Recent Advances in Convolutional Neural Networks

Jiuxiang Gu*, Zhenhua Wang*, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang, *Member, IEEE*

arXiv:1512.07108v1 [cs.CV] 22 Dec 2015

***Abstract*—In the last few years, deep learning has lead to very good performance on a variety of problems, such as object recognition, speech recognition and natural language processing. Among different types of deep neural networks, convolutional neural networks have been most extensively studied. Due to the lack of training data and computing power in early days, it is hard to train a large high-capacity convolutional neural network without overfitting. Recently, with the rapid growth of data size and the increasing power of graphics processor unit, many researchers have improved the convolutional neural networks and achieved state-of-the-art results on various tasks. In this paper, we provide a broad survey of the recent advances in convolutional neural networks. Besides, we also introduce some applications of convolutional neural networks in computer vision.**

***Index Terms*—Convolutional Neural Network, Deep learning.**

I. Introduction

CONVOLUTIONAL Neural Network (CNN) is first introduced by LeCun *et al*. in [1] and improved in [2]. They developed a multi-layer artificial neural network called LeNet-5 which can classify handwritten digits. Like other neural networks, LeNet-5 has multiple layers and can be trained with the backpropagation algorithm [3]. It can obtain effective representations of the original image, which makes it possible to recognize visual patterns directly from raw pixels with little-to-none preprocessing. However, due to the lack of large training data and computing power at that time, LeNet-5 can not perform well on more complex problems, *e.g*., large-scale image and video classification.

Since 2006, many methods have been developed to overcome the difficulties encountered in training deep neural networks. Most notably, Krizhevsky *et al*. propose a classic CNN architecture and show significant improvements upon previous methods on the image classification task. The overall architecture of their method, *i.e*., AlexNet [4], is similar to LeNet-5 but with a deeper structure. It contains 8 learned layers (5 convolutional layers with pooling interspersed and 3 fully-connected layers) where the early layers are splitted over the two GPUs. ReLU [5] is used as the nonlinear activation function and Dropout [6] is used to reduce overfitting.

With the success of AlexNet, several works are proposed to improve its performance. Among them, three representative works are ZFNet [7], VGGNet [8] and GoogleNet [9]. ZFNet improves AlexNet by reducing the filter size of first layer from 11×11 to 7×7 as well as reducing the stride of the convolution from 4 to 2. In such a setting, the sizes of middle convolutional layers are expanded so as to capture more meaningful features. VGGNet pushes the network depth up to 19 weight layers and uses a very small filter size 3×3 in each convolutional layer. The results demonstrate that the depth is a critical factor for good performance. GoogleNet increases both the depth and the width of the network. It achieves a significant quality gain at modest increase of computational requirement compared to shallower and less wider networks.

Other than these works, there are also lots of works that aim to improve CNN in different aspects, *e.g*., layer design, activation function, loss function, regularization, optimization and fast computing, or apply CNN in different kinds of computer vision tasks. In the following sections, we identify broad categories of works related to CNN. We first give an overview of the basic components of CNN in Section II. Then, some recent improvements on different aspects of CNN are introduced in Section III and the fast computing techniques are introduced in Section IV. Next, we discuss some typical applications of CNN in Section V and finally we conclude this paper in Section VI.

II. Basic CNN components

There are numerous variants of CNN architectures in the literature. However, their basic components are very similar. The basic CNN architecture typically consists of three types of layers, namely convolutional layer, pooling layer and fully-connected layer. Fig. 1 shows the architecture of LeNet-5 [1] which is introduced by Yann LeCun.

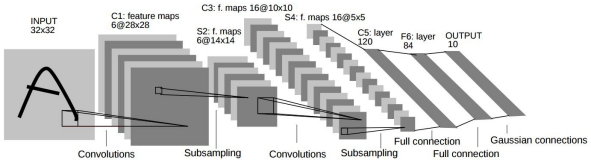


Fig. 1: The architecture of LeNet-5 network, which works well on digit classification task (Adapted from [1]).

The convolutional layer aims to learn feature representations of the inputs. As shown in Fig. 1, convolutional layer is composed of several feature maps. Each neuron of a feature map is connected to a neighborhood of neurons in the previous layer. Such a neighborhood is referred to as the neuron's receptive field in the previous layer. To compute a new feature map, the input feature maps are first convolved with a learned kernel and then the results are passed into a nonlinear activation function. By applying several different kernels, the complete new feature maps are obtained. Note that the kernel

JX. Gu, ZH. Wang, J. Kuen, LY. Ma, A. Shahroudy, B. Shuai, T. Liu, XX. Wang, G. Wang are with the Rapid-Rich Object Search (ROSE) Lab at the Nanyang Technological University, Singapore (e-mail: jxgu@ntu.edu.sg; wzh@ntu.edu.sg; jasonkuen@ntu.edu.sg; lyma@ntu.edu.sg; amir3@ntu.edu.sg; BSHUAI001@e.ntu.edu.sg; LIUT0016@e.ntu.edu.sg; wangxx@ntu.edu.sg; WangGang@ntu.edu.sg).
* equal contribution.

IPR2025-01399
CrowdStrike EX1014 Page 1

for generating a single feature map is the same. Such a share-weight mode has several advantages such as it can reduce the model complexity and make the network easier to train. The activation function introduces non-linearities to CNN, which are desirable for multi-layer networks to detect non-linear features. The typical activation functions are sigmoid, tanh and ReLU [5].

The pooling layer aims to achieve spatial invariance by reducing the resolution of the feature maps. It is usually placed between two convolutional layers. Each feature map of pooling layer is connected to its corresponding feature map of the preceding convolutional layer. Thus, they has the same number of feature maps. The typical pooling operations are average pooling [10] and max pooling [11–13]. By stacking several convolutional and pooling layers, we could extract more abstract feature representations.

After several convolutional and pooling layers, there may be one or more fully-connected layers which aim to perform high level reasoning. They take all neurons in the previous layer and connect them to every single neuron of current layer. There are no spatial information preserved in fully-connected layers.

The outputs of the last fully-connected layer will be fed to an output layer. For classification tasks, softmax regression is commonly used as it generates a well-formed probability distribution of the outputs [4]. Another commonly used method is SVM, which can be combined with CNNs to solve different classification tasks [14].

III. Improvements on CNNs

There has been various improvements on CNNs since the success of AlexNet in 2012. In this section, we describe the major improvements on CNNs from six aspects: convolutional layer, pooling layer, activation function, loss function, regularization, and optimization.

A. Convolutional layer

Convolution filter in basic CNNs is a generalized linear model (GLM) for the underlying local image patch. It works well for abstraction when instances of latent concepts are linearly separable. Here we introduce two works which aim to enhance its representation ability.

1) Network in network: Network In Network (NIN) is a general network structure proposed by Lin *et al.* [15]. It replaces the linear filter of the convolutional layer by a micro network, *e.g.*, multilayer perceptron convolution (mlpconv) layer in the paper, which makes it capable of approximating more abstract representations of the latent concepts. The overall structure of NIN is the stacking of such micro networks. To see the difference between convolutional layer and mlpconv layer, let us consider how the feature maps are computed in each of them. Formally, the feature maps of convolutional layers are computed as:

$$f_{i,j,k} = \max(w_k x_{i,j}, 0). \tag{1}$$

where i and j are the pixel indexes in the feature map, x_{ij} is the input patch centered at (i, j), and k is the channel index of the feature map. As a comparison, the computation performed by mlpconv layer is formulated as:

$$\begin{cases} f^1_{i,j,k_1} & = \max(w^1_{k_1} x_{i,j} + b_{k_1}, 0). \\ & \vdots \\ f^n_{i,j,k_n} & = \max(w^n_{k_n} f^{n-1}_{i,j} + b_{k_n}, 0). \end{cases} \tag{2}$$

where n is the number of layers in the mlpconv layer. It can be found that Eq. (2) is equivalent to cascaded cross channel parametric pooling on a normal convolutional layer.

