abbreviation	Term
'343 application	U.S. Patent Appl. No. 17/136,343
'461 patent	U.S. Patent No. RE49,461
'828 patent	U.S. Patent No. 9,189,828
ANN	Artificial Neural Network Computation on Graphic Process Unit
Board	Patent Trial and Appeal Board
Buck	U.S. Patent Application Publication No. 2005/0197977
Buck '085	U.S. Patent No. 7,219,085
Challenged Claims	Claims 21–41 of U.S. Patent No. RE49,461
Ex.	Exhibit
Fig.	Figure
GPU Gems	GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (2005)
IPR	<i>inter partes</i> review
Litigation	<i>Neural AI, LLC v. NVIDIA Corporation</i> , No. 7:24-cv-00221 (W.D. Tex.)
nnet	Neural Network Toolbox User's Guide Version 4
Patent Owner	Neural AI, LLC
Petitioner	NVIDIA Corporation
POSITA[s]	person[s] of ordinary skill in the art
USPTO	United States Patent and Trademark Office
Wilt	U.S. Patent Application Publication No. 2003/0140179

MANDATORY NOTICES

A. Real Parties-in-Interest

NVIDIA Corporation is the real party-in-interest.

B. Related Matters [37 C.F.R. § 42.8(b)(2)]

The '461 patent is asserted in *Neural AI, LLC v. NVIDIA Corporation*, No. 7:24-cv-00221 (W.D. Tex.) (“the Litigation”). Petitioner has initiated the following proceedings as to patents related to the '461 patent and asserted in the Litigation: IPR2025-00606 (U.S. Patent No. 8,648,867) and IPR2025-00609 and IPR2025-00610 (U.S. Patent No. RE48,438).

C. Lead and Back-Up Counsel [37 C.F.R. § 42.8(b)(3)]

Pursuant to 37 C.F.R. §§ 42.8(b)(3), 42.8(b)(4), and 42.10(a), Petitioner provides the following designation of counsel:

Lead Counsel for Petitioner	Back-Up Counsel for Petitioner
Brian M. Buroker (Reg. No. 39,125) Gibson, Dunn & Crutcher LLP 1700 M Street NW Washington, DC 20036 Phone: (202) 955-8500 Fax: (202) 467-0539 Email: bburoker@gibsondunn.com	L. Kieran Kieckhefer (<i>pro hac vice</i> forthcoming) Gibson, Dunn & Crutcher LLP One Embarcadero Center San Francisco, CA 94111 Phone: (415) 393-8337 Email: kkieckhefer@gibsondunn.com

Lead Counsel for Petitioner	Back-Up Counsel for Petitioner
	Nathan Curtis (Reg. No. 70,471) Gibson, Dunn & Crutcher LLP 2001 Ross Avenue Dallas, TX 75201 Phone: (214) 698-3423 Email: ncurtis@gibsondunn.com Vivian Lu (Reg. No. 74,443) Gibson, Dunn & Crutcher LLP 200 Park Avenue New York, NY 10166 Phone: (212) 351-3837 Email: vlu@gibsondunn.com

A Power of Attorney accompanies this Petition in accordance with 37 C.F.R. § 42.10(b).

D. Service Information [37 C.F.R. § 42.8(b)(4)]

Service via hand delivery or postal mail may be made at the addresses of the lead and back-up counsel above. Petitioner hereby consents to electronic service, and service by email may be made at:

- GDC-NVIDIA-IPR@gibsondunn.com

I. INTRODUCTION

Petitioner hereby petitions for IPR of U.S. Patent No. RE49,461 (Ex1001) and cancellation of Claims 21–41¹ as unpatentable under 35 U.S.C. § 103.

The '461 patent is the last in a family of patents directed to a system for performing *general-purpose* computations (such as simulating an artificial neural network) on traditionally *special-purpose* graphics processing units (“GPUs”). The '461 patent’s previous claims were each discarded and replaced by reissue claims drafted with nearly 14 years of hindsight. But the new claims cover only well-known techniques for systems that involve both a CPU and a GPU (called the “processing unit” in the claims), set in the context of artificial neural networks.

For example, the claims recite maintaining separate instruction “streams” (or threads) for execution on the different processors for separate data types, which the patent refers to as “user interaction” (for the CPU) and “computational” streams (for the GPU), Ex. 1001, 2:11–19, 7:60–8:6, and the implementation of artificial neural network computations on a GPU, *id.*, 1:59–63. Stream processing, which is taught in Buck and GPU Gems, was a basic, fundamental concept in underlying CPU and GPU programming in the prior art.

¹ Claims 1–20 were cancelled during reissue.

The claims also recite other basic techniques used in systems with multiple processors. The claims ensure that the CPU cannot corrupt the data being used by the GPU for its calculations by “shifting control of a data exchange” to the GPU while it is executing a calculation and then back to the CPU after the calculation is done or interrupted. Similarly, the claims recite “queueing” commands intended for the GPU while the GPU is performing other calculations so that they are ready when the GPU is next available. These concepts were well-known and taught by Buck, Wilt, and other references discussed herein.

The dependent claims add nothing inventive. Instead, they recite rudimentary elements like deciding which equations to use, setting the parameters for the equations, or deciding which outputs to save to disk or display on a screen.

II. REQUIREMENTS OF *INTER PARTES* REVIEW

A. Standing

Pursuant to 37 C.F.R. § 42.104(a), Petitioner certifies the '461 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR on the grounds identified herein.

B. Identification of Challenge and Relief Requested

Pursuant to 37 C.F.R. § 42.104(b), Petitioner requests the Board institute IPR of Claims 21–41 of the '461 patent under pre-AIA 35 U.S.C. § 103.

The precise relief requested by Petitioner is that Claims 21–41 be cancelled based on the grounds below:

Ground	Claims	Basis for Rejection ²
1	21–41	Obviousness over Buck in view of Wilt
2	23–26, 34–36, 38–40	Obviousness over Buck in view of Wilt and nnet
3	32–37	Obviousness over Buck in view of Wilt and ANN
4	32–37	Obviousness over Buck in view of Wilt and GPU Gems
5	34–36	Obviousness over Buck in view of Wilt, ANN and nnet
6	34–36	Obviousness over Buck in view of Wilt, GPU Gems, and nnet

These challenges rely on the following prior art references:

1. U.S. Patent Application Publication No. 2005/0197977 (“Buck”) (Ex1004)

Buck is a September 8, 2005 publication of a non-provisional application filed June 25, 2004, and claims priority from a provisional application filed December 9, 2003. Buck is assigned to Microsoft. Buck qualifies as prior art to the ’461 patent under pre-AIA 35 U.S.C. §§ 102(a), (b) and/or (e). Buck was not of record during prosecution.

² All obviousness grounds include the knowledge of a POSITA.

2. U.S. Patent Application Publication No. 2003/0140179 (“Wilt”) (Ex1005)

Wilt is a July 24, 2003 publication of a non-provisional application filed January 4, 2002. Wilt is assigned to Microsoft. Wilt qualifies as prior art to the ’461 patent under pre-AIA 35 U.S.C. §§ 102(a) and/or (b). Wilt was not of record during prosecution.

3. Neural Network Toolbox User’s Guide Version 4 (“nnet”) (Ex1006)

nnet was published in 2004 by The MathWorks, Inc., the provider of the well-known computing platform MATLAB. Ex1006, 1; Ex1010, ¶¶50–74. nnet was available online at MathWorks’s website, www.mathworks.com, at least by October 2004, and POSITAs would have been aware of nnet due to their familiarity with MATLAB. nnet would have been indexed through search engines by late 2004, and would have been locatable by keyword searching anything in nnet, such as “neural network.” Ex1010, ¶¶50–74. A POSITA, exercising reasonable diligence and searching for references regarding neural networks, would have located nnet no later than around October 2004, as further evidenced by the fact that contemporary publications in the field cited it. *Id.* nnet was also catalogued by the OCLC by June 24, 2005, and would have been located shortly thereafter by searching for “neural networks,” “modeling,” “simulation,” and “MATLAB.” *Id.* nnet therefore qualifies as prior art to the ’461 patent under at least pre-AIA 35 U.S.C. §§ 102(a) and/or (b). nnet was not of record during prosecution.

4. Artificial Neural Network Computation on Graphic Process Unit (“ANN”) (Ex1007)

ANN (which bears a copyright date from IEEE of 2005) was presented at and published in the Proceedings of the 2005 IEEE International Conference on Neural Networks, held from July 31 to August 4, 2005, in Montreal, Canada. Ex1008, 622; Ex1010, ¶¶31–49; Ex1032. ANN was made available to conference participants no later than the last day of the conference. Ex1010, ¶¶31–49. ANN was added to IEEE Xplore (a well-known, searchable research database) on December 27, 2005, and ANN would have been located by searching for terms such as “artificial neural networks,” “neural networks,” or “graphics processing unit.” *Id.* The conference proceedings including ANN was cataloged in the University of Arizona’s library by January 4, 2006. *Id.* A POSITA, exercising reasonable diligence and searching for references regarding neural networks would have located ANN around January 4, 2006. *Id.* A skilled researcher would have located ANN in the library by searching for the conference specifically or a term such as “neural networks.” *Id.* ANN therefore qualifies as prior art to the ’461 patent under at least pre-AIA 35 U.S.C. §§ 102(a) and/or (b). ANN was of record during prosecution of the ’461 patent but was not substantively discussed or applied by the Examiner.

5. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (“GPU Gems”) (Ex1008)

GPU Gems was published by NVIDIA in March 2005 and publicly available no later than July 2005. Ex1010, ¶¶20–30. GPU Gems’ “[f]irst printing” was March 2005 and it was cataloged and indexed at the University of Texas–Austin by July 18, 2005. *Id.* A POSITA, exercising reasonable diligence and searching for references regarding programming GPUs, would have located GPU Gems in July 2005. *Id.* A skilled researcher would have located GPU Gems by searching for keywords like “GPU” and “programming,” which are contained in the searchable title, or “[c]omputer graphics” or “[r]eal-time programming,” which are contained in the searchable subject headings for GPU Gems. *Id.* GPU Gems therefore qualifies as prior art to the ’461 patent under pre-AIA 35 U.S.C. §§ 102(a) and/or (b). GPU Gems was not of record during prosecution.

C. How the Challenged Claims Are To Be Construed Under 37 C.F.R. § 42.104(b)(3)

Petitioner submits that no construction of any claim term is necessary for the Board to resolve, and that the Challenged Claims would have been obvious under any reasonable construction.

D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.104(b)(4)

An explanation of how the Challenged Claims are unpatentable under the statutory grounds identified above, including the identification of where each element of the claim is found in the prior art, is provided below.

E. Supporting Evidence Under 37 C.F.R. § 42.104(b)(5)

Exhibit numbers of the supporting evidence relied upon and the relevance of that evidence to the Challenged Claims, including an identification of specific portions of the evidence that support the challenge, are provided below. A list of exhibits with the exhibit number and a brief description of each exhibit is also included in this Petition under 37 C.F.R. § 42.63(e). The technical information and grounds for rejection explained in this Petition are further supported by Dr. Tajana Rosing's Declaration (Ex1003) ¶¶1–407.

F. Payment of Fees

Pursuant to 37 C.F.R. §§ 42.103 and 42.15(a), the required fee is being submitted herewith. The Office is authorized to charge any fee deficiency, or credit overpayment, to deposit account no. 50-1408.

III. TECHNICAL BACKGROUND

A. State of the Art Before the '461 Patent

The concepts discussed in the '461 patent are not new. Ex1003, ¶¶31–85.

1. GPU Acceleration of Non-Graphics Computations and its Applications

Parallel processing, for both graphics and general-purpose computing, was known for decades before the '438 patent. Ex1003, ¶¶33–42, 44–51. In 1999, NVIDIA developed the first consumer-level graphics card that could offload the entire graphics pipeline from the CPU, dubbing it a “Graphics Processing Unit,” or “GPU.” *Id.* Unlike CPUs, which excel at handling a few tasks sequentially, GPUs are optimized for parallel processing, making them ideal for the thousands of simultaneous calculations required to render graphics. *Id.* Prior art GPUs operated according to the “stream programming model,” which represents data as a “stream,” or “an ordered set of data of the same data type.” Ex1008, 464; Ex1003, ¶¶52–54. Computations on individual stream elements can be performed independently, allowing serial operations to be mapped onto data-parallel hardware. Ex1008, 465.

The parallel nature of GPU programming contributed directly to the rise in popularity of using GPUs for non-graphics applications. Ex1003, ¶¶33–42. This type of computing is called “general-purpose computing on GPUs,” or “GPGPU.” *Id.* GPGPU was pioneered by NVIDIA and others. *Id.* It was known that many complex computations, including simulating artificial neural networks, can efficiently be implemented on a GPU. Ex1008, 495; Ex1003, ¶¶73–78. Before the '461 patent, artificial neural networks on GPUs were already used in a wide array of applications, such as GPU-based computer vision programs that processed real-time

video data to perform image processing, object tracking, and motion detection. Ex1007; Ex1008, 649; Ex1003, ¶¶73–78.

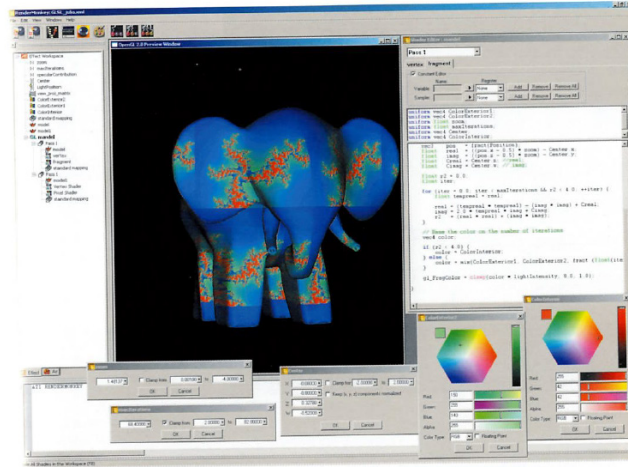
2. Resource and Task Management

GPGPU involves running many tasks concurrently within a single stage of processing and coordinating dependencies between stages of processing. Ex1003, ¶¶81–84. It was therefore known that efficient management of tasks assigned to the GPU and of system resources was needed for optimal utilization of a GPU’s parallel processing capabilities. *Id.* In the prior art, the operating system of a computer was typically responsible for managing resources (*e.g.*, GPUs, CPUs, memory, and software) within the system. *Id.* More specifically, an operating system kernel component called a “scheduler” assigns resources to perform tasks. Schedulers determine when tasks should be executed and make necessary computational resources available to the executing tasks according to scheduling policies that account for requirements such as task priority, timing, and resource needs. *Id.* As tasks come in, they can be processed and queued for later execution. *Id.* By managing tasks that require access to the same data, computational resources currently in-use, or have data dependencies, schedulers optimize efficiency, balance computational loads, and improve data coherency in a system.

These same scheduling techniques had been employed in GPGPU processes before the ’461 patent. Ex1005.

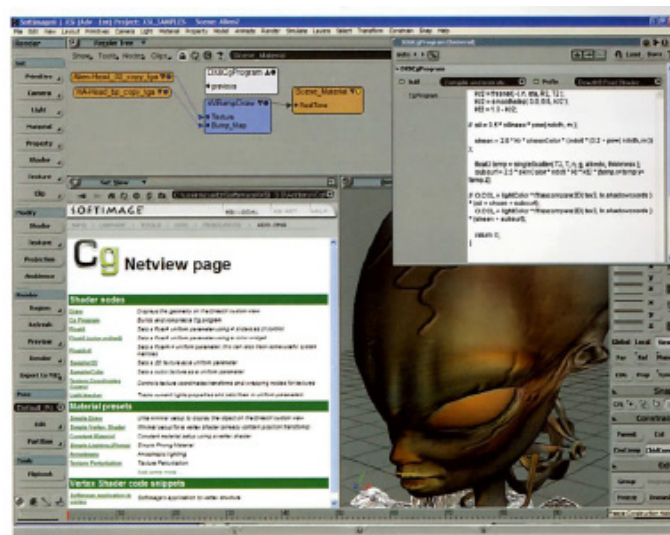
3. APIs and Programming GPUs

As adoption of GPGPU spread, so did systems for interacting with and managing the complex, parallel processes executing on the GPU. Graphics hardware manufacturers developed improved hardware and software for programmable GPUs, including additional functionality for existing graphics application programming interfaces (APIs). Ex1003, ¶¶55–60. Graphics APIs act as a bridge between software applications and the graphics hardware by providing high-level, simple-to-use functions for developers while handling low-level communication with the hardware. As the '461 patent acknowledges, graphics APIs, like DirectX (created by Microsoft) and OpenGL (created by Silicon Graphics, Inc.), and high-level shading languages like HLSL (High-Level Shader Language) and Cg (C for Graphics) were widely used. *Id.*; Ex1001, 2:20–22. Visual tools were also created to allow users to easily develop and interact with GPGPU processes. Ex1003, ¶¶55–60. For example, GUIs allowing a user to perform functions including selecting inputs to be processed, interactively adjusting parameters used by the GPU for a simulation, and changing the appearance of outputs were widely used with popular graphics APIs.



Color Plate 1 Screen shot of the RenderMonkey IDE user interface. The shader that is procedurally generating a Julia set on the surface of the elephant is shown in the source code window. Color selection tools and user interface elements for manipulating user-defined uniform variables are also shown.

Ex1016, 27.



Ex1017, 21.

B. Alleged Invention of the '461 Patent

In the Background, the '461 patent acknowledges that “manufacturers of GPUs have included general purpose programmability into the GPU architecture leading to the increased popularity of using GPUs for highly parallelizable and computationally expensive algorithms outside of the computer graphics domain.”

Ex1001, 1:38–50. However, the patent asserted that “general purpose GPU (GPGPU) applications are not able to achieve optimal performance” because “[t]here is overhead for graphics-related features and algorithms that are not necessary for these non-video applications.” *Id.*, 1:50–55.