2) Inception module: Inception module is introduced by Szegedy *et al.* [9] which can be seen as a logical culmination of NIN. [9] uses variable filter sizes to capture different visual patterns of different sizes, and approximates the optimal sparse structure by the inception module. Specifically, inception module consists of one pooling operation and three types of convolution operations. 1×1 convolutions are placed before 3×3 and 5×5 convolutions as dimension reduction modules, which allow for increasing the depth and width of CNN without increasing the computational complexity. With the help of inception module, the network parameters can be dramatically reduced to 5 millions which are much less than those of AlexNet (60 millions) and ZFNet (75 millions). In their recent paper [16], to find high performance networks with a relatively modest computation cost, they propose several design principles to scale up CNNs according to their experimental evaluation. Specifically, they suggest that: (1) One should avoid representation bottlenecks, especially early in the network. In general, the representation size should gently decrease from inputs to outputs. (2) Higher dimensional representations are easier to process locally. (3) Spatial aggregation can be done over lower dimensional embeddings without much loss in representational power. (4) Optimal performance of the network can be reached by balancing the number of filters per layer and the depth of the network.

B. Pooling layer

Pooling is an important concept of CNN. It lowers the computational burden by reducing the number of connections between convolutional layers. In this section, we introduce some recent pooling methods used in CNNs.

1) L_p Pooling: L_p pooling is a biologically inspired pooling process modelled on complex cells [17], [18]. It has been theoretically analysed in [19], [20], which suggests that L_p pooling provides better generalization than max pooling. L_p pooling can be represented as $(\sum_{i=1}^{N} |x_{I_i}|^p)^{1/p}$, where $\{x_{I_1}, ..., x_{I_N}\}$ is a finite set of input nodes. When $p = 1$, L_p reduces to average pooling, and when $p = 2$, L_p reduces to L_2 pooling. Finally, when $p = \infty$, L_p reduces to max pooling.

2) Mixed Pooling: Inspired by random Dropout [6] and DropConnect [21], Yu *et al.* [22] propose a mixed pooling method which is the combination of max pooling and average pooling. The function of mixed pooling can be formulated as follows:

$$y_{kij} = \lambda \max_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} + (1-\lambda)\frac{1}{|\mathcal{R}_{ij}|}\sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} \tag{3}$$

IPR2025-01399
CrowdStrike EX1014 Page 2

where y_{kij} is the output of the pooling operator related to position (i, j) in k-th feature map, λ is a random value being either 0 or 1 which indicates the choice of using max pooling or average pooling, $\mathcal{R}_{ij}$ is a local neighbourhood around the position (i, j), and x_{kpq} is the element at (p, q) within the pooling region $\mathcal{R}_{ij}$ in k-th feature map. During forward propagation process, λ is recorded and will be used for the backpropagation operation. Experiments in [22] show that mixed pooling can better address the overfitting performs and it performs better than max pooling and average pooling.

3) Stochastic pooling: Stochastic pooling [23] ensures that the non-maximal activations of feature maps are also possible to be utilized. Specifically, stochastic pooling first computes the probabilities p for each region R_j by normalizing the activations within the region, *i.e.*, $p_i = a_i / \sum_{k \in \mathcal{R}_j}(a_k)$. Then it samples from the multinomial distribution based on p to pick a location l within the region. The pooled activation is $s_j = a_l$, where $l \sim P(p_1, ..., p_{|\mathcal{R}_j|})$. Stochastic pooling has the advantages of max pooling, and can avoid overfitting due to the stochastic component.

4) Spectral pooling: Spectral pooling [24] performs dimensionality reduction by cropping the representation of input in frequency domain. Given an input feature map $x \in R^{M \times N}$, suppose the dimension of desired output feature map is $H \times W$, spectral pooling first computes the discrete Fourier transform (DFT) of the input feature map, then crops the frequency representation by maintaining only the central $H \times W$ submatrix of the frequencies, and finally uses inverse DFT to map the approximation back into spatial domain. Compared with max pooling, the linear low-pass filtering operation of spectral pooling can preserve more information for the same output dimensionality. Meanwhile, it also does not suffer from the sharp reduction in output map dimensionality exhibited by other pooling methods. What is more, the process of spectral pooling is achieved by matrix truncation, which makes it capable of being implemented with little computational cost in CNNs (*e.g.*, [25]) that employ FFT for convolution kernels.

5) Spatial pyramid pooling: Spatial pyramid pooling (SPP) is introduced by He *et al*. [26]. The key advantage of SPP is that it can generate a fixed-length representation regardless of the input sizes. SPP pools input feature map in local spatial bins with sizes proportional to the image size, resulting in a fixed number of bins. This is different from the sliding window pooling in the previous deep networks, where the number of sliding windows depends on the input size. By replacing the last pooling layer with SPP, they propose a new SPP-net which is able to deal with images with different sizes.

6) Multi-scale Orderless Pooling: Inspired by [27], Gong *et al*. [28] use multi-scale orderless pooling (MOP) to improve the invariance of CNNs without degrading their discriminative power. They extract deep activation features for both the whole image and local patches of several scales. The activations of the whole image is the same as those of previous CNNs, which aim to capture the global spatial layout information. The activations of local patches are aggregated by VLAD encoding [29], which aim to capture more local, fine-grained details of the image as well as enhancing invariance. The new image representation is obtained by concatenating the global activations and the VLAD features of the local patch activations.

C. Activation function

A proper activation function significantly improves the performance of a CNN for a certain task. In this section, we introduce the recent used activation functions in CNNs.

1) ReLU: Rectified linear unit (ReLU) [5] is one of the most notable non-saturated activation functions. The ReLU activation function is defined as:

$$y_i = \max(0, z_i) \tag{4}$$

where z_i is the input of i-th channel. ReLU is a piecewise linear function which prunes the negative part to zero and retains the positive part (see Fig. 2(a)). The simple $\max$ operation of ReLU allows it to compute much faster than sigmoid or tanh activation functions, and it also induces the sparsity in the hidden units and allows the network to easily obtain sparse representations. It has been shown that deep networks can be trained efficiently using ReLU even without pre-training [4]. Even though the discontinuity of ReLU at 0 may hurt the performance of backpropagation, many works have shown that ReLU works better than sigmoid and tanh activation functions empirically.

2) Leaky ReLU: A potential disadvantage of ReLU unit is that it has zero gradient whenever the unit is not active. This may cause units that do not active initially never active as the gradient-based optimization will not adjust their weights. Also, it may slow down the training process due to the constant zero gradients. To alleviate this problem, Mass *et al*. introduce leaky ReLU (LReLU) [30] which is defined as:

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ a z_i & z_i < 0 \end{cases} \tag{5}$$

where a is a predefined parameter in range $(0, 1)$. Compared with ReLU, Leaky ReLU compresses the negative part rather than mapping it to constant zero, which makes it allow for a small, non-zero gradient when the unit is not active.

3) Parametric ReLU: Rather than using a predefined parameter in Leaky ReLU, *e.g.* a in Eq.(5), He *et al*. [31] propose Parametric Rectified Linear Unit (PReLU) which adaptively learns the parameters of the rectifiers in order to improve accuracy. Mathematically, PReLU function is defined as

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ a_i z_i & z_i < 0 \end{cases} \tag{6}$$

where a_i is the learned parameters for the i-th channel. As PReLU only introduces a very small number of extra parameters, *e.g.*, the extra parameter number is the same as the channel number of the whole network, there is no extra risk of overfitting and the extra computational cost is negligible. It also can be simultaneously trained with other parameters by backpropagation.

4) Randomized ReLU: Another variant of Leaky ReLU is Randomized Leaky Rectified Linear Unit (RReLU) [32]. In RReLU, the parameters of negative parts are randomly sampled from a uniform distribution in training, and then fixed in testing (see Fig. 2(c)). Formally, RReLU function is defined as:

$$y_i^{(j)} = \begin{cases} z_i^{(j)} & z_i^{(j)} \geq 0 \\ a_{ji} z_i^{(j)} & z_i^{(j)} < 0 \end{cases} \tag{7}$$

where $z_i^{(j)}$ denotes the i-th channel in j-th example, $a_i^{(j)}$ denotes its corresponding sampled parameter, and $y_i^{(j)}$ denotes its corresponding output. It could reduce overfitting due to its randomized nature. [32] also evaluates ReLU, LReLU, PReLU and RReLU on standard image classification task, and concludes that incorporating a non-zero slop for negative part in rectified activation units could consistently improve the performance.

5) ELU: [33] introduces Exponential Linear Unit (ELU) which enables faster learning of deep neural networks and leads to higher classification accuracies. Like ReLU, LReLU, PReLU and RReLU, ELU avoids the vanishing gradient problem by setting the positive part to identity. In contrast to ReLU, ELU has a negative part which is beneficial for fast learning. Compared with LReLU, PReLU and RReLU who also have negative parts, ELU employs a saturation function as negative part which is more robust to noise. The function of ELU is defined as:

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ a(\exp(z_i) - 1) & z_i < 0 \end{cases} \tag{8}$$

where a is a predefined parameter for controlling the value to which an ELU saturate for negative inputs.

6) Maxout: Maxout [34] is an alternative non-linear function that takes the maximum response across multiple channels at each spatial position. As stated in [34], the maxout function is defined as:

$$y = \max_{i \in [1,k]} z_i \tag{9}$$

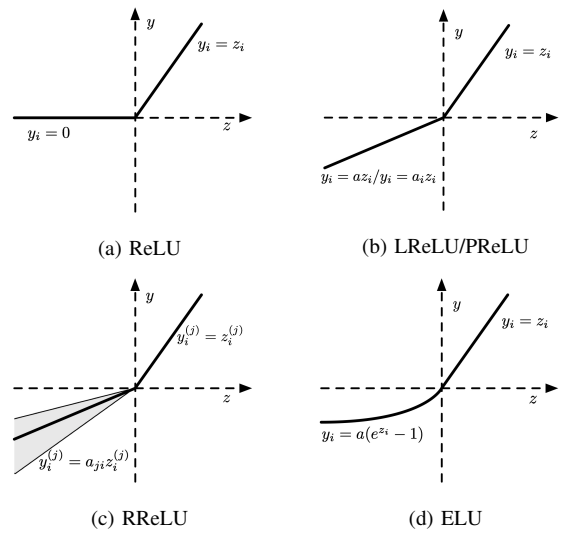
where z_i is the i-th channel of the feature map. It is worth noting that maxout enjoys all the benefits of ReLU, since ReLU is actually a special case of maxout, *e.g.*, $\max(w_1 x + b_1, w_2 x + b_2)$ where w_1 is a zero vector and b_1 is zero. Besides, maxout is particularly well suited for training with Dropout.

7) Probout: [35] proposes a probabilistic variant of maxout called probout. They replace the maximum operation in maxout with a probabilistic sampling procedure, and combine the Dropout with probout. Specifically, they first define a probability for each of the k linear units as:

$$p_i = \frac{e^{\lambda z_i}}{\sum_{j=1}^{k} e^{\lambda z_j}} \tag{10}$$

where λ is a hyperparameter for controlling the variance of the distribution. To incorporating with dropout, they actually re-define the probabilities as:

$$\hat{p}_0 = 0.5, \ \hat{p}_i = \frac{e^{\lambda z_i}}{2. \sum_{j=1}^{k} e^{\lambda z_j}} \tag{11}$$



(a) ReLU (b) LReLU/PReLU (c) RReLU (d) ELU

Fig. 2: The comparison among ReLU, LReLU, PReLU, RReLU and ELU. For Leaky ReLU, a is empirically predefined. For PReLU, a_i is learned from training data. For RReLU, a_{ji} is a random variable which is sampled from a given uniform distribution in training and keeps fixed in testing. For ELU, a is empirically predefined.

The activation function is then sampled as:

$$y_i = \begin{cases} 0 & \text{if } i = 0 \\ z_i & \text{else} \end{cases} \tag{12}$$

where $i \sim Multinomial\{\hat{p}_0, ..., \hat{p}_k\}$. Probout can achieve the balance between preserving the desirable properties of maxout units and improving their invariance properties. However, in testing process, probout is computationally expensive than maxout due to the additional probability calculations.

D. Loss function

It is important to choose an appropriate loss functions for a specific task. We introduce three representative ones in this subsection: softmax loss, hinge loss, and contrastive loss.

1) Softmax loss: Softmax loss is a commonly used loss function which is essentially a combination of multinomial logistic loss and softmax. Given a training set $\{(x^{(i)}, y^{(i)}); i \in 1, \ldots, N, y^{(i)} \in 0, \ldots, K-1\}$, where $x^{(i)}$ is the i-th input image patch, and $y^{(i)}$ is its class label. The prediction $a_j^{(i)}$ of j-th class for i-th input is transformed with the following softmax function:

$$p_j^i = e^{a_j^{(i)}} / \sum_{l=0}^{K-1} e^{a_l^{(i)}} \tag{13}$$

Softmax turns the predictions into non-negative values and normalizes them to get a probability distribution over classes. Such probabilistic predictions are used to compute the multinomial logistic loss, *i.e.*, the softmax loss, as follows:

$$\mathcal{L}_{softmax} = -\frac{1}{N} [\sum_{i=1}^{N} \sum_{j=0}^{K-1} 1\{y^{(i)} = j\} \log p_j^i] \tag{14}$$

IPR2025-01399
CrowdStrike EX1014 Page 4

2) Hinge loss: Hinge loss is usually used to train large margin classifiers such as Support Vector Machine (SVM). The hinge Loss function of a multi-class SVM is defined in Eq.(15), where x_n is the given feature vector and $\ell_n \in [0, 1, 2, ..., K-1]$ indicates its correct class label among the K classes.

$$\begin{aligned} \mathcal{L}_{Hinge} &= \frac{1}{N}\sum_{n=1}^{N}\sum_{k=0}^{K-1}[\max(0, 1-\delta(\ell_n, k)w^T x_n)]^p \\ \delta(\ell_n, k) &= \begin{cases} 1, & \text{if } \ell_n = k \\ -1, & \text{if } \ell_n \neq k \end{cases} \end{aligned} \tag{15}$$

Note that if $p = 1$, Eq.(15) is Hinge-Loss (L_1-Loss), while if $p = 2$, it is the Squared Hinge-Loss (L_2-Loss) [36]. The L_2-Loss is differentiable and imposes a larger loss for point which violates the margin comparing with L_1-Loss. [14] investigates and compares the performance of softmax with L_2-SVMs in deep networks. The results on MNIST [37] demonstrate the superiority of L_2-SVM over softmax.

3) Contrastive loss: Contrastive loss [38] is commonly used to train Siamese network. It is a weakly-supervised scheme for learning a similarity measure from pairs of data instances labelled as matching or non-matching. The contrastive loss is usually defined for every layer $l \in [1, \ldots, L]$ and the backpropagations for the loss of individual layers are performed at the same time. Given a pair of data (z_α^0, z_β^0), let (z_α^l, z_β^l) denote the output pair of layer l, the contrastive loss $\mathcal{L}_l$ for layer l is defined as :

$$\begin{aligned} \mathcal{L}_l &= (y)d_l + (1-y)\max(\mathrm{m} - d_l, 0) \\ d_l &= ||z_\alpha^l - z_\beta^l||_2^2 \end{aligned} \tag{16}$$

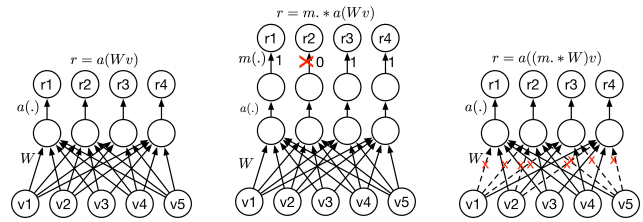
where d_l is the similarity between z_α^l and z_β^l, and m is a margin parameter affecting non-matching pairs. If (z_α^0, z_β^0) is a matching pair, then $y = 1$. Otherwise, $y = 0$. This loss function is also referred to as a single margin parameter loss function. Lin *et al*. [39] find that the recall rate quickly collapses when using this function. The reason may be that the indefinite contraction of matching pairs well beyond what is necessary to distinguish them from non-matching pairs is a damaging behaviour. To solve this problem, they propose a double margin loss function which adds another parameter to affect the matching pairs:

$$\mathcal{L}_l = (y)\max(d_l - m_1, 0) + (1-y)\max(\mathrm{m}_2 - d_l, 0) \tag{17}$$

E. Regularization

Overfitting is an unneglectable problem in deep CNNs, which can be effectively reduced by regularization. In the following subsection, we introduce some effective regularization techniques: Dropout [6], [40], [41] and DropConnect [21].