To address these alleged drawbacks, the patent describes an “accelerator” (Figure 2, below) having at least “one or more graphics processing units” (GPU 240), “two or more associated memory banks that are logically or physically partitioned” (shader memory bank 210 and texture memory bank 250), and “a specialized controller” (controller 220). *Id.*, 2:34–41. This arrangement allegedly improves performance because, rather than the system CPU, “[t]he controller handles most of the primitive operations needed to set up and control GPU computation.” *Id.*, 2:41–44. As a result, “the CPU is freed from this function and is dedicated to other tasks.” *Id.*

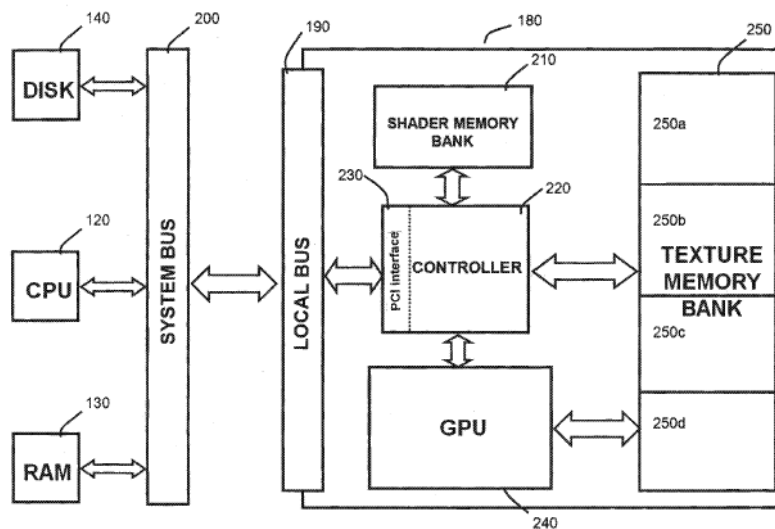


FIG. 2

During reissue, however, the '461 claims patent moved away from this alleged novelty. Instead, the reissued claims focus on maintaining separate “execution streams,” implemented as “separate threads,” and managing interactions between the execution streams. *Id.*, 7:60–9:22. The claims recite a CPU that executes a “user interaction stream” for processing user interactions; another processing unit that executes a “computational stream” for processing computations representing an artificial neural network; and a data exchange between the streams. *Id.*, 2:11–52, 7:60–8:15. Once the processing unit begins executing computations, the computational stream has control over the data exchange. *Id.*, 8:35–9:22. When execution is complete or interrupted, control shifts from the computational stream to the user interaction stream. *Id.* For example, when no simulation is running, the user can set and edit parameters and elements of the artificial neural network, change input sources, and edit display or storage parameters. *Id.*, 8:50–52.

The claims further require that a user command received while computations are executing is queued for execution at a time determined by the computational stream. *Id.*, 8:35–9:22. Although the user “retains the ability to interrupt the simulation, change the input, or change the display properties of the framework,” these “interactions are queued to be performed at times determined” by the computational stream to “avoid the corruption of data.” *Id.* These features were well known and widely used in the GPGPU field before the '461 patent. Ex1003.

C. Prosecution History of the '461 Patent

The '461 patent originated from reissue Application No. 17/136,343 (“the '343 application”) filed December 29, 2020, seeking reissue of U.S. Patent No. 9,189,828. Ex1002, 1–22.

The reissue application cancelled all claims of the '828 patent, introducing new claims. *Id.*, 289–298. Eventually, Applicant amended Claim 21 by (1) replacing the language “shifting priority” with “shifting control of data exchange between the user interaction stream and the computational stream,” and (2) adding “queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network.” *Id.*, 294. Applicant also amended Claim 33, adding “a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption.” *Id.*

The Examiner allowed the pending 21 claims on November 8, 2022 and stated that the prior art did not “teach or fairly suggest” the newly amended claimed combination of limitations. *Id.*, 359–67.

D. Person of Ordinary Skill in the Art

The '461 patent relates to the field of computer systems and architecture, including the use of GPUs. Ex1001, 1:38–55. A POSITA in the field around 2005–

2006 would have had a Bachelor's degree in electrical or computer engineering (or equivalent discipline) and at least two years' experience in research, design and/or development of computer systems, including experience with GPUs. Ex1003, ¶¶15–20. Additional education could substitute for experience, and vice versa. *Id.*

IV. GROUNDS OF UNPATENTABILITY

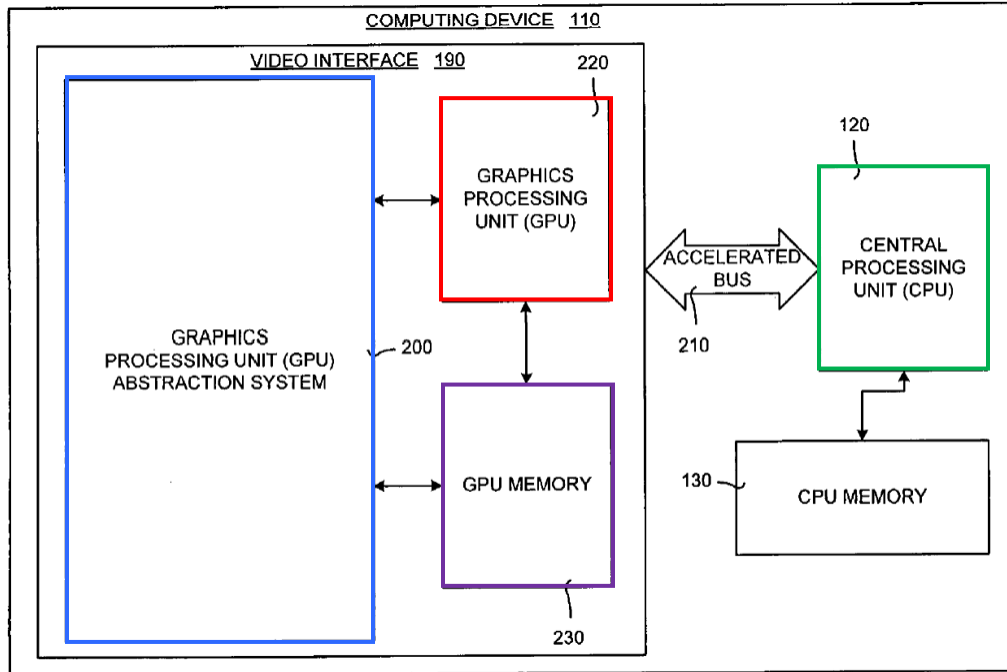
A. Grounds 1–6

1. Overview of Prior Art

a. Buck

Buck explains that it was well known to use GPUs for non-graphics processing, like simulating artificial neural networks, due to their “increased processing power and programmability.” Ex1004, [0034]. Buck discloses a computer system that uses a GPU to process and execute “operations traditionally handled by the CPU,” such as simulating “layers in a neural network.” *Id.*, [0011], [0029]–[0031], [0054]. Buck's optimized GPU-accelerated system “take[s] full advantage of the new flexibility introduced by programmable pixel shaders of GPUs” and “parallel architecture of the GPU” to train, test, and use neural networks. *Id.*

Buck's computing device 110 includes CPU 120 (green), GPU 220 (red), “GPU optimization system 200” (blue), and GPU memory 230 (purple). *Id.*, [0056]–[0057].

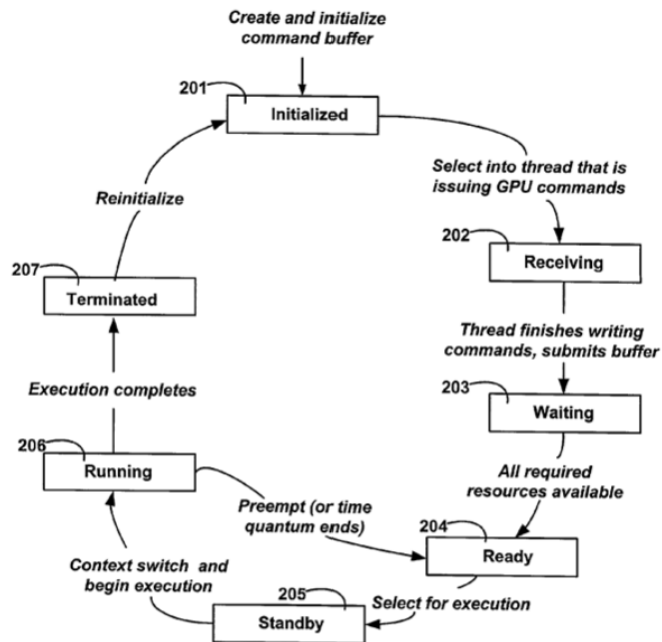


Id., Fig. 2 (annotated); Ex1003, ¶¶114–19.

b. Wilt

Wilt discloses “efficient and effective management of computational resources of graphics processor(s)” in a computing system having a host processor CPU and a co-processor GPU. Ex1005, [0009]. Wilt’s GPU is controlled by command buffers sent by its host CPU. *Id.*, [0095]. Wilt discloses a system for constructing multiple hardware-specific commands, or “command buffers,” in parallel and scheduling and queuing the command buffers for dispatch to the GPU through the use of a scheduler. *Id.*, [0010], [0107], [0126].

FIG. 2



Id., Fig. 2.

The scheduler queues command buffers for execution according to resource availability, command buffer priority, and real-time requirements of the system. *Id.* Wilt's system "enables multiple threads of execution" to "efficiently share the computational resources available in the system." *Id.* For example, "a process, or a thread in a process, that implements some specific graphical application" in Wilt's computer system can use an API call to request an action from the GPU. *Id.*, [0098]. That API call can be translated into a hardware-specific command written into a command buffer. *Id.*, [0097]–[0105]. Once the command buffer is full or the command has been completely written into the command buffer, the command buffer enters a waiting state until its waiting criteria are fulfilled, at which time it

can be submitted to the GPU. *Id.*, [0106]–[0107]. Wilt’s scheduler “has a great deal of flexibility in deciding which command buffer should be submitted to the hardware,” and can consider “the priority, if any, of the command buffer, the priority of the requesting thread, the real-time requirements of the system, and so on.” *Id.*

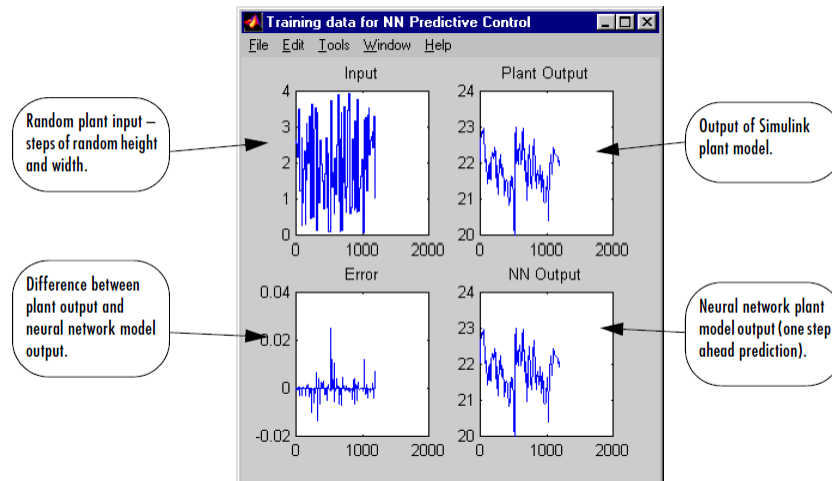
To prevent the GPU from “wast[ing] valuable time waiting for another command buffer to be submitted,” Wilt’s scheduling and resource management system queues the next command buffer for dispatch while the GPU is executing a command buffer. *Id.*, [0126]. When the command buffer is ready for execution, the scheduler “submits the command buffer to the hardware.” *Id.*, [0108]. The command buffer remains in the running state until it is preempted or its entire contents have been read and executed. *Id.*; Ex1003, ¶¶120–26.

c. nnet

nnet is a user manual for MathWorks’ Neural Network Toolbox, a collection of functions for creating and interacting with artificial neural networks in MATLAB. Ex1006. MATLAB is a well-known, widely used programming language and platform that “integrates computation, visualization, and programming in an easy-to-use environment.” Ex1031, 1–2. Typical uses for MATLAB include modeling, simulation, and scientific and engineering graphics. *Id.*

In addition to describing the functions available for implementing neural networks, nnet also explains how to use its GUI to perform standard user interface

functions, such as “creat[ing] a network, view[ing] it, train[ing] it, simulat[ing] it and export[ing] the final results,” and “import[ing]...for use in the GUI.” Ex1006, 91–100, 237–244.



Id., 218; Ex1003, ¶¶127–33.

d. ANN

ANN explores the application of GPUs for artificial neural network computation. Ex1007. ANN teaches that artificial neural networks are “widely used in pattern recognition,” and are particularly beneficial where “computational load is very heavy” or where “real time process[ing] is required.” *Id.*, 1. ANN explains the significant potential for using GPUs for artificial neural networks due to their inherently parallel and repetitive nature. *Id.*

In one example, ANN applies these principles by implementing a “[t]hree layer MLP neural network” to recognize a soccer ball (object recognition) and “trace the ball in real time” (feature tracking). *Id.*, 2–5; Ex1003, ¶¶134–38. Video data of

the soccer ball gathered in real-time would be transferred by the CPU to GPU memory for processing, and in other implementations, the GPU may have a “video in” function such that video data is “retrieved directly from [the] camera and store[d]” in GPU memory. Ex1007, 5.

e. GPU Gems

GPU Gems consists of 48 chapters describing the state of the art in GPU programming. Ex1008, xxxi–xxxii. Part IV is a “Primer” on “General-Purpose Computation on GPUs,” which “aims to provide a gentle introduction to the world of general-purpose computation on graphics processing units, or ‘GPGPU.’” *Id.*, xx, 453.

Chapter 40—titled “Computer Vision on the GPU”—explores “methods of efficiently mapping common mathematical operations of computer vision onto modern computer graphics architecture.” *Id.*, 649. GPU Gems explicitly discloses GPU programming techniques that require receiving and processing “real-time” input from a video camera. *Id.*, 649–66.

Chapter 40 explains that the “single-instruction, multiple-data (SIMD) capability of the GPU makes it suitable for running computer vision tasks, which often involve similar calculations operating on an entire image.” *Id.*, 649–50. Thus, GPUs can be used to “free up the CPU for other tasks.” *Id.*, 650. The computer vision algorithms discussed can be used to “create a fully interactive, real-time

computer-mediated reality, in which the environment of a user appears to be augmented, diminished, or otherwise altered.” *Id.*, 651. All methods discussed in the chapter were publicly available in “the OpenVIDIA GPU computer vision library.” *Id.*, 650–51.

One example of an application GPU Gems discusses is “feature tracking.” *Id.*, 650–51. Chapter 40 explains how “[t]he GPU is used to detect and track distinctive points (features) in real time as the camera pans,” *id.*, 661, and includes shader program code for implementing feature tracking, *id.*, 663. Ex1003, ¶¶139–45.

2. Combinations

a. Combination of Buck and Wilt (All Grounds)

Before the ’461 patent, it was well known that resource and task management methods could be implemented in GPU-accelerated systems to optimize efficiency, balance computational loads, and manage data dependencies. Section III.A.2. These concepts had already been applied by companies in the industry, including Microsoft, which is the assignee on both Buck and Wilt. Ex1003, ¶¶31–85, 146–58. A POSITA would have been motivated to implement Wilt’s scheduling and task management system in Buck’s GPU-accelerated system for simulating artificial neural networks to optimize the performance of the system. *Id.*, ¶¶149–58.

Buck discloses a computer system that uses GPU-acceleration for artificial neural networks. Ex1004, [0011], [0029]–[0031], [0054]. While Buck does not

focus on how its system schedules and manages tasks on the GPU, Wilt does. In particular, Wilt describes managing the computational resources of a GPU to facilitate multitasking and the execution of multiple instruction streams. Ex1005, [0009]–[0010], [0107], [0126]. A POSITA would have been motivated to implement Wilt’s resource management method within Buck’s co-processor system and would have reasonably expected that the combination would enable multiple tasks to “efficiently share the computational resources available in the system,” as taught in Wilt. Ex1003, ¶¶149–58; Ex1005, [0010].

A POSITA reviewing Buck’s CPU and optimized GPU computing system would have looked to other art describing the resource management operations of GPUs and CPUs. Ex1003, ¶¶149–58. It was common for a patentee to focus on describing one set of features in one patent application and another set in a different patent application. Therefore, another Microsoft patent application publication, such as Wilt, would have been a natural reference for a POSITA to consult. Both Buck and Wilt are Microsoft patent application publications; both focus on a system having a host CPU and a co-processor GPU; and Buck’s and Wilt’s example computing environment are nearly identical. *Compare* Ex1004, Fig. 1, *with* Ex1005, Fig. 1B. Whereas Buck focuses on the use of the GPU for complex computations instead of the CPU, Wilt supplies the behind-the-scenes details of managing the computational resources of the CPU and the GPU. Ex1003, ¶¶149–58.

In computationally intensive applications, such as simulating an artificial neural network, it would have been important to “careful[ly] manage[]” co-processor computational resources. Ex1005, [0007]; Ex1003, ¶¶149–58. Wilt explains that, as GPUs “become more functional,” the number of applications that demand “significant computational resources” from GPUs increase. Ex1005, [0008]. Wilt’s system for “the scheduling and dispatch of graphics processing requests,” addresses “the need for efficient and effective management of computational resources of graphics processor(s) in a computing system.” *Id.*, [0009]. Buck’s computing system is specifically designed to perform the “more sophisticated computations” that Wilt contemplates a GPU performing, and a POSITA would have looked to the art for solutions to efficiently manage the complex computing tasks delegated to Buck’s GPU. Ex1004, [0010]–[0012]; Ex1005, [0005]; Ex1003, ¶¶149–58.

A POSITA would have had a reasonable expectation of success implementing Wilt’s teachings in Buck’s system. Ex1003, ¶¶149–58. Wilt’s system efficiently distributes a GPU’s computational resources and accommodates various synchronization requirements in co-processor system; and Buck’s optimized GPU-accelerated system is such a co-processor system having complex synchronization and resource management requirements. Thus, a POSITA would have been motivated, and would have understood how, to incorporate Wilt’s system into Buck’s optimized GPU-accelerated system. *Id.* A POSITA would have understood

that Wilt's teachings would be applicable to Buck's GPU-accelerated system and would have provided benefits such as optimal resource allocation. *Id.*

Ultimately, a POSITA would have understood that Buck/Wilt would have been an implementation of a known technique (*e.g.*, Wilt's resource and task management) within a known system (*e.g.*, Buck's optimized GPU-accelerated system) to yield predictable results (*e.g.*, fast and efficient simulation of artificial neural networks). *Id.*

b. Combination of nnet with Buck and Wilt (Ground 2)

User interfaces, such as GUIs, were widely used before the '461 patent, including for designing the simulation of artificial neural networks. Section III.A.3. nnet describes a "simple and user friendly" user interface through which users can create, edit, train, and simulate artificial neural networks. Ex1006, 31. A POSITA would have been motivated to implement nnet's interactive GUI for simulating artificial neural networks in the Buck/Wilt system. Ex1003, ¶¶311–18. nnet's GUI would allow users to interact with Buck/Wilt's artificial neural network in an accessible way and input information and commands that are processed by the CPU, managed and scheduled by the scheduler, and executed by the GPU while providing visual feedback in response to user-indicated changes and display the results of the computations so that users can review, analyze, and use the results of the computations. Ex1006, 91.