1) Dropout: Dropout is first introduced by Hinton *et al*. [6], and it has been proven to be very effective in reducing overfitting. In [6], they apply Dropout to fully-connected layers. The output of Dropout is $r = m * a(Wv)$, where $v = [v_1, v_2, \ldots, v_n]^T$ is the output of the feature extractor,



(a) No-Drop (b) DropOut (c) DropConnect

Fig. 3: The illustration of No-Drop network, DropOut network and DropConnect network.

W (of size $d \times n$) is a fully-connected weight matrix, $a(\cdot)$ is a non-linear activation function, and m is a binary mask of size d whose elements are independently drawn from a Bernoulli distribution, *i.e.* $m_i \sim Bernoulli(p)$. Dropout can prevent the network from becoming too dependent on any one (or any small combination) of neurons, and can force the network to be accurate even in the absence of certain information. Several methods have been proposed to improve Dropout. [40] proposes a fast Dropout method which can perform fast Dropout training by sampling from or integrating a Gaussian approximation. [41] proposes an adaptive Dropout method, where the Dropout probability for each hidden variable is computed using a binary belief network that shares parameters with the deep network. In [42], they find that applying standard Dropout before 1×1 convolutional layer generally increases training time but does not prevent overfitting. Therefore, they propose a new Dropout method called SpatialDropout, which extends the Dropout value across the entire feature map. This new Dropout method works well especially when the training data size is small.

2) DropConnect: DropConnect [21] takes the idea of Dropout a step further. Instead of setting the output of neurons to zero, DropConnect chooses the weights W of fully-connected layer with a probability p. The output of DropConnect is given by $r = a((m * W)v)$, where $m_{ij} \sim Bernoulli(p)$. Additionally, the biases are also masked out during the training process. Fig. 3 illustrates the differences among No-Drop, Dropout and DropConnect networks.

F. Optimization

In this subsection, we discuss some key techniques for optimizing CNNs.

1) Weights initialization: Training a deep CNN model is difficult as generally the model has a huge amount of parameters and the loss function is non-convex. To achieve a fast convergency in training, a proper initialization is one of the most important prerequisites. The bias parameters can be initialized to zero, while the weight parameters should be initialized carefully to break the symmetry among hidden units of the same layer. For example, if we simply initialize all the weights to the same value, *e.g*., zero or one, each hidden unit of the same layer will get exactly the same signal.

The most commonly used initialization method is to randomly set the weights according to Gaussian distributions [4],

[6], [8], [43]. Glorot and Bengio [44] propose a normalized initialization which sets the weights according to a distribution with zero mean and a specific variance $Var(W) = \sqrt{6}/\sqrt{n_j + n_{j+1}}$, where n_j is the size of layer j. One of its variants is called "Xavier" in Caffe [45]. He *et al*. [31] derive a robust initialization method that particularly considers the rectifier nonlinearities. Their method allows for the training of extremely deep models (*e.g*., [9]) to converge while the "Xavier" method [44] cannot. In their method, weights are initialized according to a zero-mean Gaussian distribution whose standard deviation is $\sqrt{2/n_l}$, where $n_l = k_l^2 d_{l-1}$, k_l is the spatial filter size in layer l and d_{l-1} is the number of filters in layer $l-1$.

2) Stochastic gradient descent: The backpropagation algorithm [46] is the standard training method which uses gradient descent to update the parameters. Standard gradient descent algorithm updates the parameters θ of the objective $J(\theta)$ as $\theta_{t+1} = \theta_t - \alpha \nabla_\theta E[J(\theta_t)]$, where $E[J(\theta_t)]$ is the expectation of $J(\theta)$ over the full training set and α is the learning rate. Instead of computing $E[J(\theta_t)]$, stochastic gradient descent (SGD) [47], [48] estimates the gradients on the basis of a single randomly picked example $(x^{(t)}, y^{(t)})$ from the training set:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta J(\theta_t; x^{(t)}, y^{(t)}) \tag{18}$$

In practice, each parameter update in SGD is computed with respect to a mini-batch as opposed to a single example. This could help to reduce the variance in the parameter update and can lead to more stable convergency. The convergence speed is controlled by the learning rate α_t. A common method is to use a small constant learning rate that gives stable convergence in the initial stage, and then reduce the learning rate as the convergence slows down.

Parallelized SGD methods [49–51] improve SGD to be suitable for parallel, large-scale machine learning. Unlike standard (synchronous) SGD in which the training will be delayed if one of the machines is slow, these parallelized methods use the asynchronous mechanism so that no other optimizations will be delayed except for the one on the slowest machine. Jeffrey Dean *et al*. [52] use another asynchronous SGD procedure called Downpour SGD to speed up the large-scale distributed training process on clusters with many CPUs. There are also some works that use asynchronous SGD with multiple GPUs. [53] basically combines asynchronous SGD with GPUs to accelerate the training time by several times compared to training on a single machine. [54] also uses multiple GPUs to asynchronously calculate gradients and update the global model parameters, which achieves 3.2 times of speedup on 4 GPUs compared to training on a single GPU.

3) Batch Normalization: Batch Normalization is proposed by Sergey Ioffe and Christian Szeged [55], which aims to accelerate the entire training process of deep neural networks. In [55], they suggest that the Internal Covariate Shift, *i.e*., the change in the distributions of internal nodes of a deep network, will slow down the network training. To achieve faster training, they propose an efficient method called Batch Normalization to partially alleviate this phenomenon. It accomplishes this by a normalization step that fixes the means and variances of layer inputs. In addition to accelerate the training, Batch Normalization also allows us to use much higher learning rates without the risk of divergence, and it makes it possible to use saturating nonlinearies by preventing the network from getting stuck in the saturated modes.

IV. Fast processing of CNNs

With the increasing challenges in the computer vision and machine learning tasks, the models of deep neural networks get more and more complex. These powerful models require more data for training in order to avoid overfitting. Meanwhile, the big training data also brings new challenges such as how to train the networks in a feasible amount of times. In this section, we introduce some fast processing methods of CNNs.

A. FFT

Mathieu *et al*. [25] carry out the convolutional operation in the Fourier domain with FFTs. Using FFT-based methods has many advantages. Firstly, the Fourier transformations of filters can be reused as the filters are convolved with multiple images in a mini-batch. Secondly, the Fourier transformations of the output gradients can be reused when backpropagating gradients to both filters and input images. Finally, summation over input channels can be performed in the Fourier domain, so that inverse Fourier transformations are only required once per output channel per image. There have already been some GPU-based libraries developed to speed up the training and testing process, such as cuDNN [56] and fbfft [57]. However, using FFT to perform convolution needs additional memory to store the feature maps in the Fourier domain, since the filters must be padded to be the same size as the inputs. This is especially costly when the striding parameter is larger than 1, which is common in many state-of-art networks, such as the early layers in [58] and [9].

B. Matrix Factorization

Low-rank matrix factorization has been exploited in a variety of contexts to improve the optimization problems. Given an $m \times n$ matrix A of rank r, there exists a factorization $A = B \times C$ where B is an $m \times r$ full column rank matrix and C is an $r \times n$ full row rank matrix. Thus, we can replace A by B and C. To reduce the parameters of A by a fraction p, it is essential to ensure that $mr + rn < pmn$, *i.e*., the rank of A should satisfy that $r < pmn/(m+n)$. To this end, [59] applies the low-rank matrix factorization to the final weight layer in a deep CNN, resulting about 30-50% speedup in training time at little loss in accuracy. Similarly, [60] applies singular value decomposition on each layer of a deep CNN to reduce the model size by 71% with less than 1% relative accuracy loss. Inspired by [61] which demonstrates the redundancy in the parameters of deep neural networks, Denton *et al*. [62] and Jaderberg *et al*. [63] independently investigate the redundancy within the convolutional filters and develop approximations to reduced the required computations.

C. Vector quantization

Vector Quantization (VQ) is a method for compressing densely connected layers to make CNN models smaller. Similar to scalar quantization where a large set of numbers is mapped to a smaller set [64], VQ quantizes groups of numbers together rather than addressing them one at a time. In 2013, Denil *et al*. [61] demonstrate the presence of redundancy in neural network parameters, and use VQ to significantly reduce the number of dynamic parameters in deep models. Gong *et al*. [65] investigate the information theoretical vector quantization methods for compressing the parameters of CNNs, and they obtain parameter prediction results similar to those of [61]. They also find that VQ methods have a clear gain over existing matrix factorization methods, and among the VQ methods, structured quantization methods such as product quantization work significantly better than other methods (*e.g*., residual quantization [66], scalar quantization [67]).

V. Applications of CNNs

In this section, we introduce some recent works that apply CNNs to achieve state-of-the-art performance, including image classification, object tracking, pose estimation, text detection, visual saliency detection, action recognition and scene labeling.

A. Image Classification

CNNs have been applied in image classification for a long time [68–71]. Compared with other methods, CNNs can achieve better classification accuracy on large scale datasets [4], [8], [72], [73] due to their capability of joint feature learning and classifier learning. The breakthrough of large scale image classification comes in 2012. Alex Krizhevsky *et al*. [4] develop the AlexNet and achieve the best performance in ILSVRC 2012. After the success of AlexNet, several works have made significant improvements in classification accuracy by either reducing filter size [7] or expanding the network depth [8], [9].

Building a hierarchy of classifiers is a common strategy for image classification with a large number of classes [74]. [75] is one of the earliest attempts to introduce category hierarchy in CNN, in which a discriminative transfer learning with tree-based priors is proposed. They use a hierarchy of classes for sharing information among related classes in order to improve performance for classes with very few training examples. Similar to [75], [76] builds a tree structure to learn fine-grained features for subcategory recognition. [77] proposes a training method that grows a network not only incrementally but also hierarchically. In their method, classes are grouped according to similarities and are self-organized into different levels. [78] introduces a hierarchical deep CNNs (HD-CNNs) by embedding deep CNNs into a category hierarchy. They decompose the classification task into two steps. The coarse category CNN classifier is first used to separate easy classes from each other, and then those more challenging classes are routed downstream to fine category classifiers for further prediction. This architecture follows the coarse-to-fine classification paradigm and can achieve lower error at the cost of affordable increase of complexity.

Subcategory classification is another rapidly growing subfield of image classification. There are already some fine-grained image datasets (such as Flower [79], Birds [80], [81], Dogs [82], Cars [83] and Shoes [84]). Using object part information is beneficial for fine-grained classification. Generally, the accuracy can be improved by localizing important parts of objects and representing their appearances discriminatively. Along this way, Branson *et al*. [85] propose a method which detects parts and extracts CNN features from multiple pose-normalized regions. Part annotation informations is used to learn a compact pose normalization space. They also build a model that integrates lower-level feature layers with pose-normalized extraction routines and higher-level feature layers with unaligned image features to improve the classification accuracy. Zhang *et al*. [43] propose a part-based R-CNN which can learn whole-object and part detectors. They use selective search [86] to generate the part proposals, and apply non-parametric geometric constraints to more accurately localize parts. Lin *et al*. [87] incorporate part localization, alignment, and classification into one recognition system which is called Deep LAC. Their system is composed of three sub-networks: localization sub-network is used to estimate the part location, alignment sub-network receives the location as input and performs template alignment [88], and classification sub-network takes pose aligned part images as input to predict the category label. They also propose a value linkage function to link the sub-networks and make them work as a whole in training and testing.

As can be noted, all the above mentioned methods make use of part annotation information for supervised training. However, these annotations are not easy to collect and these systems are difficult to scale up to handle many types of fine grained classes. To avoid this problem, some researchers propose to find localized parts or regions in an unsupervised manner. Krause *et al*. [89] use the ensemble of localized learned feature representations for fine-grained classification, they use co-segmentation and alignment to generate parts, and then compare the appearance of each part and aggregate the similarities together. In their latest paper [90]. They combine co-segmentation and alignment in a discriminative mixture to generate parts for facilitating fine-grained classification. [91] applies visual attention in CNN for fine-grained classification. Their classification pipeline is composed of three types of attentions: the bottom-up attention proposes candidate patches, the object-level top-down attention selects relevant patches of a certain object, and the part-level top-down attention localizes discriminative parts. These attentions are combined to train domain specific networks which can help to find foreground object or object parts and extract discriminative features. [92] proposes bilinear models for fine-grained image classification. The recognition architecture consists of two feature extractors. The outputs of two feature extractors are multiplied using outer product at each location of the image, and are pooled to obtain an image descriptor.

B. Object tracking

Object tracking has played important roles in a wide range of computer vision applications. The success in object tracking relies heavily on how robust the representation of target appearance is against several challenges such as view point changes, illumination changes, and occlusions.

CNNs have recently drawn a lot of attentions in computer vision community as well as visual tracking field. There are several attempts to employ CNNs for visual tracking. Fan *et al*. [93] use CNN as a base learner. It learns a separate class-specific network to track objects. In [93], the authors design a CNN tracker with a shift-variant architecture. Such an architecture plays a key role so that it turns the CNN model from a detector into a tracker. The features are learned during offline training. Different from traditional trackers which only extract local spatial structures, this CNN based tracking method extracts both spatial and temporal structures by considering the images of two consecutive frames. Because the large signals in the temporal information tend to occur near objects that are moving, the temporal structures provide a crude velocity signal to tracking.

Li *et al*. [94] propose a target-specific CNN for object tracking, where the CNN is trained incrementally during tracking with new examples obtained online. They employ a candidate pool of multiple CNNs as a data-driven model of different instances of the target object. Individually, each CNN maintains a specific set of kernels that favourably discriminate object patches from their surrounding background using all available low-level cues. These kernels are updated in an online manner at each frame after being trained with just one instance at the initialization of the corresponding CNN. Instead of learning one complicated and powerful CNN model for all the appearance observations in the past, [94] uses a relatively small number of filters in the CNN within a framework equipped with a temporal adaptation mechanism. Given a frame, the most promising CNNs in the pool are selected to evaluate the hypothesises for the target object. The hypothesis with the highest score is assigned as the current detection window and the selected models are retrained using a warm-start backpropagation which optimizes a structural loss function.

In [95], a CNN object tracking method is proposed to address limitations of handcrafted features and shallow classifier structures in object tracking problem. The discriminative features are first automatically learned via a CNN. To alleviate the tracker drifting problem caused by model update, the tracker exploits the ground truth appearance information of the object labeled in the initial frames and the image observations obtained online. A heuristic schema is used to judge whether updating the object appearance models or not.

Hong *et al*. [96] propose a visual tracking algorithm based on a pre-trained CNN, where the network is trained originally for large-scale image classification and the learned representation is transferred to describe target. On top of the hidden layers in the CNN, they put an additional layer of an online SVM to learn a target appearance discriminatively against background. The model learned by SVM is used to compute a target-specific saliency map by back-projecting the information relevant to target to input image space. And they exploit the target-specific saliency map to obtain generative target appearance models and perform tracking with understanding of spatial configuration of target.

C. Pose estimation

Since the breakthrough in deep structure learning, many recent works pay more attention to learn multiple levels of representations and abstractions for human-body pose estimation task with CNNs. Like other visual recognition tasks, state-of-the-art performance on the task of human-body pose estimation has achieved a significant improvement due to the large learning capacity of CNNs and the availability of more comprehensive training.

DeepPose [97] is the first application of CNNs to human pose estimation problem. In this work, pose estimation is formulated as a CNN-based regression problem to body joint coordinates. A cascade of 7-layered CNNs are presented to reason about pose in a holistic manner. Unlike the previous works that usually explicitly design graphical model and part detectors, the DeepPose ascribes to a holistic view of human pose estimation to capture the full context of each body joint by taking the whole image as the input and output the final human pose. From the experiments, DeepPose can outperform the deformable part models based methods [98–100] on two widely used datasets, FLIC [101] and LSP [102].