Command line interfaces and GUIs had been ubiquitous in the field for decades by the time of the '461 patent, and nnet's discussion of both discloses just two examples of the many simple user interfaces available that provided standard interface functions. Ex1003, ¶¶31–85, 312–18. nnet's user interfaces are similar to those used with graphics APIs, such as OpenGL and DirectX, that the '461 patent explicitly characterizes as prior art. Ex1001, 2:20–22; Ex1003, ¶¶31–85, 312–18. Given the common nature of such user interfaces and functions, a POSITA would have had a reasonable expectation of success of implementing a known feature (nnet's GUI) into a known system (Buck/Wilt's GPU-accelerated system) to yield predictable results (*e.g.*, allowing a user to easily provide input). Ex1003, ¶¶312–18.

c. Combination of ANN (Ground 3) or GPU Gems (Ground 4) with Buck and Wilt

Computationally intensive and repetitive, computer vision algorithms were a common early application for GPGPU computing. Section III.A.1; Ex1008, 649–66. Several “fully interactive,” “real-time computer vision programs” for execution by a GPU were publicly available prior to the '461 patent. *Id.*; Section III.A.1; Ex1003, ¶¶31–85, 353–360. ANN and GPU Gems provide examples of such interactive, real-time computer vision programs are implemented on GPUs. A POSITA would have been motivated to implement these programs on GPU-

accelerated parallel processing systems that are optimized for these kinds of applications, such as that of Buck/Wilt. Ex1003, ¶¶353–60.

i. ANN (Ground 3)

A POSITA seeking to execute the multi-layer artificial neural network for real-time object tracking disclosed in ANN would have sought out advanced systems with a programmable GPU, such as that in Buck/Wilt. Ex1003, ¶¶354–355. To achieve efficiency in computation, a POSITA would have been motivated to implement ANN’s multi-layer artificial neural network on state-of-the-art GPU-accelerated parallel processing systems, such as the Buck/Wilt system. Ex1007, 3; Ex1003, ¶355. A POSITA would have understood that the GPU-accelerated system for implementing neural networks disclosed by Buck/Wilt would be an ideal system for the artificial neural network underlying ANN’s computer vision algorithm. Ex1003, ¶355. The Buck/Wilt system is specifically designed to accelerate the parallel computations required for simulating artificial neural networks. Buck emphasizes that its system “exploit[s] the parallelism and high-memory bandwidth of GPUs” as well as the “flexibility introduced by programmable pixel shaders” to implement machine learning algorithms, such as training and testing artificial neural networks. Ex1004, [0010]–[0011], [0055].

ii. GPU Gems (Ground 4)

A POSITA would also have been motivated to execute GPU Gems’ real-time computer vision programs on the system of Buck/Wilt. Ex1003, ¶¶354, 356. GPU Gems describes receiving and processing “real-time” input from a video camera to perform feature tracking. Ex1008, 650–51. A POSITA reviewing GPU Gems’ discussion of using GPUs “to accelerate computer vision computation and free up the CPU for other tasks,” would have been motivated to implement GPU Gems’ feature-tracking algorithm on state-of-the-art GPU-accelerated parallel processing systems and that would have led to the Buck/Wilt system. *Id.*, 650; Ex1003, ¶356. A POSITA would have understood that Buck/Wilt’s optimized system for implementing “general matrix operations” involving “large amounts of data and processing” would be an ideal system to accelerate GPU Gems’ computer vision program. Ex1003, ¶356. The Buck/Wilt system is specifically designed to accelerate the parallel computations required for simulating artificial neural networks. Buck emphasizes that its system “exploit[s] the parallelism and high-memory bandwidth of GPUs” as well as “flexibility introduced by programmable pixel shaders” to implement computationally intensive non-graphics applications. Ex1004, [0010]–[0011], [0055].

iii. Implementing the Computer Vision Method of ANN or GPU Gems in Buck/Wilt's System

The computer vision techniques discussed by ANN or GPU Gems are two examples of well-known GPGPU computing applications that could be performed on Buck/Wilt's GPU-accelerated system. Ex1003, ¶¶357–60. Computer vision tasks often “exceed[ed] the real-time capabilities of the CPU,” but “map[p]ed efficiently onto the modern GPU,” and had been performed on GPUs years prior to the '461 patent. Ex1008, 649–66. Given the number of publicly available computer vision algorithms that were already being implemented on GPU-accelerated systems, a POSITA would have readily understood how to perform ANN or GPU Gems' techniques on Buck/Wilt's system. *Id.*, 649–66. A POSITA reading ANN or GPU Gems would have understood not only that the object recognition method disclosed in either reference could be performed on a GPU (such as the optimized GPU of Buck/Wilt), but also how to use Buck/Wilt's GPU-accelerated system to implement it. Ex1003, ¶¶357–60. And a POSITA would have had a reasonable expectation of success in the combination because Buck/Wilt's system is configured to “exploit the parallelism and high-memory bandwidth of GPUs” and operate on “large floating point matrices and vectors” as contemplated in both of ANN and GPU Gems. Ex1004, [0010]–[0012]; Ex1007, Abstract, 1, 3; Ex1008, 649–66; Ex1003, ¶¶357–60.

A POSITA would have been motivated by ANN's discussion of receiving artificial neural network input data from a video camera and GPU Gems' discussion of receiving real-time data from a video camera to incorporate this source of input into the Buck/Wilt optimized GPU-accelerated system. Ex1003, ¶¶357–60. The Buck/Wilt system explicitly discloses receiving input from various external input devices, including devices that generate audio and/or video input data in real-time, through its user input interface connected to the CPU. Ex1004, [0049]–[0050]; Ex1005, [0071], [0107]. Thus, a POSITA would have found it obvious to add video camera input to Buck/Wilt's system as input to the artificial neural network as disclosed by ANN/GPU Gems and would have had a reasonable expectation of success in doing so. Ex1003, ¶¶357–60.

A POSITA would have understood that the combination of Buck/Wilt with ANN or GPU Gems would involve incorporating a known feature (receiving input from a video camera) into a known system (Buck/Wilt's GPU-accelerated system) ready for improvement to yield predictable results (receiving real-time video data to be processed by an artificial neural network implemented on a GPU). *Id.*, ¶¶353–60.

**d. Combination of nnet with Buck, Wilt, and ANN (Ground 5)
or Buck, Wilt, and GPU Gems (Ground 6)**

For the same reasons a POSITA would have applied nnet's GUI in the Buck/Wilt system (Section IV.A.2.b), a POSITA would have been motivated to use

nnet's GUI when ANN's or GPU Gems' computer vision algorithms are implemented on Buck/Wilt. Ex1003, ¶¶311–18, 397–402. GUIs were widely used with GPU-accelerated systems to allow users to interact with and control the systems well before the '461 patent. Section III.A; Ex1003, ¶¶31–85. GPU-accelerated systems for applying artificial neural networks to perform real-time detecting and tracking of an object in a video at the time were “fully interactive.” Section III.A; Ex1008, 651. nnet's GUI is a “simple and user friendly” way to create, edit, train, and simulate artificial neural networks for these applications. Ex1003, ¶¶397–402. Doing so was well within the skill of a POSITA and would have provided a known feature (nnet's GUI) into a known system (Buck/Wilt's GPU-accelerated system) that yields predictable results (allowing a user to easily provide input) and the POSITA would have had a reasonable expectation of success. *Id.*

3. Detailed Analysis of Grounds

Claim 21 and its dependents, Claims 22–32 and 38–41 are analyzed first, followed by analysis of Claims 33–37, which recite similar concepts to Claim 21 with the addition of trivial features and steps, all of which are disclosed in the prior art raised in this Petition.

For each element, the analysis describes how the limitation is disclosed by Buck/Wilt, which applies to all grounds. Where noted in headings, the analysis

additionally shows how the limitation is disclosed in one of the supplemental references and to which grounds that analysis applies. Ex1003, ¶¶146–402.

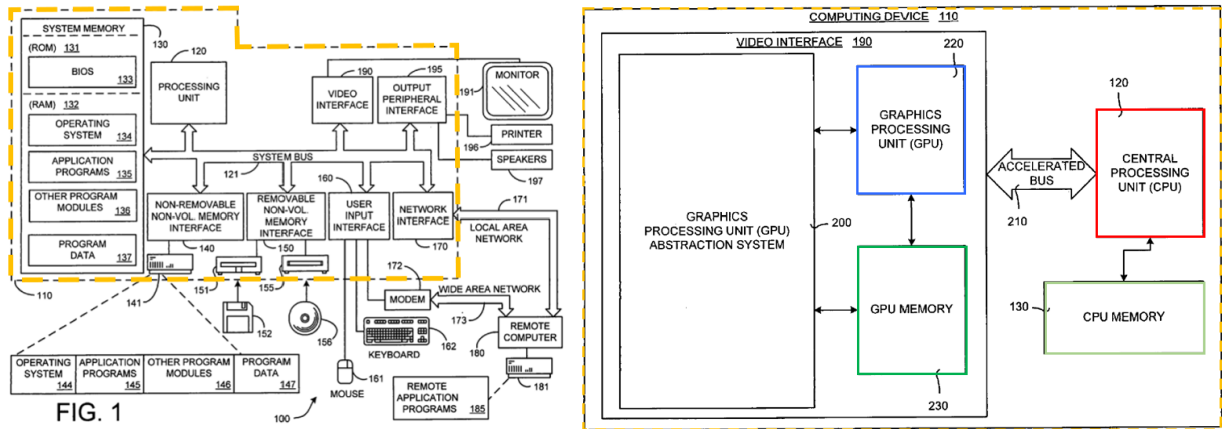
a. Claim 21

i. 21[pre]

To the extent the preamble is limiting, Buck/Wilt discloses it. Ex1003, ¶¶160–64.

Buck discloses “systems and methods” for “processing and execution of general matrix operations” such as “neural network[s].” Ex1004, Abstract, [0029]–[0031], [0055], [0076]–[0083], [0129]–[0132], [0157]–[0158], [0210]–[0250], Figs. 14–16, cls. 22, 26, 28. Buck walks through working examples of “a training session” and “a testing session” of a “pair of layers in a neural network.” *Id.*, [0029]–[0031], [0229]–[0250]. Thus, Buck discloses a method of executing computations representing an artificial neural network as claimed. Ex1003, ¶161.

Buck’s computer system (computing environment 100) includes a computer 110 (orange) having a central processing unit (CPU 120, red), a processing unit (GPU 220, blue), a CPU memory 130 (light green), and a GPU memory 230 (green). *Id.*, [0037]–[0057].



Ex1004, Figs. 1, 2 (annotated).

Buck's CPU memory teaches at least a "first memory partition." Ex1003, ¶163. Buck's segmented GPU memory additionally teaches at least a "first memory partition" and a "second memory partition." *Id.*; Ex1001, 5:65–6:11 (explaining that "partitioning" can "be done logically," and that different portions of a single memory bank can be used to store different types of data, such as "input data, output data, intermediate results, and parameters"). Buck's GPU memory 230 includes a workspace and "read only buffers (segments)" outside of the workspace that "may be used as input data." Ex1004, [0123], Fig. 9. The workspace includes scratch space having a "temporary buffer" (*i.e.*, partition) and "a permanent buffer" (*i.e.*, partition) and contains "all operands as well as a destination for computation." *Id.*, [0070], [0124]. GPU memory 230 stores various types of data in buffers (partitions), including shader "programs P," "initial weights," "learning parameters," "input

data,” “target data,” “intermediate results,” and “trainable parameters.” *Id.*, [0122], [0235]–[0236], Figs. 9, 15, 16.

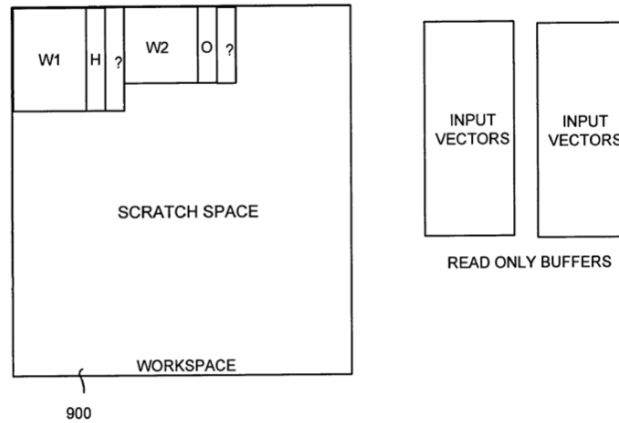


FIG. 9

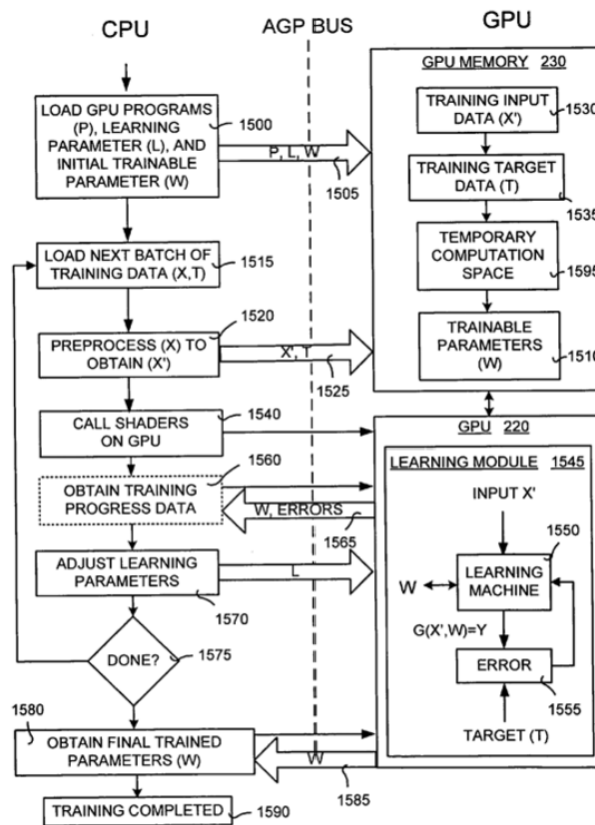


FIG. 15

Id.

ii. 21[a]

Buck/Wilt discloses this limitation. Ex1003, ¶¶165–75.

A user “may enter commands and information into” Buck’s computer 110 “through input devices such as a keyboard 162, and pointing device 161.” Ex1004, [0049]–[0050]. Such devices are connected to Buck’s CPU (processing unit 120) “through a user input interface 160 that is coupled to the system bus 121” or “by other interface and bus structures,” including “a parallel port, game port[,] or a universal serial bus (USB).” *Id.* Other input devices, including a microphone, a television or broadcast video receiver, “or the like” can be connected to and communicate with the CPU through the user input interface 160 as well. *Id.*

Buck’s CPU processes these user interactions and commands by executing a user interaction stream. Ex1003, ¶167. The CPU maintains its own “stream[s]” of instructions and data. Ex1004, [0008]. As the “host” processor, Buck’s CPU receives and interprets “commands and information” from user interactions through input interface 160. *Id.*; Ex1003, ¶¶167–68. The CPU stores these commands and information in its CPU memory 130 (a first memory partition). Ex1004, [0046]. CPU memory 130 includes RAM 132, which “typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by” CPU 120. *Id.*

A POSITA would have understood that Buck's interactive system for simulating artificial neural networks would receive a variety of inputs to the artificial neural network, including commands and information, specifying how the artificial neural network should process information and what information to process. Ex1003, ¶¶169–70; Section III.A.3. For example, commands include instructions defining, setting, or editing functions (such as activation, learning, and training functions of the artificial neural network), parameters, or variables of an artificial neural network. Ex1003, ¶170. Information includes the memory locations from which data should be loaded, user-defined arrays of data, or live streams of audio, video, or text data. *Id.*

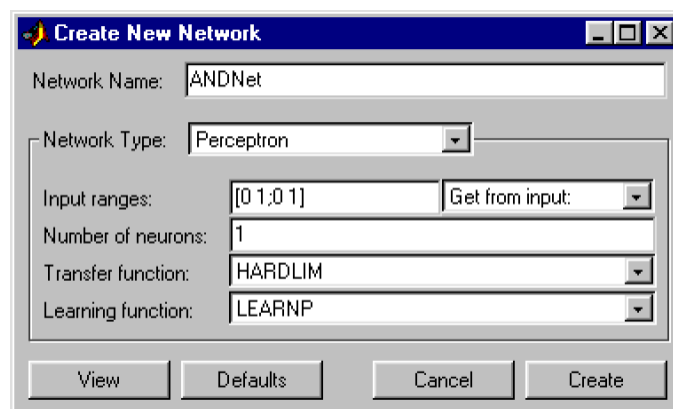
Buck's CPU prepares instructions to be executed by the GPU as shader programs and data to be used in operations performed by the GPU as texture data. Ex1004, [0010]–[0012], [0055], [0058]–[0068], [0069]–[0210], [0211]–[0250], Figs. 9, 15, 16; Ex1003, ¶171. The CPU then transfers inputs to the artificial network to its segmented GPU memory, storing some input data in a texture in one segment, “a first memory partition,” and other input data in a texture in another segment, or “a second memory partition.” Ex1004, [0106]–[0120], [0235]–[0250]; Ex1003, ¶172. For example, one memory partition may contain “initial weights” and/or “learning parameters,” while a second memory partition may contain other input data such as “trainable parameters.” Ex1004, [0106]–[0120], [0235]–[0250].