Except this holistic manner, some recent works exploit CNN to directly learn representation of local body parts instead of using hand-crafted low-level features. Ajrun *et al*. [103] present a CNN based end-to-end learning approach for full-body human pose estimation, in which CNN part detectors and an Markov Random Field (MRF)-like spatial model are jointly trained, and pair-wise potentials in the graph are computed using convolutional priors. In a series of papers, Tompson *et al*. [104] use a multi-resolution CNN to compute heat-map for each body part. Different from [103], Tompson *et al*. [104] learn the body part prior model and implicitly the structure of the spatial model. Specifically, They start by connecting every body part to itself and to every other body part in a pair-wise fashion, and use a fully-connected graph to model the spatial prior. As an extension of [104], Tompson *et al*. [42] propose a CNN architecture which includes a position refinement model after a rough pose estimation CNN. This refinement model, which is a Siamese network [105], is jointly trained in cascade with the off-the-shelf model [104]. In a similar work with [104], Chen *et al*. [106], [107] also combine graphical model with CNN. They exploit a CNN to learn conditional probabilities for the presence of parts and their spatial relationships, which are used in unary and pairwise terms of the graphical model. The learned conditional probabilities can be regarded as low-dimensional representations of the body pose. There is also a pose estimation method called dual-source CNN [108] that integrates graphical models and holistic style. It takes the full body image and the holistic view of the local parts as inputs to combine both local and contextual information.

In addition to still image pose estimation with CNN, recently researchers also apply CNN to human pose estimation

IPR2025-01399
CrowdStrike EX1014 Page 8

in videos. Based on the work [104], Jain *et al*. [109] also incorporate RGB features and motion features to a multi-resolution CNN architecture to further improve accuracy. Specifically, The CNN works in a sliding-window manner to perform pose estimation. The input of the CNN is a 3D tensor which consists of an RGB image and its corresponding motion features, and the output is a 3D tensor containing response-maps of the joints. In each response map, the value of each location denote the energy for presence the corresponding joint at that pixel location. The multi-resolution processing is achieved by simply down sampling the inputs and feeding them to the network.

D. Text detection and recognition

The task of recognizing text in image has been widely studied for a long time. Traditionally, optical character recognition (OCR) is the major focus. OCR techniques mainly perform text recognition on images in rather constrained visual environments (*e.g*., clean background, well-aligned text). Recently, the focus has been shifted to text recognition on scene images due to the growing trend of high-level visual understanding in computer vision research. The scene images are captured in unconstrained environments where there exists a large amount of appearance variations which poses great difficulties to existing OCR techniques. Such a concern can be mitigated by using stronger and richer feature representations such as those learned by CNN models. Along the line of improving the performance of scene text recognition with CNN, a few works have been proposed. The works can be coarsely categorized into three types: (1) text detection and localization without recognition, (2) text recognition on cropped text images, and (3) end-to-end text spotting that integrates both text detection and recognition:

1) Text detection: One of the pioneering works to apply CNN for scene text detection is [110]. The CNN model employed by [110] learns on cropped text patches and non-text scene patches to discriminate between the two. The text are then detected on the response maps generated by the CNN filters given the multiscale image pyramid of the input. To reduce the search space for text detection, [111] proposes to obtain a set of character candidates via Maximally Stable Extremal Regions (MSER) and filter the candidates by CNN classification. Another work that combines MSER and CNN for text detection is [112]. In [112], CNN is used to distinguish text-like MSER components from non-text components, and cluttered text components are split by applying CNN in a sliding window manner followed by Non-Maximal Suppression (NMS). Other than localization of text, there is an interesting work [113] that makes use of CNN to determine whether the input image contains text, without telling where the text is exactly located. In [113], text candidates are obtained using MSER which are then passed into a CNN to generate visual features, and lastly the global features of the images are constructed by aggregating the CNN features in a Bag-of-Words (BoW) framework.

2) Text recognition: [114] proposes a CNN model with multiple softmax classifiers in its final layer, which is formulated in such a way that each classifier is responsible for character prediction at each sequential location in the multi-digit input image. As an attempt to recognize text without using lexicon and dictionary, [115] introduces a novel Conditional Random Fields (CRF)-like CNN model to jointly learn character sequence prediction and bigram generation for scene text recognition. The more recent text recognition methods supplement conventional CNN models with variants of recurrent neural networks (RNN) to better model the sequence dependencies between characters in text. In [116], CNN extracts rich visual features from character-level image patches obtained via sliding window, and the sequence labelling is carried out by an enhanced RNN variant called Long Short-Term Memory (LSTM) [117]. The method presented in [118] is very similar to [116], except that in [118], lexicon can be taken into consideration to enhance text recognition performance.

3) End-to-end text spotting: For end-to-end text spotting, [10] applies a CNN model originally trained for character classification to perform text detection. Going in a similar direction as [10], the CNN model proposed in [119] enables feature sharing across the four different subtasks of an end-to-end text spotting system: text detection, character case-sensitive and insensitive classification, and bigram classification. [120] makes use of CNNs in a very comprehensive way to perform end-to-end text spotting. In [120], the major subtasks of its proposed system, namely text bounding box filtering, text bounding box regression, and text recognition are each tackled by a separate CNN model.

E. Visual saliency detection

The technique to locate important regions in imagery is referred to as visual saliency prediction. It is a challenging research topic, with a vast number of computer vision and image processing applications facilitated by it. Recently, a couple of works have been proposed to harness the strong visual modeling capability of CNNs for visual saliency prediction.

Multi-contextual information is a crucial prior in visual saliency prediction, and it has been used concurrently with CNN in most of the considered works [121–125]. [121] introduces a novel saliency detection algorithm which sequentially exploits local context and global context. The local context is handled by a CNN model which assigns a local saliency value to each pixel given the input of local image patches, while the global context (object-level information) is handled by a deep fully-connected feedforward network. In [122], the CNN parameters are shared between the global-context and local-context models, for predicting the saliency of superpixels found within object proposals. The CNN model adopted in [123] is pre-trained on large-scale image classification dataset and then shared among different contextual levels for feature extraction. The outputs of the CNN at different contextual levels are then concatenated as input to be passed into a trainable fully-connected feedforward network for saliency prediction. Similar to [122], [123], the CNN model used in [124] for saliency prediction are shared across three CNN streams, with each stream taking input of a different contextual scale. [125] derives a spatial kernel and a range kernel to

produce two meaningful sequences as 1-D CNN inputs, to describe color uniqueness and color distribution respectively. The proposed sequences are advantageous over inputs of raw image pixels because they can reduce the training complexity of CNN, while being able to encode the contextual information among superpixels.

There are also CNN-based saliency prediction approaches [126–128] that do not consider multi-contextual information. Instead, they rely very much on the powerful representation capability of CNN. In [126], an ensemble of CNNs is derived from a large number of randomly instantiated CNN models, to generate good features for saliency detection. The CNN models instantiated in [126] are however not deep enough because the maximum number of layers is capped at three. By using a pre-trained and deeper CNN model with 5 convolutional layers, [127] (Deep Gaze) learns a separate saliency model to jointly combine the responses from every CNN layer and predict saliency values. [128] is the only work making use of CNN to perform visual saliency prediction in an end-to-end manner, which means the CNN model accepts raw pixels as input and produces saliency map as output. [128] argues that the success of the proposed end-to-end method is attributed to its not-so-deep CNN architecture which attempts to prevent overfitting.

F. Action recognition

Action recognition, the behaviour analysis of human subjects and classifying their activities based on their visual appearance and motion dynamics, is one of the challenging problems in computer vision. Generally, this problem can be divided to two major groups: action analysis in still images and in videos. For both of these two groups, effective CNN based methods are proposed. In this subsection we briefly introduce the latest advances on these two groups.

1) Action Recognition in Still Images: The work of [129] has shown the output of last few layers of a trained CNN can be used as a general visual feature descriptor for a variety of tasks. The same intuition is utilized for action recognition by [8], [130], in which they use the outputs of the penultimate layer of a pre-trained CNN to represent full images of actions as well as the human bounding boxes inside them, and achieve a high level of performance in action classification. Gkioxari *et al*. [131] add a part detection to this framework. Their part detector is a CNN based extension to the original Poselet [132] method.

CNN based representation of contextual information is utilized for action recognition in [133]. They search for the most representative secondary region within a large number of object proposal regions in the image and add contextual features to the description of the primary region (ground truth bounding box of human subject) in a bottom-up manner. They utilize a CNN to represent and fine-tune the representations of the primary and the contextual regions.

2) Action Recognition in Video Sequences: Applying CNNs on videos is challenging because traditional CNNs are designed to represent two dimentional pure spatial signals but in videos a new temporal axis is added which is essentially different from the spatial variations in images. The sizes of the video signals are also in higher orders in comparison to those of images which makes it more difficult to apply convolutional networks on.

Ji *et al*. [134] propose to consider the temporal axis in a similar manner as other spatial axes and introduce a network of 3D convolutional layers to be applied on video inputs. Recently Tran *et al*. [135] study the performance, efficiency, and effectiveness of this approach and show its strengths compared to other approaches.

Another approach to apply CNNs on videos is to keep the convolutions in 2D and fuse the feature maps of consecutive frames, as proposed by [136]. They evaluate three different fusion policies: late fusion, early fusion, and slow fusion, and compare them with applying the CNN on individual single frames. One more step forward for better action recognition via CNNs is to separate the representation to spatial and temporal variations and train individual CNNs for each of them, as proposed by Simonyan and Zisserman [137]. First stream of this framework is a traditional CNN applied on all the frames and the second receives the dense optical flow of the input videos and trains another CNN which is identical to the spatial stream in size and structure. The output of the two streams are combined in a class score fusion step. Chéron *et al*. [138] utilize the two stream CNN on the localized parts of the human body and show the aggregation of part-based local CNN descriptors can effectively improve the performance of action recognition. Another approach to model the dynamics of videos differently from spatial variations, is to feed the CNN based features of individual frames to a sequence learning module *e.g*., a recurrent neural network. [139] studies different configurations of applying LSTM units as the sequence learner in this framework.

G. Scene Labeling

Scene Labeling (also termed as scene parsing, scene semantic segmentation) builds a bridge towards deeper scene understanding. The goal is to relate one semantic class (road, water, sea, *etc*.) to each pixel. Generally, "thing" pixels (car, person, *etc*.) in real world images can be quite different due to their scale, illumination and pose variation, meanwhile "stuff" pixels are very similar (road, sea, etc) in a local close-up view. Therefore, the local classification for pixels is quite challenging.

The recent advance of Convolutional Neural Networks (CNNs) [1], [4] has revolutionized the computer vision community due to their outstanding performance in a wide variety of tasks. Recently, CNNs have also been successfully applied to scene labeling tasks. In this scenario, CNNs are used to model the class likelihood of pixels directly from local image patches. They are able to learn strong features and classifiers to discriminate the local visual subtleties.

Farabet *et al*. [140] have pioneered to apply CNNs to scene labeling tasks. They feed their Multi-scale ConvNet with different scale image patches, and they show that the learned network is able to perform much better than systems with handcrafted features, which implicitly elucidate the discriminative

IPR2025-01399
CrowdStrike EX1014 Page 10

power of the features generated from CNNs. Besides, this network is also successfully applied to RGB-D scene labeling [141]. To enable the CNNs to have a large field of view over pixels, Pinheiro *et al*. [142] develop the recurrent CNNs. More specifically, the identical CNNs are applied recurrently to the output maps of CNNs in the previous iterations. By doing this, they can achieve slightly better labeling results while significantly reduces the inference times. Shuai *et al*. [143–145] train the parametric CNNs by sampling image patches, which speeds up the training time dramatically. They find that patch-based CNNs suffer from local ambiguity problems, and [143] solve it by integrating global beliefs. [144] and [145] use the recurrent neural networks to model the contextual dependencies among image features, and dramatically boost the labeling performance.

Meanwhile, researchers are exploiting to use the pre-trained deep CNNs for object semantic segmentation. Mostajabi *et al*. [146] apply the local and proximal features from a ConvNet and apply the Alex-net [4] to obtain the distant and global features, and their concatenation gives rise to the zoom-out features. They achieve very competitive results on the semantic segmentation tasks. Long *et al*. [147] train a fully convolutional Network (FCN) to directly map the input images to dense label maps. The convolution layers of the FCNs are initialized from the model pre-trained on ImageNet classification dataset, and the deconvolution layers are learned to upsample the resolution of label maps. They also achieve very promising labeling performance on semantic segmentation. Chen *et al*. [148] also apply the pre-trained deep CNNs to emit the labels of pixels. Considering that the imperfectness of boundary alignment, they further use fully connected Conditional Random Fields (CRF) to boost the labeling performance.

VI. Conclusions

Deep CNNs have made breakthroughs in processing images, video, speech and text. In this paper, we give an extensive survey on recent advances of CNNs. We have discussed the improvements of CNN on different aspects, namely, layer design, activation function, loss function, regularization, optimization and fast computation. We focus on applying CNNs in computer vision tasks, and introduce some CNN based works that achieve state-of-the-art performance, including image classification, object tracking, pose estimation, text detection, visual saliency detection, action recognition and scene labeling.

Although there has been great success of CNN in experimental evaluation, it still lacks of theoretical proof of why does it work. It is desirable to make more efforts on investigating the fundamental principles of CNNs. We hope that this paper does not only provide a better understanding of CNNs but also facilitates future research activities and application developments in the field of CNNs.

Acknowledgment

This research was carried out at the Rapid-Rich Object Search (ROSE) Lab at the Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Prime Ministers Office, Singapore, under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*. Citeseer, 1990.

[3] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *International Joint Conference on Neural Networks*, 1989, pp. 593–605.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[7] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, 2014.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[10] T. Wang, D. Wu, A. Coates, and A. Ng, "End-to-end text recognition with convolutional neural networks," in *International Conference on Pattern Recognition (ICPR)*, 2012, pp. 3304–3308.

[11] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR*, 2009.

[12] Y. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *ICML*, 2010, pp. 111–118.

[13] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *CVPR*, 2007.

[14] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[15] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2015, arXiv:1512.00567v1.

[17] E. P. Simoncelli and D. J. Heeger, "A model of neuronal responses in visual area mt," *Vision research*, vol. 38, no. 5, pp. 743–761, 1998.

[18] A. Hyvärinen and U. Köster, "Complex cell pooling and the statistics of natural images," *Network: Computation in Neural Systems*, vol. 18, no. 2, pp. 81–100, 2007.

[19] J. B. Estrach, A. Szlam, and Y. Lecun, "Signal recovery from pooling representations," in *ICML*, 2014, pp. 307–315.

[20] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 530–546.

[21] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML*, 2013, pp. 1058–1066.

[22] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Rough Sets and Knowledge Technology*. Springer, 2014, pp. 364–375.

[23] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *CoRR*, vol. abs/1301.3557, 2013.

[24] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," *arXiv preprint arXiv:1506.03767*, 2015.

[25] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *arXiv preprint arXiv:1312.5851*, 2013.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision–ECCV 2014*, 2014, pp. 346–361.

[27] S. Singh, A. Gupta, and A. Efros, "Unsupervised discovery of mid-level discriminative patches," *ECCV*, pp. 73–86, 2012.

[28] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014.

[29] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *PAMI*, vol. 34, no. 9, pp. 1704–1716, 2012.

[30] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv preprint arXiv:1502.01852*, 2015.

[32] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[33] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[34] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[35] J. T. Springenberg and M. Riedmiller, "Improving deep neural networks with probabilistic maxout units," *arXiv preprint arXiv:1312.6116*, 2013.

[36] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 116.

[37] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[38] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005.

[39] J. Lin, O. Morere, V. Chandrasekhar, A. Veillard, and H. Goh, "Deephash: Getting regularization, depth and fine-tuning right," *arXiv preprint arXiv:1501.04711*, 2015.

[40] S. Wang and C. Manning, "Fast dropout training," in *ICML*, 2013.

[41] J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *Advances in Neural Information Processing Systems*, 2013, pp. 3084–3092.

[42] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," *arXiv preprint arXiv:1411.4280*, 2014.

[43] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based r-cnns for fine-grained category detection," in *ECCV*, 2014, pp. 834–849.

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[45] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[46] G. B. Orr and K.-R. Müller, *Neural networks: tricks of the trade*, 2003.

[47] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT*, 2010, pp. 177–186.

[48] R. G. J. Wijnhoven and P. H. N. de With, "Fast training of object detection using stochastic gradient descent," in *ICPR*, 2010, pp. 424–427.

[49] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[50] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.

[51] Y. Bengio, "Deep learning of representations: Looking forward," in *Statistical Language and Speech Processing*. Springer, 2013, pp. 1–37.

[52] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1223–1231.

[53] T. Paine, H. Jin, J. Yang, Z. Lin, and T. Huang, "Gpu asynchronous stochastic gradient descent to speed up neural network training," *arXiv preprint arXiv:1312.6186*, 2013.

[54] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, "A fast parallel sgd for matrix factorization in shared memory systems," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 249–256.