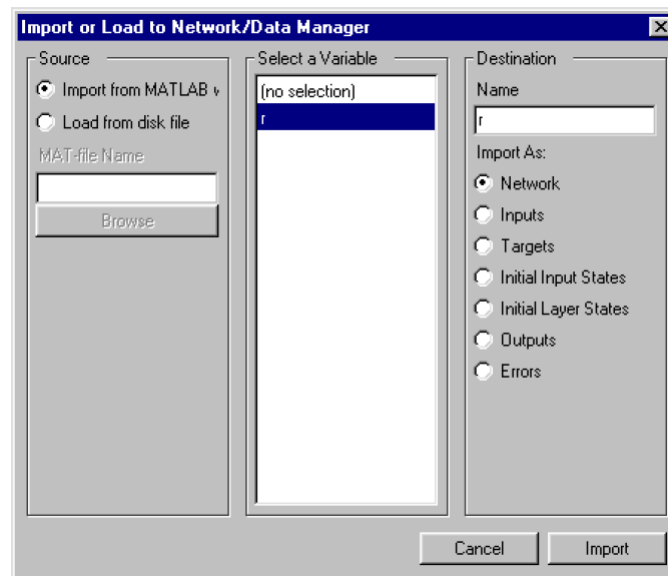
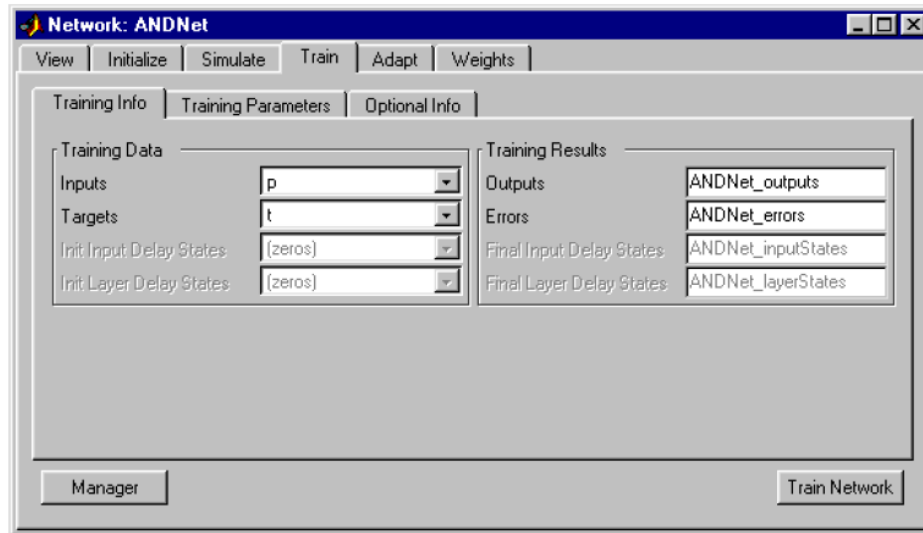
Additionally, the input shaders (programs) are transferred to a different memory partition, which can also be the claimed “second memory partition.” *Id.*

Because the CPU’s processing commands and information causes inputs to the artificial neural network to be transferred to the CPU memory and/or segments of GPU memory, Buck discloses that the user interaction stream controls transfer of inputs to the artificial neural network to the first memory partition and the second memory partition. Ex1003, ¶¶165–75.

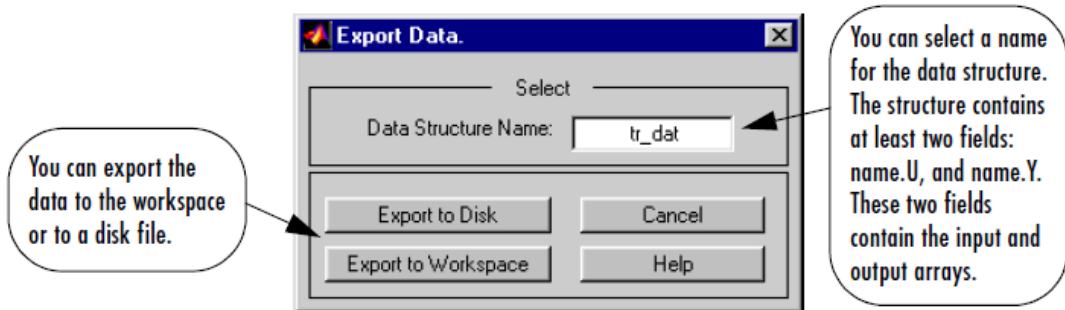
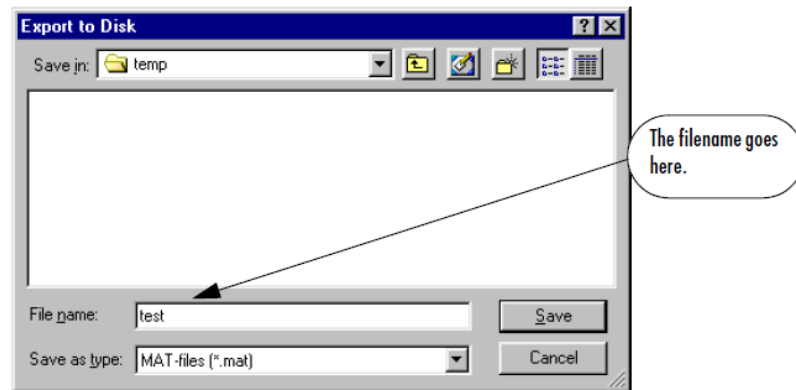
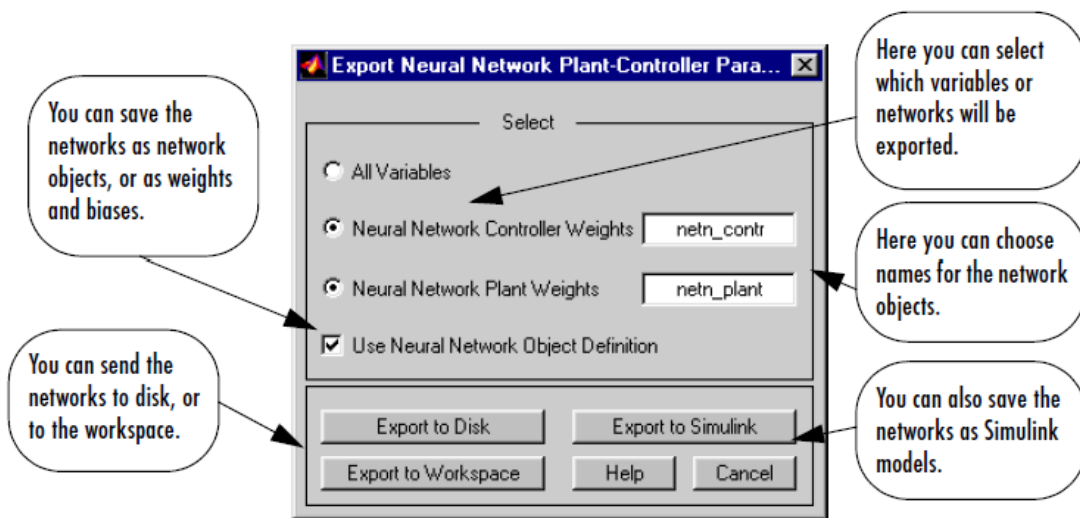
(1) nnet (Grounds 2, 5, 6)

nnet discloses an interactive GUI through which a user can define, set, or edit functions, parameters, variables, and sources of inputs to an artificial neural network, such as commands and data. Ex1006, 92–100, 392–413. The GUI also allows users to export, save, display, or otherwise manipulate outputs of the artificial neural network as well. *Id.*, 97–98, 237–243.





Ex1006, 94, 95, 99.



Id., 237, 238, 241; Ex1003, ¶¶319–20.

iii. 21[b]

Buck/Wilt discloses this limitation. Ex1003, ¶¶176–86.

Buck discloses a processing unit executing a computational stream including computations representing an artificial neural network. Ex1004, [0008]; Ex1003, ¶177. Buck's GPU (a "processing unit") is an autonomous processor with its "own instruction stream[], datapaths, and dedicated memory" separate from those of the CPU. Ex1004, [0008]. Buck's "GPU optimization system and method" ports "some or all of the CPU processing" operations "well-suited for processing on the GPU," to the GPU. *Id.*, [0035]. These operations include computations to train and test a two-layer artificial neural network. *Id.*, [0211]–[0250].

Buck discloses that the computational stream controls the data exchange between the user interaction stream and the computational stream. Ex1003, ¶179. As in the '461 patent, Buck's computational stream "interacts with the User Interaction Stream...through initialization routines and data exchange in between simulations." Ex1001, 7:60–9:52; Ex1003, ¶179. "After the user starts the simulation," although Buck's CPU can continue to receive and process information and commands for the GPU while the GPU executes computations representing an artificial neural network, the GPU does not receive any information or commands until "until the simulation is finished or interrupted" and thus Buck's computational stream controls data exchange in the way described in the '461 patent. Ex1001, 8:14–9:52; Ex1004, [0008], [0055], [0106]–[0120], [0235], [0211]–[0250]; Ex1003, ¶179. In the '461 patent, the "user can perform operations" to direct the CPU to

parse new input textures for the GPU or convert computational elements to GPU programs “as many times as necessary prior to starting the simulation,” but once the simulation has been initialized, the user’s commands “to interrupt the simulation, change the input, or to change the display properties...are queued to be performed at times determined by the controller-driven exchange.” Ex1001, 8:14–9:52. Similarly, once Buck’s GPU begins executing a simulation of the artificial neural network, Buck’s CPU (or a particular thread of the CPU interacting with the GPU) “blocks,” and remains in a state of suspension until the GPU completes executing the simulation. Ex1004, [0235], [0211]–[0250]; Ex1003, ¶180. Buck’s blocked CPU, thus, is prevented from providing instructions or data to the GPU, perform synchronization with the GPU, or otherwise exchange data with the GPU to affect the computations representing the simulation of the artificial neural network (the computational stream). Ex1003, ¶¶180–81. In other words, when Buck’s GPU begins computations, the computational stream controls data exchange data exchange between the user interaction stream executed by the CPU and the computational stream executed by the GPU while the computations are executed by causing suspension of Buck’s CPU. *Id.*

Wilt also discloses a GPU that executes a stream of hardware-specific commands that together constitute a “computational stream” that performs requested computations. Ex1005, [0028], [0046], [0095]. As explained for 21[pre], in the

Buck/Wilt system, these computations represent an artificial neural network. Ex1004, [0055], [0230], [0233]–[0250]. Wilt’s CPU translates high-level commands from an application to perform a graphics operation into a lower-level “stream of hardware-specific” commands “ready for consumption by graphics hardware,” or a “command buffer.” Ex1005, [0028], [0046]. The CPU sends these “command buffers” to the GPU for execution. *Id.*, [0095].

Wilt explains that a “graphics processor can only execute one command buffer at a time.” Ex1005, [0086], [0108]. Once the GPU begins executing a command buffer, “the command buffer remains in the running state...until some event preempts execution of the command buffer or the entire contents of the command buffer have been read and executed.” *Id.* While Wilt’s GPU is executing a command buffer (a portion of the computational stream), the GPU does not execute any other command buffer and thus cannot change the instructions it executes or the data on which it operates while it is executing a command buffer. Ex1003, ¶¶182–84. Like the “computational stream” in the ’461 patent, Wilt’s command buffer controls whether data exchange takes place between the user interaction stream executed by the CPU and the computational stream executed by the GPU in the same way the ’461 patent describes that feature in the claims. *Id.*; Ex1001, 8:14–9:52.

Wilt’s discussion of preemption additionally teaches that its computational stream controls whether data exchange takes places. Ex1005, [0108], [0127].

Preemption is the reclaiming of a resource from a process before it has finished using the resource. Ex1003, ¶¶185–86. When a currently executing task is preempted, it is interrupted so that another task, typically with a higher priority, can be executed. *Id.* The priorities of various command buffers, which are portions of the computational stream, are attributes of the computational stream. *Id.* Based on these attributes of the computational stream, Wilt’s GPU may halt execution of the preempted command buffer to execute the preempting command buffer and change the instructions it executes or the data on which it operates. *Id.* In this additional way, Wilt teaches that the computational stream controls whether data exchange takes place between the user interaction stream and the computational stream at least in the same way the ’461 patent describes that feature in the claims. *Id.*

iv. 21[c]

Buck/Wilt discloses this limitation. Ex1003, ¶¶187–90.

As explained for 21[a], Buck/Wilt discloses that its CPU executes a user interaction stream that can provide instructions and information to its GPU. Ex1003, ¶¶165–75, 187–88; Ex1004, [0049]–[0050], [0069]–[0075].

As explained for 21[b], Buck/Wilt discloses that its GPU’s execution of computations (*i.e.*, simulating an artificial neural network) prevents the CPU from exchanging data with the GPU until the computations are complete or preempted, and therefore that the computational stream has control over data exchange between

the user interaction stream executing on the CPU and the computational stream executing on the GPU while the GPU is executing the computations. Ex1003, ¶¶176–86, 189; Ex1004, [0211]–[0250]; Ex1005, [0086], [0108].

This suspension of the data exchange between the user interaction stream and the computational stream once the GPU begins executing computations teaches that control of the data exchange shifts between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network. Ex1003, ¶190.

v. 21[d]

Buck/Wilt discloses this limitation. Ex1003, ¶¶191–95.

As explained for 21[b] and 21[c], Buck/Wilt discloses the computational stream has control over data exchange between the user interaction stream executing on the CPU and the computational stream executing on the GPU while the GPU is executing the computations. Ex1003, ¶¶176–92.

As explained for 21[b], the Buck/Wilt GPU continues executing a computation representing the artificial neural network until it is complete or interrupted. *Id.*; Ex1004, [0211]–[0250]; Ex1005, [0086], [0108].

Once execution of a command buffer is completed or preempted, Wilt’s system “release[s] all of the resources that were needed for the command buffer and unblocks pending command buffers that need those resources.” Ex1005, [0115],

[0151]. Similarly, Buck discloses that its CPU (or a particular thread of the CPU) blocks until the GPU completes the computation; when Buck's CPU unblocks, it can resume providing instructions or data to the GPU, performing synchronization with the GPU, or otherwise exchanging data with the GPU. Ex1004, [0235]. This resumption of the data exchange between the user interaction stream and the computational stream once the GPU's execution of the computations is complete or preempted teaches that control of the data exchange shifts between the user interaction stream and the computational stream back to the user interaction stream in response to starting execution of the computations representing the artificial neural network. Ex1003, ¶193.

Additionally, Wilt explains that the scheduler accounts for variables such as the "priority" of the command buffer itself and of a requesting thread when scheduling tasks. Ex1005, [0107]. Priority scheduling was well-known at the time of the '461 patent, and it would have been known and obvious that different threads, such as threads for executing a computation or threads requiring data transfer could have different priority values. Ex1003, ¶¶31–85, 194. It would have been obvious that a new instruction to display the results of a computation received through the user input interface could be assigned a lower priority value than the execution of a computation for the neural networks, but that a new instruction to halt the computation received through the user input interface could be assigned a higher

priority value. *Id.*, ¶194. Thus, to the extent this limitation is not expressly disclosed, it would have been obvious in Buck/Wilt. *Id.*, ¶195.

vi. 21[e]

Buck/Wilt discloses this limitation. Ex1003, ¶¶196–201.

As explained for 21[a], Buck discloses that user commands received through its user input interface coupled to the CPU are received by the user interaction stream executed by the CPU. Ex1004, [0049]–[0050]; Ex1003, ¶¶165–75, 197.

When Wilt’s CPU receives a command for the GPU, it writes that command into a command buffer. Ex1005, [0097]–[0106]. Once the command buffer is full or the command has been completely written into the command buffer, the command buffer enters a waiting state until various waiting criteria are fulfilled. *Id.*

The “scheduler,” a component of Wilt’s operating system, then “decides which command buffer should be dispatched.” *Id.*, [0024], [0146]–[0154]. Wilt’s scheduler minimizes latency between the completion of one command buffer and the submission of another command buffer to the GPU by “queu[ing]” command buffers for execution “before the current command buffer has finished executing.” *Id.*, [0093], [0126]. The GPU can then “immediately begin execution of another command buffer when the current command buffer has completed.” *Id.* In particular, Wilt’s resource management system “maintain[s] multiple DMA buffers and ‘ping-pong[s]’ between them, writing to one while the graphics processor reads

and executes another.” *Id.*, [0085]–[0086], [0093]. Although the GPU can only execute one command buffer at a time, the CPU continues to prepare and queue new command buffers—based on commands received while the GPU is executing a different command buffer—to be executed later. *Id.*, [0086]; Ex1003, ¶¶196–201.

vii. 21[f]

Buck/Wilt discloses this limitation. Ex1003, ¶¶202–208.

As explained for 21[b] and 21[e], Buck/Wilt discloses the computational stream has control over data exchange between the user interaction stream executing on the CPU and the computational stream executing on the GPU while the GPU is executing the computations. Ex1003, ¶¶176–86, 196–203; Ex1004, [0035], [0211]–[0250]; Ex1005, [0086], [0108]. Wilt’s scheduler considers attributes of the computational stream, such as whether a command buffer (part of the computational stream executed by the GPU) is currently executing, its “priority, if any,” the priority of other command buffers, the “priority of the requesting thread,” and resources and data required to execute the command buffer are available, when “deciding which command buffer should be submitted to the hardware.” Ex1005, [0107].

Wilt’s scheduler then submits the command buffer for execution at a particular point within the computational stream. *Id.*, [0107]. Thus, in the Buck/Wilt system, a command buffer generated from a user command received while the GPU is executing computations would be submitted by the scheduler for execution at a

time determined by the computational stream because the scheduler's decision is based on the status and priority of the other command buffers. Ex1003, ¶204. Furthermore, because Buck/Wilt teaches that the command buffer can be scheduled for execution during the stream of computations that represent the artificial neural network, the combination discloses that the user command is executed during execution of the computations. *Id.*, ¶205.

If each currently executing command buffer, and not the set of all command buffers, is interpreted as a computational stream, as explained for 21[b], 21[d], and 21[e], the Buck/Wilt system discloses this limitation through its support of preemption. Ex1005, [0127]. If an event preempts the execution of the currently executing computation, the computation can be suspended and a new command buffer can be submitted to the GPU. Ex1005, [0108]–[0110], [0127], [0152]–[0153]; Ex1003, ¶206. After the preempting command buffer is executed, the execution of the preempted command buffer can then “be resumed where the preemption occurred.” Ex1005, [0110], [0152]–[0153].

In this additional, alternative way, Buck/Wilt teaches that the user command is executed during execution of the computations representing the artificial neural network at a time determined by the computational stream. Ex1003, ¶¶207–208. At a minimum, Buck/Wilt renders this limitation obvious as described above. *Id.*

b. Claim 22

Buck/Wilt discloses this limitation. Ex1003, ¶¶209–13.

As explained for Claim 21, the Buck/Wilt GPU's computational stream and the CPU's user interaction stream can exchange data. Ex1003, ¶¶160–210; Ex1004, [0211]–[0250]; Ex1005, [0095]–[0115], [0151].

Although Buck/Wilt's computational stream controls the data exchange while the GPU is executing computations representing the artificial neural network, when the GPU is not executing computations, it is idle and can receive different command buffers specifying different instructions and data on which it should operate. Ex1003, ¶211; Ex1004, [0211]–[0250]; Ex1005, [0086], [0108].

As explained for 21[d], upon completion or interruption of the GPU's execution of computations, control over the data exchange between the user interaction stream and the computational stream shifts to the user interaction stream. Ex1003, ¶¶191–95, 212. Specifically, the CPU's instruction stream related to the user interactions provided by the CPU through the user input interface (the user interaction stream) can resume providing instructions or data to the GPU, performing synchronization with the GPU, or otherwise exchanging data with the GPU when the GPU is not executing computations. Ex1004, [0211]–[0250]; Ex1005, [0086], [0108].

c. Claim 23

Buck/Wilt discloses this limitation. Ex1003, ¶¶214–18.

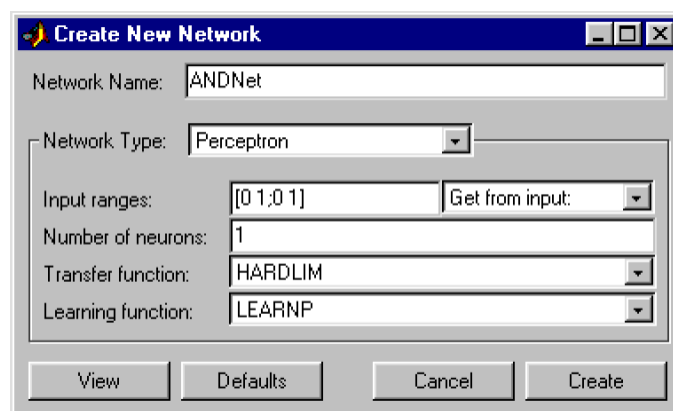
Buck and Wilt each disclose a user input interface through which a “user may enter commands and information into the computer.” Ex1004, [0049]–[0050]; Ex1005, [0071]. The “commands and information” that a user provides to Buck/Wilt’s computer allow a user to program the “programmable pixel shade[r]s” to “implement the various primitive operations” used in the artificial neural networks, define variables, implemented as textures, and to define neural networks to train or simulate using these shaders and textures. Ex1004, [0011]–[0012]; Ex1003, ¶215. The ’461 patent explains that “computational elements” refers to the system of equations defining the artificial neural network and includes “textures that track variables plus shaders that define processing algorithms.” Ex1001, 5:42–47, 8:55–58.

Thus, at a minimum, it would have been obvious in view of Buck/Wilt that a user can define and edit the shaders and textures used to “implement the various primitive operations used in” the computations representing an artificial neural network. Ex1004, [0011]–[0012]; Ex1003, ¶216. As the ’461 patent acknowledges, editing an artificial neural network “is a user-interactive process.” Ex1001, 8:63–65. Simple user interfaces for programming, such as command line interfaces and GUIs, were well-known and in-use for decades at the time of the ’461 patent. Buck’s

discussion of shaders that are “programmable” for a variety of applications, including running simulations, discloses that a user could program the shaders or update variables, stored as textures, as appropriate for their use case. Ex1004, [0007], [0011]–[0012], [0055]; Ex1003, ¶217. Because there was a finite number of known, predictable user interface options, including a GUI or command line interface, a POSITA would have found it obvious to implement an interface through which a user could create or edit the shaders and textures, and would have had a reasonable expectation of success in doing so. Ex1003, ¶¶214–18.

(1) nnet (Grounds 2, 5, 6)

nnet also discloses controlling setting and editing of computational elements. For example, nnet teaches that users can select from a number of predefined simulation, initialization, learning, and self-organizing map functions, or to create their own when creating neural networks. *Id.*, 392–413. Users can define their own functions using nnet’s command line interface (i.e., the “workspace”) and, as illustrated below, through its GUI.



Ex1006, 94; Ex1003, ¶¶321–23.

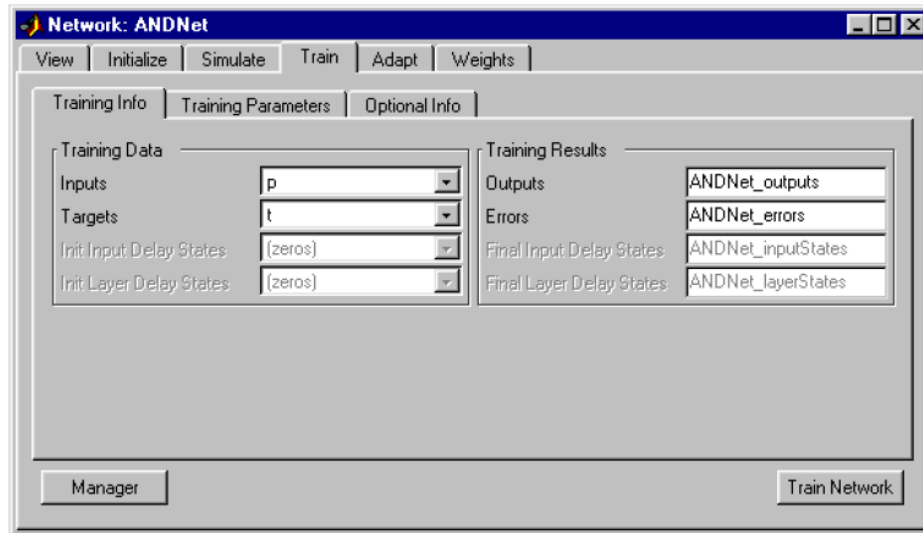
d. Claim 24

Buck/Wilt discloses this limitation. Ex1003, ¶¶219–21.

As explained for Claim 23, the Buck/Wilt system allows a user to input commands and information through a user input interface. Ex1003, ¶¶214–20. Those inputs would include setting or editing the parameters of the artificial neural network algorithms when using Buck’s system to process an artificial neural network. Ex1003, ¶220. For example, Buck discusses loading “initial weights” and “other learning parameters” during training and testing. Ex1004, [0235]–[0249]. Thus, Buck discloses that a user could enter or import parameters of the computations to set up and run the networks for training or testing. Ex1003, ¶220. It would have been obvious that a user can define and edit the parameters of the computations representing an artificial neural network as appropriate for their use case. Ex1003, ¶¶219–21.

(1) nnet (Grounds 2, 5, 6)

nnet also discloses allowing users to change various parameters (e.g., weights and biases) of an artificial neural network when training a network through nnet’s GUI. *See* Ex1006, 95.



Id.;Ex1003, ¶¶324–26.

e. Claim 25

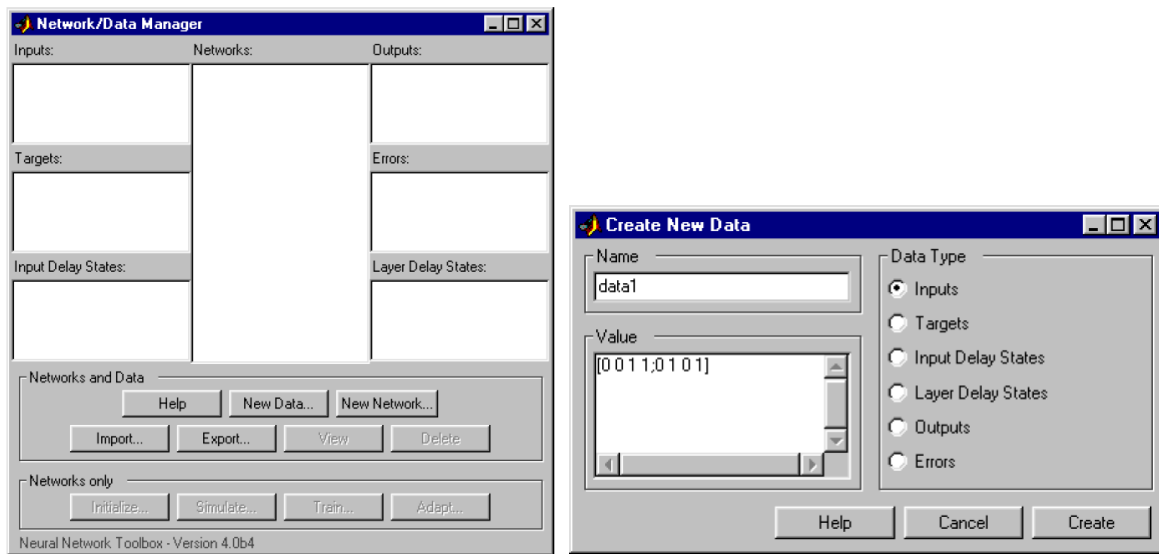
Buck/Wilt discloses this limitation. Ex1003, ¶¶222–24.

As explained for Claims 23 and 24, the Buck/Wilt system allows a user to provide information and parameters that will be used in the calculations during the training and testing of the artificial neural network through a user input interface. Ex1003, ¶¶214–23. For example, Buck discusses loading training data and query data during training and testing, along with parameters used for the inputs. Ex1004, [0235]–[0249]. Thus, Buck discloses that a user could set or edit parameters of the inputs to the artificial neural network, such as the source of inputs, weights, and learning parameters. Ex1003, ¶223. Additionally, it would have been obvious that a user can define and edit the parameters of the inputs to an artificial neural network,

because artificial neural networks operate on changing inputs to yield different outputs. Ex1003, ¶¶222–24.

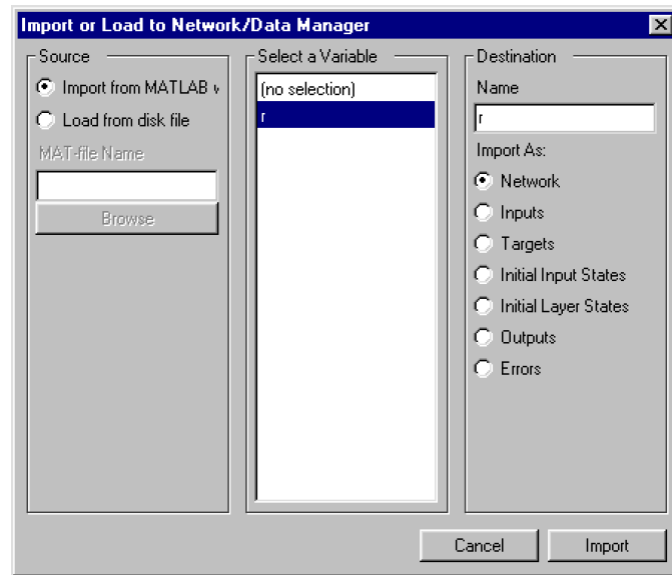
(1) nnet (Grounds 2, 5, 6)

nnet users can create input files and variables using nnet's command line interface and, as illustrated below, through its GUI. Ex1006, 92–100.



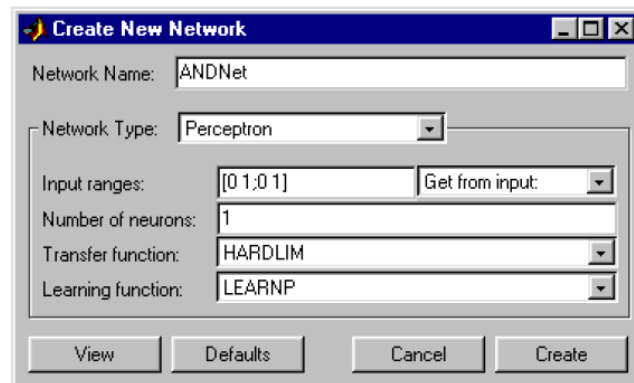
Id., 92, 93.

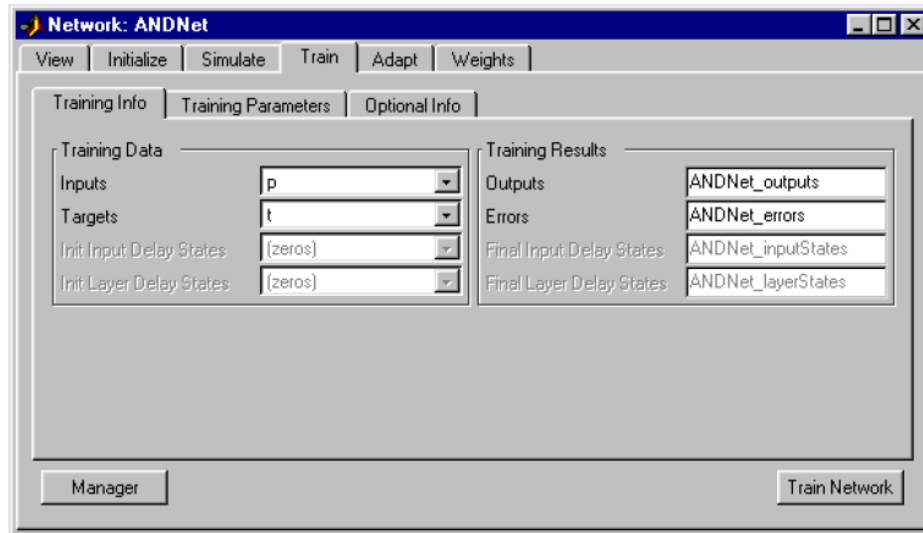
Users can also import data from the workspace or load data from a file through the command line interface or the GUI. Ex1006, 92–100.



Id., 99.

Regardless of how the input variables or files are obtained, nnet allows users to select input sources and ranges when creating, training, and simulating neural networks.





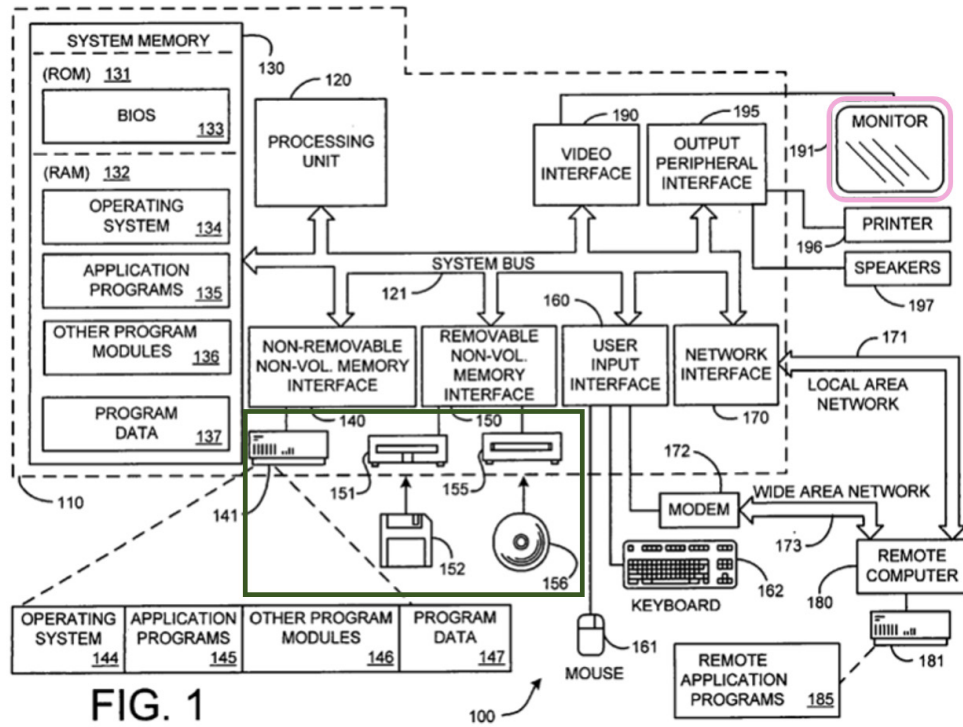
Id., 94, 95.

The range of data from within an input file or anything that will be used for the data (e.g., the “Training Parameters” shown in nnet) are a “parameter” of the input. Ex1003, ¶¶327–32.

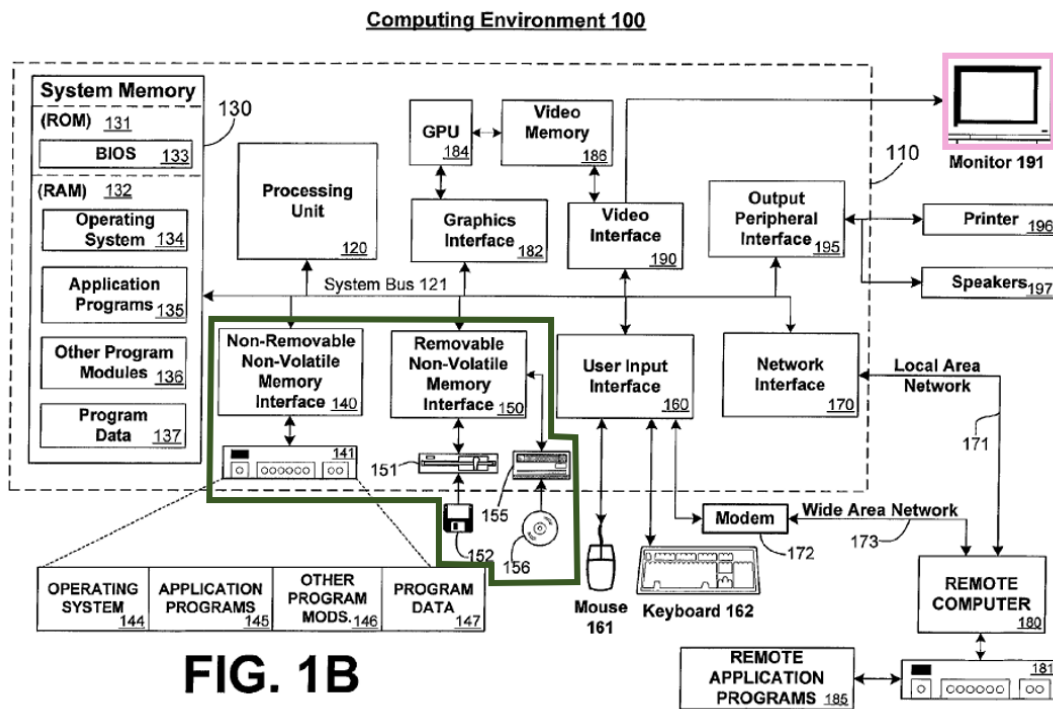
f. Claim 26

Buck/Wilt discloses this limitation. Ex1003, ¶¶225–28.

Buck discloses that the output of a computation performed by its GPU-accelerated system “is displayed on the display device 191” (annotated in pink) or stored by storage media such as a “hard disk drive 141,” a “magnetic disk drive 151,” and an “optical disk drive 155” that “reads from or writes to” “non-removable, nonvolatile magnetic media,” “a removable nonvolatile magnetic disk 152,” and “a removable, nonvolatile optical disk 156,” respectively (annotated in dark green). Ex1004, [0046]–[0047], [0060].



Id., Fig. 1 (annotated).

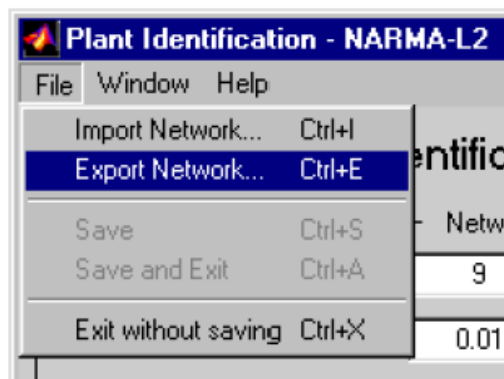


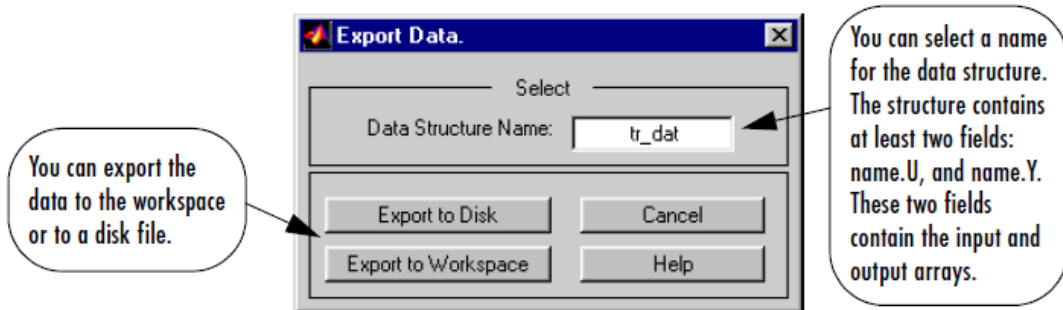
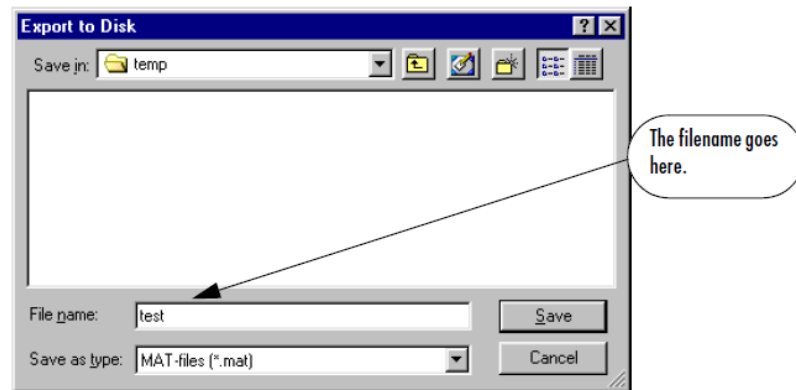
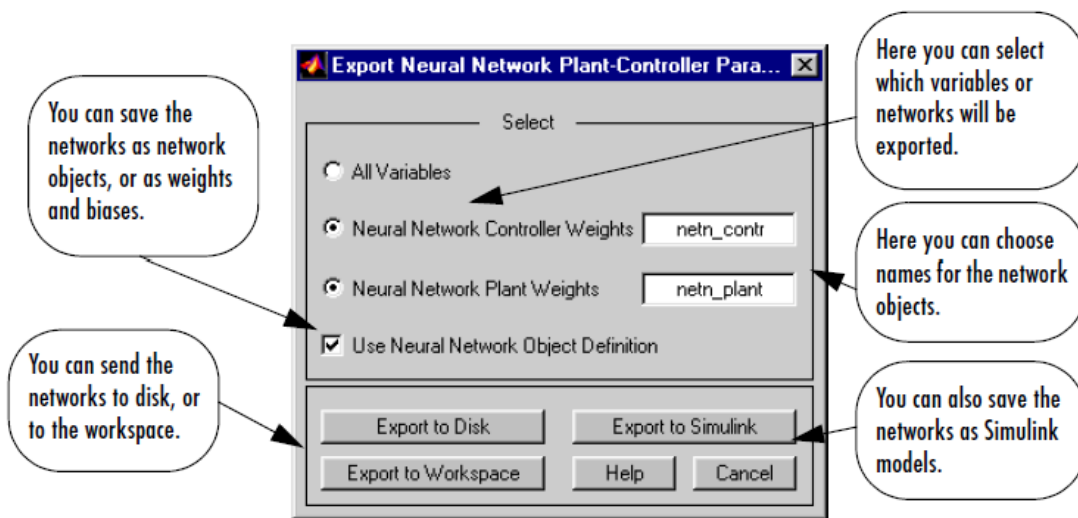
Ex1005, Fig. 1B (annotated), [0070].

As explained for Claims 23 and 24, a user can provide commands and information to Buck/Wilt's computer through a user input interface. Ex1003, ¶¶214–21, 227. Those commands would include specifying an output to be saved to disk and/or displayed on a screen to view or save the results of training or testing the artificial neural networks. *Id.* Thus, Buck/Wilt discloses that a user could specify an output to be saved to disk and/or displayed on a screen. *Id.* It would have been obvious that a user can choose to save to disk or display the output of a set of computations representing an artificial neural network. *Id.* Indeed, a user of the Buck/Wilt system simulating an artificial neural network would want to be able to review the output of the simulation to use, modify, or perform further analysis on the results. Ex1003, ¶¶225–28.

(1) nnet (Grounds 2, 5, 6)

nnet allows users to save neural network outputs, networks themselves, and error reports to disk or to the workspace using nnet's command line interface and, as illustrated below, through its GUI. Ex1006, 97–98, 237–243.

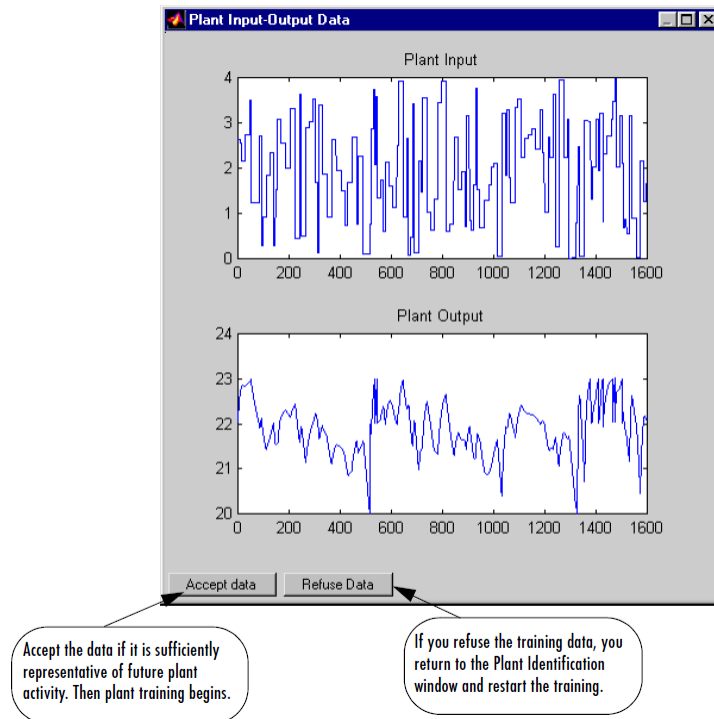




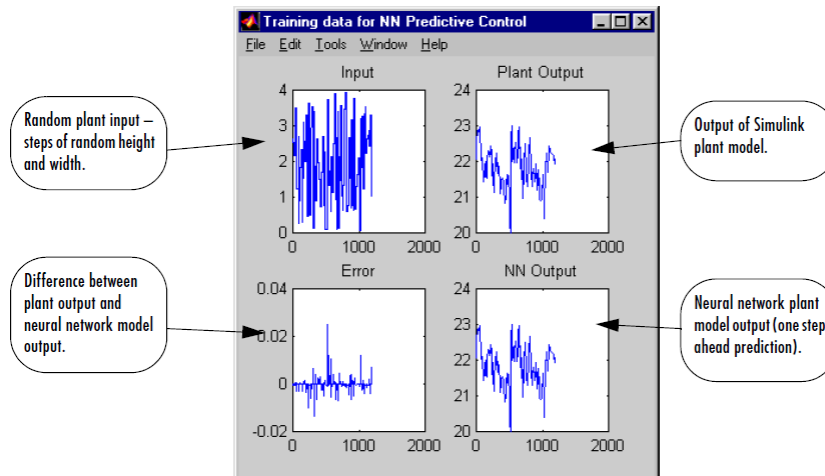
Id., 237, 238, 241.

In addition to displaying training data and neural network output data, nnet functions “disp” and “display” allow a user to display a neural network’s properties

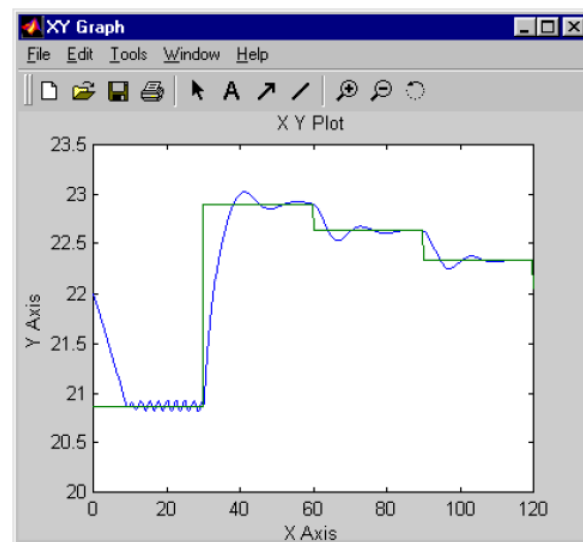
and its variables' names and properties. Ex1006, 454 (“Network Use Functions” including “disp” to “Display a neural network’s properties” and “display” to “Display a neural network variable’s name and properties”), 503 (“disp”), 504 (“display”).



Ex1006, 217 (output of “Generate Training Data”).



Ex1006, 218 (output of “Train Network”).



Ex1006, 219 (output of “Simulate”); Ex1003, ¶¶333–36.

g. Claim 27

Buck/Wilt discloses this limitation. Ex1003, ¶¶229–32.

Buck explains that “[f]unctions operating on the inputs are expressed as sequences of instructions,” and that a “complete program is broken into successive passes of these functions over the data,” where each pass represents “a single data

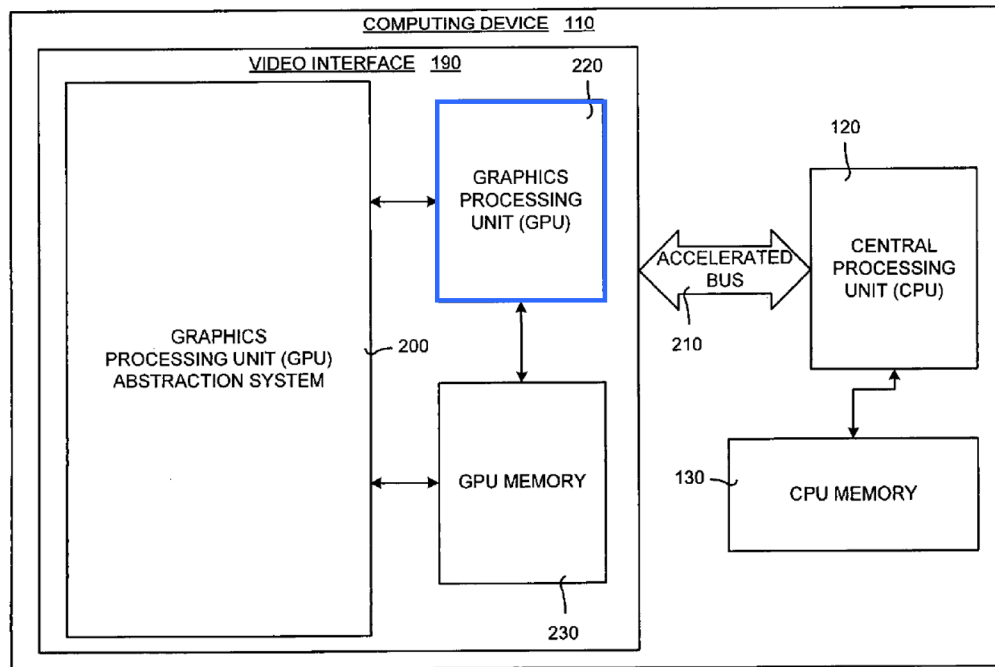
parallel operation in the program.” Ex1004, [0069]–[0075], [0076]–[0210], Fig. 8. Buck’s discussion of breaking an application or function into a series of passes teaches parsing elements to be used in the computations representing the artificial neural network. Ex1003, ¶230.

And, as explained for Claims 23 and 24, Buck/Wilt system allows a user to input commands and information, such as defining, setting, or editing of computational elements through a user input interface. Ex1003, ¶¶214–221, 231. Those inputs would include commands parsing elements to be used in the artificial neural network computations. Ex1003, ¶¶229–32.

h. Claim 28

Buck/Wilt discloses this limitation. Ex1003, ¶¶233–38.

As explained for 21[b] and illustrated below, the Buck/Wilt processing unit is a GPU (GPU 220, annotated in blue).



Ex1004, Fig. 1 (annotated).

As explained for Claim 27, the Buck/Wilt system parses machine learning algorithms into “successive passes.” Ex1004, [0069]–[0075]. Each pass is implemented with “simple primitives (or building blocks)” “using one or more pixel shaders,” or programs expressed as a “series of instructions.” *Id.*, [0076]–[0087]. Thus, the Buck/Wilt system teaches converting the elements into GPU programs. Ex1003, ¶¶235–36.

As explained for Claims 23 and 24, the Buck/Wilt system allows a user to provide commands and information to its computer to define, set, and edit the computations representing the artificial neural network through a user input interface. *Id.*, ¶237. Those inputs would include commands that result in converting the successive passes (elements) into GPU programs. It was well-known before the

'461 patent that GPGPU applications were implemented on GPUs through programmable shaders, and thus it would have been obvious that a user instruction to implement a GPGPU algorithm would result in converting elements of a computation in the GPGPU algorithm into GPU programs. *See* Section III.A.

i. Claim 29

Buck/Wilt discloses this limitation. Ex1003, ¶¶239–41.

Buck explains that “[a]ctual execution” of algorithms “in the GPU requires that the list of passes be created in a preprocess,” and that the “individual functions must be compiled and downloaded to the GPU.” Ex1004, [0228]. And as discussed for Claims 23 and 24, Buck/Wilt discloses a user interface through which a user can enter commands into the computer to define, set, and edit the computations representing the artificial neural network. Ex1003, ¶240. Those commands can include instructions to execute a function, which would initiate compilation of the GPU programs. *Id.*, ¶¶239–41.

j. Claim 30

Buck/Wilt discloses this limitation. Ex1003, ¶¶242–45.

See 21[a]. Ex1003, ¶¶165–75, 243. Buck describes loading the “programs P” for the different shaders into its segmented GPU memory as part of the training and testing process for an artificial neural network. Ex1004, [0236], [0246], Figs. 15, 16. As discussed for 21[a], the various segments of GPU memory 230 disclose

multiple memory partitions. *Id.*, [0070], [0122]–[0124], [0235]–[0236], Figs. 9, 15, 16. Thus, Buck’s disclosure of loading the shader programs P to GPU memory, is a transfer of the program into the second memory partition. Ex1003, ¶243.

As discussed for Claims 23 and 24, Buck/Wilt discloses a user interface through which a user can enter commands and information into the computer. Ex1003, ¶¶214–21, 244. Those commands would include commands to transfer the GPU programs to a segment of GPU memory (e.g., the segment for shader programs, which is a second memory partition) so they can be executed. Ex1004, [0011]–[0012]; Ex1003, ¶¶242–45.

k. Claim 31

Buck/Wilt discloses this limitation. Ex1003, ¶¶246–49.

As explained for 21[a] and Claim 26, Buck’s CPU maintains its own “stream[s]” of instructions and data, including a data output stream, and is connected to a number of storage media, such as hard drive disk, magnetic disk drive, and optical disk drive through various memory interfaces. Ex1003, ¶¶165–75, 225–28, 247–48. As the host processor, the CPU manages the flow of data within the system and transfers an output of the computations representing the artificial neural network being performed by its GPU to the hard drive disk. Ex1003, ¶¶246–49.

l. Claim 32

Buck/Wilt discloses this limitation. Ex1003, ¶¶250–56.

The Buck/Wilt system includes a variety of input devices such as “a television or broadcast video receiver” that are capable of providing streaming video data. Ex1004, [0049]–[0050]; Ex1005, [0071]. These devices are “connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121” or through “other interface and bus structures.” Ex1004, [0049]–[0050]; Ex1005, [0071]. The system expressly supports “real-time processing” and uses “predominantly DMA-based input,” which is commonly used with “peripheral” devices, “such as a digital camera.” Ex1005, [0085], [0093], [0107].

It would have been obvious to a POSITA that, in Buck/Wilt, the CPU receives a stream of input data from a device such as a video camera. Ex1003, ¶253. Like the “video receiver” in Buck/Wilt, a video camera provides video input. Thus, a POSITA would have understood that Buck/Wilt’s optimized GPU-accelerated system can be used to process a stream of video data input generated by video camera. *Id.*

Furthermore, the Buck/Wilt resource management system “facilitate[s] the host microprocessor’s performance of useful work while the coprocessor hardware is running.” Ex1005, [0114]. Therefore, the Buck/Wilt CPU can continue to receive and store input data gathered by the video camera in real-time and transmitted to the CPU while the GPU is executing computations, and this video camera data is used as input to the Buck/Wilt artificial neural network. Ex1003, ¶¶250–56.

(1) ANN (Grounds 3, 5)

In the Buck/Wilt-ANN combination, the input data received by the CPU is real-time video data from a video camera. Ex1007, 3–5; Ex1003, ¶¶361–64, 366–67.

ANN discloses receiving visual data from a soccer-playing robot in real-time to “recognize” and “trace” the ball. Ex1007, 2–3. ANN explains that systems having a GPU with a “video in” function can retrieve image information “directly” from the video camera and store that information in GPU memory, allowing the system to avoid “transfer[ring] data between CPU and GPU” and “speed [up] the process.” *Id.* GPUs that do not have a “video in” function, such as the ANN system, require the CPU to receive, process, and transfer video data to GPU memory. Thus, real-time video data from ANN’s video camera is input to the Buck/Wilt CPU through the user input interface (or another interface) and is transferred from the CPU to GPU memory for processing. Ex1007, 5; Ex1003, ¶¶364, 366–67.

(2) GPU Gems (Grounds 4, 6)

In the Buck/Wilt/GPU Gems combination, the input data received by the CPU is real-time video data from a video camera. Ex1008, 649–66; Ex1003, ¶¶361–63, 365–67.

GPU Gems discloses receiving and processing video data from a video camera to “detect and track distinctive points (feature) in real time as the camera pans.”

Ex1008, 661. Real-time video data from GPU Gems' video camera is input to the Buck/Wilt CPU through the user input interface (or another interface) and is transferred from the CPU to GPU memory for processing. Ex1008, 649–66; Ex1003, ¶¶365–67.

m. Claim 38

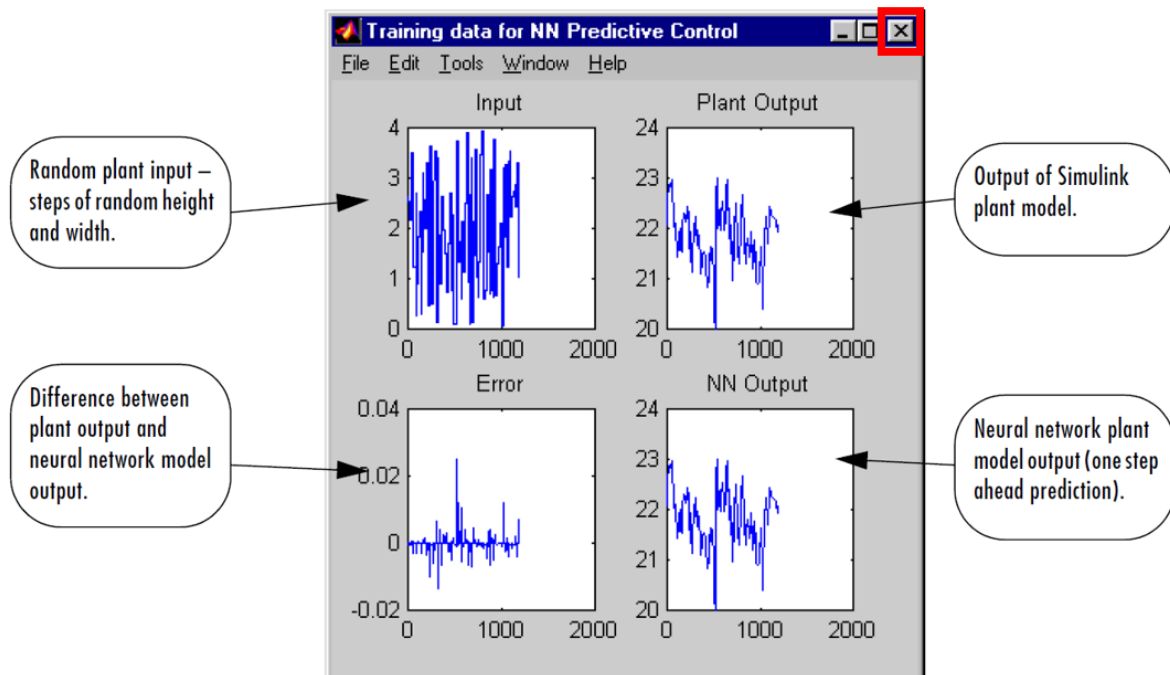
Buck/Wilt discloses this limitation. Ex1003, ¶¶257–60.

As discussed for 21[d], the Buck/Wilt system for implementing artificial neural networks supports preemption. Ex1003, ¶¶191–95, 258. Certain commands—such as halting a simulation or changing the input data for the simulation—can be assigned a higher priority value than a currently executing simulation, and preempt (interrupt) the currently executing artificial neural network computation. *Id.*, ¶258.

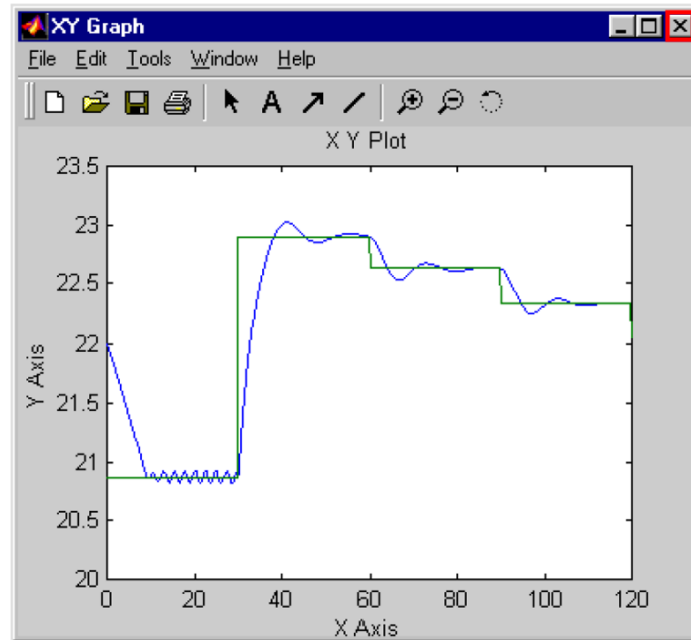
As discussed for Claims 23 and 24, Buck/Wilt discloses a user interface through which a user can enter commands and information into the computer to define, set, and edit the computations representing the artificial neural network through a user input interface. *Id.*, ¶¶214–21, 259. Those user interactions, or user commands, would include commands that interrupt the artificial neural network computation. *Id.*, ¶¶257–60.

(1) nnet (Grounds 2, 5, 6)

nnet's GUI provides a user with interface elements that allow a user to close the window or cancel a process. Ex1006, 218–219. These, and other commands, can be high priority commands that preempt, or interrupt, the execution of a currently running command buffer on the Buck/Wilt/nnet system. Ex1003, ¶¶337–39.



Ex1006, 218 (output of "Train Network").



Ex1006, 219 (output of “Simulate”).

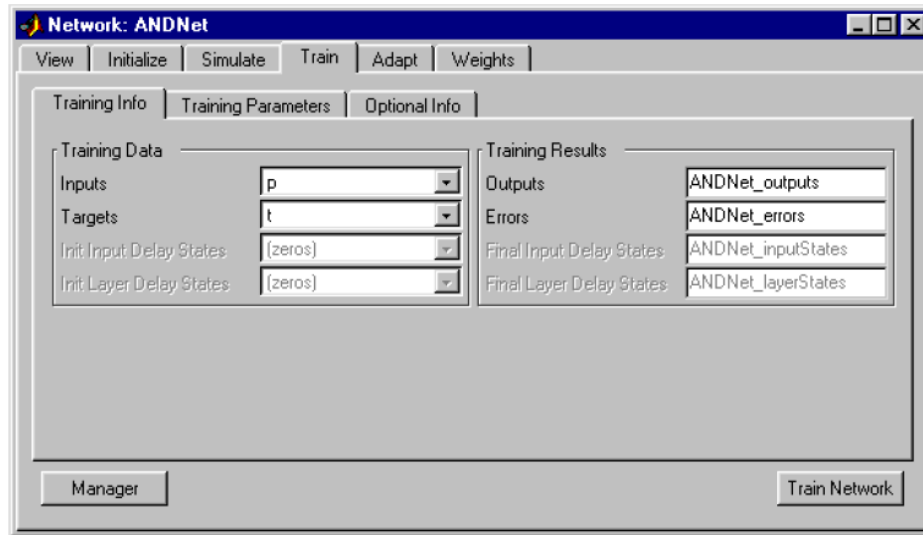
n. Claim 39

Buck/Wilt discloses this limitation. Ex1003, ¶¶261–63.

As explained for Claims 23 through 25, a user can provide user interactions, or user commands, through the user input interface to the Buck/Wilt system to define, set, and edit inputs to the artificial neural network, including weights, biases, shaders, and textures, through a user input interface. Ex1003, ¶¶214–24, 262. Furthermore, it would have been obvious that a user can cause a change in the inputs to an artificial neural network, because artificial neural networks operate on changing inputs to yield different outputs. Ex1003, ¶¶261–63.

(1) nnet (Grounds 2, 5, 6)

See Claims 23 to 25. nnet discloses that the user command received by the user interaction stream causes a change in the inputs to the artificial neural network.



Ex1006, 92–100; Ex1003, ¶¶340–42.

o. Claim 40

Buck/Wilt discloses this limitation. Ex1003, ¶¶264–66.

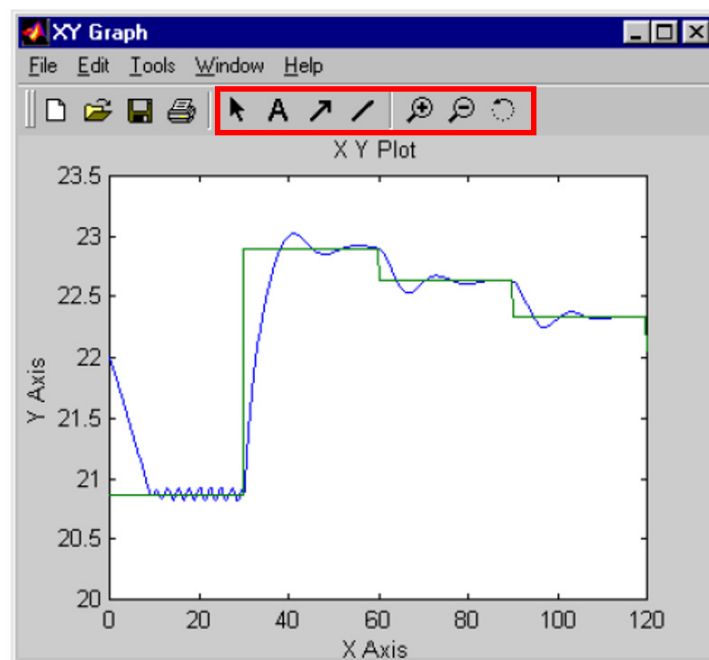
As explained for Claims 23, 24, and 26, the Buck/Wilt system user input interface allows a user to provide commands and information to its computer to specify an output of the artificial neural network to be displayed through a user input interface. Ex1003, ¶¶214–21, 225–28, 265. These user interactions, or user commands, can include changing display properties of the output to be displayed. *Id.* For example, Buck discloses that the output of a computation performed by its GPU-accelerated system “is displayed on the display device 191.” Ex1004, [0060].

A POSITA would have found it obvious that the user interaction, or user command, could cause a change in display properties of an output of the computations. Ex1003, ¶265. Changing display properties, such as color, size, etc. of an output of a simulation was standard for user interfaces at the time of the '461 patent. *Id.*, ¶¶31–85, 264–66.

(1) nnet (Grounds 2, 5, 6)

nnet teaches that an application can receive user commands that cause a change in display properties of an output of the computations representing the artificial neural network. Ex1003, ¶¶343–46.

An nnet user can change display properties of an output of the computations by using standard interface functions like zooming in and out, rotating a plot, and adding labels. Ex1006, 219.



Id., 219.

p. Claim 41

Buck/Wilt discloses this limitation. Ex1003, ¶¶267–70.

As explained for Claim 21, by preventing data exchange when a computation is being executed, the command buffers (part of the computational stream) control the data exchange between the user interaction stream executed by the CPU and the computational stream executed by the GPU. *Id.*, ¶¶160–208, 268.

Just as in the '461 patent, in the Buck/Wilt system, when a computation (i.e., from a command buffer) is executing, new shaders and textures are made available to the GPU through the command buffer “not when ready but upon the request of the controller.” Ex1001, 9:10–52. Wilt’s GPU can notify the CPU when certain events occur, such as “the completion, or partial completion, of a command buffer, completion of a DMA operation to write data into memory visible to the host processor...or completion of a specific command.” Ex1005, [0088], [0094]. Once Wilt’s GPU has completed execution of a command buffer or the command buffer has been preempted, the GPU can notify the CPU that it is no longer executing that command buffer, signalling that it can now execute another command buffer and change the instructions it executes or the data on which it operates. *Id.*

It would have been obvious to a POSITA that, if the GPU has completed execution of the command buffer and new input data is required, the GPU’s

notification could result in a user being prompted for input data. Ex1003, ¶269. Because the completion or interruption of a command buffer (a portion of the computational stream) causes the GPU to provide a notification to the CPU, which can, in turn, cause a user to be prompted for input data, the command buffer controls the data exchange between the user interaction stream and the computational stream by requesting the inputs. *Id.* If the required resources included inputs to the artificial neural network, the command buffer would cause requesting the inputs from the user and transferring them from the CPU to the GPU memory. *Id.*, ¶¶267–70.

q. Claim 33

i. 33[pre]

To the extent the preamble is limiting, Buck/Wilt discloses it. Ex1003, ¶¶271–72.

See 21[pre]. Ex1003, ¶¶160–64. Buck/Wilt discloses “systems and methods” for executing computations representing artificial neural networks.

ii. 33[a]

Buck/Wilt discloses this limitation. Ex1003, ¶¶273–75.

See 21[pre]; Claim 32. Ex1003, ¶¶160–64, 250–56. Buck/Wilt discloses a video camera that generates inputs to the artificial neural network and thus discloses a camera to acquire input data for the artificial neural network.

(1) ANN (Grounds 3, 5)

See Claim 32 (a camera that acquires input for the artificial neural network).
Ex1003, ¶¶370–72.

(2) GPU Gems (Grounds 4, 6)

See Claim 32 (a camera that acquires input for the artificial neural network).
Id.

iii. 33[b]

Buck/Wilt discloses this limitation. Ex1003, ¶¶276–77.
See 21[pre]. Ex1003, ¶¶160–64.

iv. 33[c]

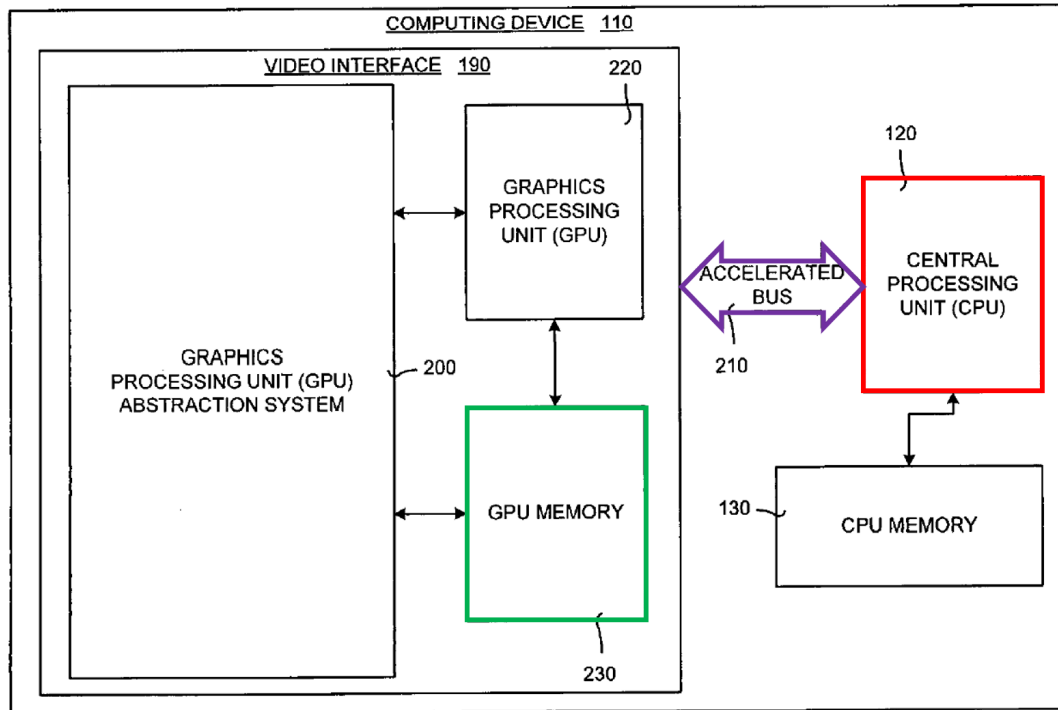
Buck/Wilt discloses this limitation. Ex1003, ¶¶278–79.
See 21[pre]. Ex1003, ¶¶160–64.

v. 33[d]

Buck/Wilt discloses this limitation. Ex1003, ¶¶280–86.
See 21[pre] and 21[a] (a CPU that executes a user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition). Ex1003, ¶¶160–75, 281.

Buck’s central processing unit (CPU 120, red) is “in communication with” video interface 190, of which the first and second memory partitions (within GPU

memory 230, green) are part, via accelerated bus 210 (purple). Ex1004, [0049]–[0050], [0055]–[0057]; Ex1005, [0067]–[0071].



Ex1004, Fig. 2 (annotated).

As explained for Claim 32, the Buck/Wilt CPU can receive real-time input data from an input device such as a video camera. Ex1003, ¶¶250–56, 283. A video camera providing input to the Buck/Wilt system would be connected to the CPU through the system bus and a user input interface (or other interface and bus structures, such as “a parallel port, game port or a universal serial bus (USB)”). Ex1004, [0050]; Ex1005, [0071].

Buck/Wilt’s resource management system “facilitate[s] the host microprocessor’s performance of useful work while the coprocessor hardware is

running.” Ex1005, [0114]. Therefore, Buck/Wilt’s CPU can continue to receive and store input data generated by the video camera while the GPU is executing computations. Ex1003, ¶284. Because this data is used as input to the Buck/Wilt artificial neural network, the CPU transfers the data to the GPU while the GPU continues to perform computations representing the artificial neural network. *Id.*; Ex1007, 5.

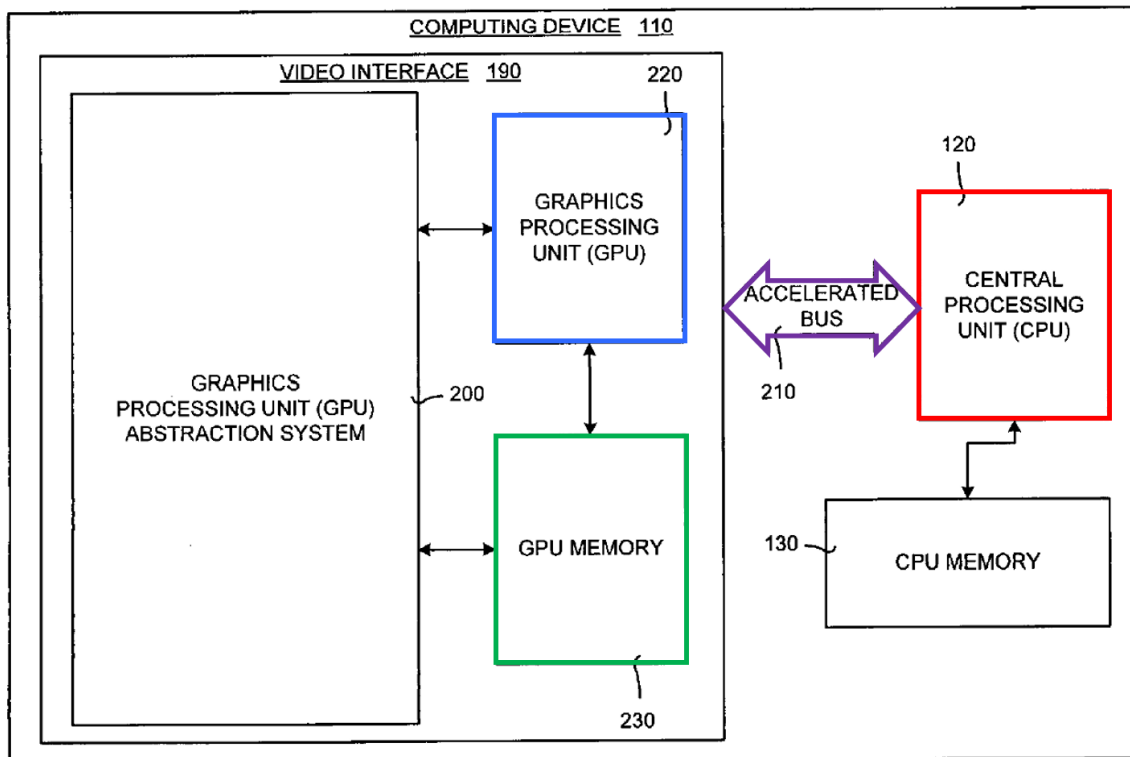
And, as explained for Claim 21, Buck/Wilt’s system discloses processing commands and information, including video data, received through the user input interface and transferring the resulting textures and shader programs to various segments of GPU memory. Ex1003, ¶¶160–208, 280–86.

vi. 33[e]

Buck/Wilt discloses this limitation. Ex1003, ¶¶287–293.

See 21[pre], 21[b], and 21[c] (a processing unit that executes a computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network). Ex1003, ¶¶160–64, 176–90, 288.

Buck’s processing unit (GPU 220, blue) is “operably coupled to” first and second memory partitions (within GPU memory 230, green) and to CPU 120 (red) via accelerated bus 210 (purple). Ex1004, [0049]–[0050], [0055]–[0057]; *see also* Ex1005, [0067]–[0071].



Ex1004, Fig. 2 (annotated).

Furthermore, as explained for Claim 41, Wilt's command buffer controls the data exchange between the user interaction stream and the computational stream by requesting the inputs. Ex1003, ¶¶267–70, 291. Specifically, the completion or interruption of a command from the command buffer causes the GPU to provide a notification to the CPU, which can then cause a user to be prompted for input data. *Id.* Wilt's scheduler makes resources and data available according to the requirements of the command buffers (part of the computational stream). Ex1005, [0024]. The scheduler can then facilitate the transfer of this input data from the CPU (first memory partition) to a segment of the GPU memory (second memory partition). Ex1003, ¶291.

And, as explained for 33[d], the Buck/Wilt CPU can continue to receive and store input data acquired by the camera while the GPU is executing computations. Ex1003, ¶¶280–93.

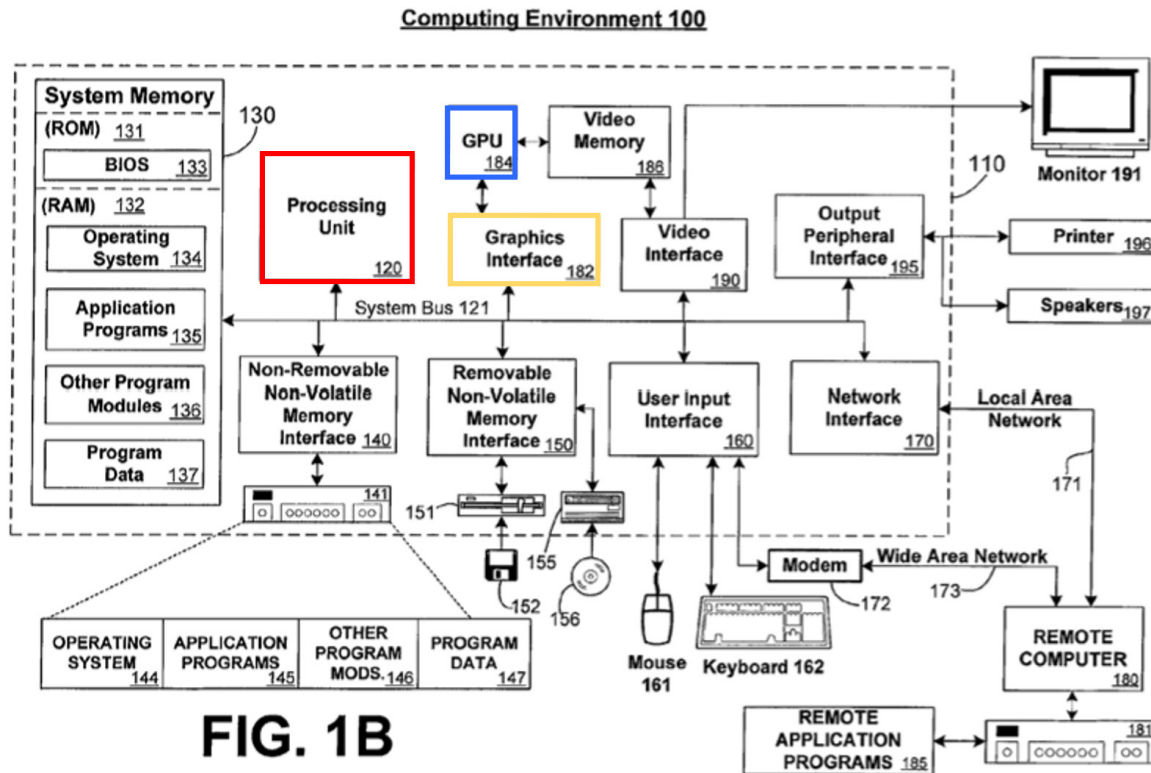
vii. 33[f]

Buck/Wilt discloses this limitation. Ex1003, ¶¶294–300.

See 21[e] and 21[f] (queuing a user command received by the user interaction stream during execution of the computations representing the artificial neural network). Ex1003, ¶¶196–208, 295.

Furthermore, Buck/Wilt’s scheduler, a component of “the operating system of the host computing system,” schedules and queues GPU command buffers. Ex1005, [0024]. The Buck/Wilt scheduler, which is operably coupled to the CPU and GPU, is a controller because it schedules and controls the execution of tasks within the system. Ex1003, ¶296; Ex1001, 5:22–24 (“a controller can be partially or even fully implemented in software, in one example.”).

Additionally, and in the alternative, a POSITA would have understood that Buck/Wilt’s scheduler could be implemented on Wilt’s graphics interface 182 in either software or hardware. Ex1003, ¶297. Wilt’s graphics interface 182 (yellow) communicates with the processing unit (GPU 184, blue) and CPU (processing unit 120, red). Ex1005, [0067]–[0071].



Ex1005, Fig. 1B (annotated).

The graphics interface “communicates with the CPU” over system bus 121 and “assumes responsibility” for Accelerated Graphics Port (AGP) communications that “transfer data from system memory to a graphics processor.” Ex1005, [0071]. Therefore, Buck/Wilt’s method of scheduling and queuing command buffers could be performed and communicated to the GPU through a controller such as graphics interface 182. Ex1003, ¶298.

Additionally, a POSITA would have understood that the scheduler queues tasks to be executed at times that would not cause data corruption. *Id.*, ¶299. Although the ’461 patent does not provide any examples of how to avoid data

corruption, data corruption was well-known as an undesirable outcome that could cause the system to malfunction or behave erratically, and a POSITA designing a sophisticated system for managing coprocessor resources and implementing artificial neural networks would have designed the scheduler to queue tasks to be executed at times that “avoid data corruption.” *Id.* For example, for a real-time input, it was known that data corruption could occur if a user interaction is processed before computations on a frame are finished and written to memory. *Id.* A POSITA therefore would have understood the basic scheduling principle of allowing necessary computations and commands to occur before allowing interruption. *Id.*, ¶¶294–300.

r. Claim 34

Buck/Wilt discloses this limitation. Ex1003, ¶¶301–02.

See Claim 38. The user commands can be received through user interactions. *Id.*, ¶¶257–60, 301–02.

(1) nnet (Grounds 2, 5, 6)

Id., ¶¶347–48.

(2) ANN (Grounds 3, 5)

Id., ¶¶393, 397–402.

(3) GPU Gems (Grounds 4, 6)

Id.

s. Claim 35

Buck/Wilt discloses this limitation. Ex1003, ¶¶303–05.

See Claim 39. The user commands can be received through user interactions.

Id., ¶¶261–63, 303–05.

(1) nnet (Grounds 2, 5, 6)

Id., ¶¶349–50.

(2) ANN (Grounds 3, 5)

Id., ¶¶394, 397–402.

(3) GPU Gems (Grounds 4, 6)

Id.

t. Claim 36

Buck/Wilt discloses this limitation. Ex1003, ¶¶306–07.

See Claim 40. The user commands can be received through user interactions.

Id., ¶¶264–66, 306–07.

(1) nnet (Grounds 2, 5, 6)

Id., ¶¶350–52.

(2) ANN (Grounds 3, 5)

Id., ¶¶395, 397–402.

(3) GPU Gems (Grounds 4, 6)

Id.

u. Claim 37

Buck/Wilt discloses this limitation. Ex1003, ¶¶308–10.

See Claim 41; 33[f]. Based on the resources and data required by specific command buffers, the Buck/Wilt scheduler (controller) makes certain resources and data available to the GPU. Therefore Buck/Wilt teaches that the controller requests the input data to be transferred to GPU memory during execution of the artificial neural network computations. *Id.*, ¶¶267–70, 294–300.

(1) ANN (Grounds 3, 5)

Id., ¶¶396, 397–402.

(2) GPU Gems (Grounds 4, 6)

Id.

V. ENABLEMENT AND SECONDARY CONSIDERATIONS

The prior art references relied upon in this Petition are enabled. Ex1003, ¶¶112–45.

Petitioner is not aware of any secondary considerations of non-obviousness tied to the Challenged Claims. Ex1003, ¶403. If Patent Owner alleges otherwise, Petitioner may request leave to file a reply to respond. Regardless, attempts by Patent Owner to rely on alleged secondary considerations of non-obviousness cannot overcome the strong showing of obviousness detailed above. Where, as here, there is a strong showing of obviousness, even relevant secondary considerations

supported by substantial evidence do not dislodge the primary conclusion of obviousness. *Ohio Willow Wood Co. v. Alps South, LLC*, 735 F.3d 1333, 1344 (Fed. Cir. 2013).

VI. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

A. Discretionary Denial Under Section 314(a) Is Not Warranted

The Board should not exercise its discretion under Section 314(a) to deny institution. Although there is currently a district court case against NVIDIA in which the '461 patent is asserted, the six-factor test set forth in *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 at 5–6 (PTAB Mar. 20, 2020) (precedential), favors institution. Given the compelling evidence of unpatentability described above, the *Fintiv* factors should not lead to discretionary denial.

Factor 1. (neutral) There is no evidence that the court will or will not grant a stay if IPR is instituted, and no stay motion has yet been filed. *TSMC Ltd. v. Alidouble, Inc.*, IPR2023-00372, Paper 7 at 10 (P.T.A.B. Jul. 13, 2023).

Factor 2. (favors institution) Trial in the Litigation is currently scheduled to begin on September 8, 2026—more than eighteen months from the date of this Petition and after the FWD deadline. *Samsung Elecs. Co. v. Staton Techiya, LLC*, IPR2022-00302, Paper 13 at 10–12 (P.T.A.B. Jul. 11, 2022).

Factor 3. (neutral) Although a *Markman* hearing is scheduled for July 29, 2025, after which fact discovery will open, the parties “have otherwise not invested

a great deal” in the Litigation. *CrowdStrike, Inc., v Webroot Inc.*, IPR2023-00126, Paper 9 at 10 (P.T.A.B. May 5, 2023). This factor is thus neutral.

Factor 4. (favors institution) This Petition challenges 10 claims not at issue in the Litigation, and Petitioner stipulates that, if the Board institutes IPR, Petitioner will not seek resolution of invalidity in the Litigation based on the specific grounds asserted in this Petition. Ex1036.

Factor 5. (favors institution) Although Petitioner and Patent Owner are also parties to the Litigation, the expected timing of the trial favors institution. *Fusion Orthopedics, LLC v. Extremity Medical, LLC*, IPR2023-00894 Paper 15 at 32 (P.T.A.B. Nov. 17, 2023).

Factor 6. (strongly favors institution) This petition presents strong arguments that all claims of the '461 patent are invalid as obvious.

In totality, the *Fintiv* factors weigh strongly in favor of institution. The Board should therefore not exercise its discretion under Section 314(a) to deny institution.

B. Discretionary Denial Under Section 325(d) Is Not Warranted

Under *Advanced Bionics*, and applying the *Becton Dickinson* factors, the Board should not deny institution under Section 325(d). Here, only ANN was of record during prosecution of the '461 patent, but ANN was not applied in rejections during the reissue prosecution. When a reference is cited in prosecution but not used in any rejection, this “weighs strongly against [the Board] exercising [its] discretion

to deny institution” under Section 325(d). *Adobe Inc. v. RAH Color Techs.*, IPR2019–00628, Paper 37 at 20 (P.T.A.B. Aug. 20, 2019).

Further, none of Buck, Wilt, nnet, or GPU Gems was before the Examiner during prosecution,³ and even ANN was among hundreds of references Applicant cited to the Examiner. None of the combinations of prior art in this Petition was presented to the Examiner nor was testimony similar to that provided here by Dr. Rosing. *Id.* at 21.

Section 325(d)’s concern with avoiding duplicative and inefficient reexamination of references already examined is mitigated here. This Petition provides a focused and substantive review of the references in the context of the grounds raised herein.

³ During prosecution, the Examiner considered Buck ’085, which claims priority to a provisional application filed on the same day as the provisional application to which Buck claims priority. Buck, however, contains disclosure not in Buck ’085 that is relied on in this Petition. Ex1004, [0058]–[0060], [0088]–[0103], [0106]–[0158], Figs. 3, 8, 9, 10A, 10B.

VII. CONCLUSION

This Petition establishes a reasonable likelihood that the Challenged Claims are unpatentable. Petitioner respectfully requests that IPR be instituted and that all Challenged Claims be found unpatentable.

DATED: March 6, 2025

Respectfully submitted,

/Brian M. Buroker/

Brian M. Buroker (Reg. No. 39,125)
Gibson, Dunn & Crutcher LLP
1700 M Street NW
Washington, DC 20036
Phone: (202) 955-8500
Fax: (202) 467-0539
Email: bburoker@gibsondunn.com

Attorney for Petitioner NVIDIA Corporation

CERTIFICATION UNDER 37 C.F.R. § 42.24(d)

Under the provisions of 37 C.F.R. § 42.24(d), the undersigned attorney hereby certifies that the word count for Sections I–VII of the foregoing Petition for *Inter Partes* Review is 13,940 according to the word count tool in Microsoft Word.

DATED: March 6, 2025

Respectfully submitted,

/Brian M. Buroker/

Brian M. Buroker (Reg. No. 39,125)
Gibson, Dunn & Crutcher LLP
1700 M Street NW
Washington, DC 20036
Phone: (202) 955-8500
Fax: (202) 467-0539
Email: bburoker@gibsondunn.com

Attorney for Petitioner NVIDIA Corporation

CERTIFICATE OF SERVICE

The undersigned certifies service pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(a), (b) on the Patent Owner via UPS overnight of a copy of this Petition for *Inter Partes* Review and supporting materials at the Patent Owner at the correspondence address of record for the '461 patent:

SMITH BALUCH LLP
376 Boylston St
Suite 401
Boston, MA
United States

A courtesy copy has also been served via UPS overnight on litigation counsel at:

Christopher C. Campbell
KING & SPALDING LLP
1700 Pennsylvania Avenue, NW
Suite 900
Washington, DC 20006
ccampbell@kslaw.com

DATED: March 6, 2025

Respectfully submitted,

/Brian M. Buroker/

Brian M. Buroker (Reg. No. 39,125)
Gibson, Dunn & Crutcher LLP
1700 M Street NW
Washington, DC 20036
Phone: (202) 955-8500
Fax: (202) 467-0539
Email: bburoker@gibsondunn.com

Attorney for Petitioner NVIDIA Corporation

APPENDIX: CHALLENGED CLAIM LISTING

No.	Limitation
21[pre]	A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:
21[a]	executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;
21[b]	executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;
21[c]	shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network;
21[d]	shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network;
21[e]	queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network; and
21[f]	executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream.
22	The method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network.

No.	Limitation
23	The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network.
24	The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network.
25	The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network.
26	The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen.
27	The method of claim 21, wherein executing the user interaction stream comprises: parsing elements to be used in the computations representing the artificial neural network.
28	The method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises: converting the elements into GPU programs.
29	The method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs.
30	The method of claim 29, wherein executing the user interaction stream comprises: transferring the GPU programs to the second memory partition.
31	The method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of the computations representing the artificial neural network to disk.
32	The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations.
33[pre]	A system for executing computations representing an artificial neural network, the system comprising:

No.	Limitation
33[a]	a camera to acquire input data for the artificial neural network;
33[b]	a first memory partition;
33[c]	a second memory partition;
33[d]	at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network;
33[e]	a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and
33[f]	a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption.
34	The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network.
35	The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network.
36	The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network.
37	The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network.

No.	Limitation
38	The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network.
39	The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network.
40	The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network.
41	The method of claim 21, wherein, during execution of the computations representing the artificial neural network, the computational stream controls the data exchange between the user interaction stream and the computational stream by requesting the inputs to the artificial neural network.