[55] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[56] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[57] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A gpu performance evaluation," *arXiv preprint arXiv:1412.7580*, 2014.

[58] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013.

[59] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *ICASSP*, 2013.

[60] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition." in *INTERSPEECH*, 2013, pp. 2365–2369.

[61] M. Denil, B. Shakibi, L. Dinh, N. de Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.

[62] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 1269–1277.

[63] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[64] G. J. Sullivan, "Efficient scalar quantization of exponential and laplacian random variables," *Information Theory, IEEE Transactions on*, vol. 42, no. 5, pp. 1365–1374, 1996.

[65] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[66] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11 259–11 273, 2010.

[67] R. Balasubramanian, C. A. Bouman, and J. P. Allebach, "Sequential scalar quantization of color images," *Journal of Electronic Imaging*, vol. 3, no. 1, pp. 45–59, 1994.

[68] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *Neural Networks, IEEE Transactions on*, vol. 8, no. 1, pp. 98–113, 1997.

[69] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *null*. IEEE, 2003, p. 958.

[70] F. J. Huang and Y. LeCun, "Large-scale learning with svm and convolutional for generic object categorization," in *CVPR*, 2006, pp. 284–291.

[71] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1237.

[72] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[73] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, vol. 111, no. 1, pp. 98–136, 2014.

[74] A.-M. Tousch, S. Herbin, and J.-Y. Audibert, "Semantic hierarchies for image annotation: A survey," *Pattern Recognition*, vol. 45, no. 1, pp. 333–345, 2012.

[75] N. Srivastava and R. R. Salakhutdinov, "Discriminative transfer learning with tree-based priors," in *Advances in Neural Information Processing Systems*, 2013, pp. 2094–2102.

[76] Z. Wang, X. Wang, and G. Wang, "Learning fine-grained features via a cnn tree for large-scale classification," *arXiv preprint arXiv:1511.04534*, 2015.

[77] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 177–186.

[78] Z. Yan, V. Jagadeesh, D. DeCoste, W. Di, and R. Piramuthu, "Hd-cnn: Hierarchical deep convolutional neural network for image classification," *arXiv preprint arXiv:1410.0736*, 2014.

[79] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *ICVGIP*, 2008, pp. 722–729.

[80] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[81] T. Berg, J. Liu, S. W. Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, "Birdsnap: Large-scale fine-grained visual categorization of birds," in *CVPR*, 2014, pp. 2019–2026.

[82] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *CVPR*, 2011.

[83] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *CVPR*, 2015.

[84] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *CVPR*, 2014, pp. 192–199.

[85] S. Branson, G. Van Horn, P. Perona, and S. Belongie, "Improved bird species recognition using pose normalized deep convolutional nets," in *British Machine Vision Conference*, 2014.

[86] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[87] D. Lin, X. Shen, C. Lu, and J. Jia, "Deep lac: Deep localization, alignment and classification for fine-grained recognition," in *CVPR*, 2015, pp. 1666–1674.

[88] J. P. Pluim, J. A. Maintz, M. Viergever *et al.*, "Mutual-information-based registration of medical images: a survey," *Medical Imaging, IEEE Transactions on*, vol. 22, no. 8, pp. 986–1004, 2003.

[89] J. Krause, T. Gebru, J. Deng, L.-J. Li, and L. Fei-Fei, "Learning features and parts for fine-grained recognition," in *ICPR*, 2014, pp. 26–33.

[90] J. Krause, H. Jin, J. Yang, and L. Fei-Fei, "Fine-grained recognition without part annotations," in *CVPR*, 2015, pp. 5546–5555.

[91] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang, "The application of two-level attention models in deep convolutional neural network for fine-grained image classification," in *CVPR*, 2015.

[92] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," *arXiv preprint arXiv:1504.07889*, 2015.

[93] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *Neural Networks, IEEE Transactions on*, vol. 21, no. 10, pp. 1610–1623, 2010.

[94] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking," in *Proceedings of the British Machine Vision Conference*, 2014.

[95] Y. Chen, X. Yang, B. Zhong, S. Pan, D. Chen, and H. Zhang, "Cntracker: Online discriminative object tracking via deep convolutional neural network," *Applied Soft Computing*, 2015.

[96] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," *arXiv preprint arXiv:1502.06796*, 2015.

[97] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *CVPR*, 2014, pp. 1653–1660.

[98] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *CVPR*. IEEE, 2011, pp. 1385–1392.

[99] F. Wang and Y. Li, "Beyond physical connections: Tree models in human pose estimation," in *CVPR*, 2013, pp. 596–603.

[100] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Poselet conditioned pictorial structures," in *CVPR*, 2013, pp. 588–595.

[101] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *CVPR*, 2013, pp. 3674–3681.

[102] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation." in *BMVC*, 2010, p. 5.

[103] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler, "Learning human pose estimation features with convolutional networks," April 2014.

[104] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in Neural Information Processing Systems*, 2014, pp. 1799–1807.

[105] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.

[106] X. Chen and A. L. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in *Advances in Neural Information Processing Systems*, 2014, pp. 1736–1744.

[107] X. Chen and A. Yuille, "Parsing occluded people by flexible compositions," in *CVPR*, 2015.

[108] X. Fan, K. Zheng, Y. Lin, and S. Wang, "Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation," in *CVPR*, 2015.

[109] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "Modeep: A deep learning framework using motion features for human pose estimation," in *ACCV*, 2014, pp. 302–315.

[110] M. Delakis and C. Garcia, "Text detection with convolutional neural networks," in *VISAPP*, 2008.

[111] H. Xu and F. Su, "Robust seed localization and growing with deep convolutional features for scene text detection," in *ICMR*, 2015, pp. 387–394.

[112] W. Huang, Y. Qiao, and X. Tang, "Robust scene text detection with convolution neural network induced mser trees," in *ECCV*, 2014.

[113] C. Zhang, C. Yao, B. Shi, and X. Bai, "Automatic discrimination of text and non-text natural images," in *ICDAR*, 2015.

[114] I. J. Goodfellow, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," in *ICLR*, 2014.

[115] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," in *ICLR*, 2015.

[116] P. He, W. Huang, Y. Qiao, C. C. Loy, and X. Tang, "Reading scene text in deep convolutional sequences," *CoRR*, vol. abs/1506.04395, 2015.

[117] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[118] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *CoRR*, vol. abs/1507.05717, 2015.

[119] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *ECCV*, 2014, pp. 512–528.

[120] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *IJCV*, pp. 1–20, 2015.

[121] L. Wang, H. Lu, X. Ruan, and M.-H. Yang, "Deep networks for saliency detection via local estimation and global search," in *CVPR*, 2015.

[122] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *CVPR*, June 2015.

[123] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *CVPR*, June 2015.

[124] N. Liu, J. Han, D. Zhang, S. Wen, and T. Liu, "Predicting eye fixations using convolutional neural networks," in *CVPR*, 2015.

[125] S. He, R. W. Lau, W. Liu, Z. Huang, and Q. Yang, "Supercnn: A superpixelwise convolutional neural network for salient object detection," *IJCV*, pp. 1–15, 2015.

[126] E. Vig, M. Dorr, and D. Cox, "Large-scale optimization of hierarchical features for saliency prediction in natural images," in *CVPR*, 2014.

[127] M. Kmmerer, L. Theis, and M. Bethge, "Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet," in *ICLR*, 2015.

[128] J. Pan and X. Gir-i Nieto, "End-to-end convolutional network for saliency prediction," in *CVPR*, 2015.

[129] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," 2014.

[130] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *CVPR*, 2014, pp. 1717–1724.

[131] G. Gkioxari, R. Girshick, and J. Malik, "Actions and attributes from wholes and parts," 2014.

[132] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Poselet conditioned pictorial structures," in *CVPR*, 2013, pp. 588–595.

[133] G. Gkioxari, R. B. Girshick, and J. Malik, "Contextual action recognition with r*cnn," *CoRR*, vol. abs/1505.01197, 2015.

[134] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *PAMI*, vol. 35, no. 1, pp. 221–231, 2013.

[135] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," *CoRR*, vol. abs/1412.0767, 2014.

[136] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014, pp. 1725–1732.

[137] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., 2014, pp. 568–576.

[138] G. Chéron, I. Laptev, and C. Schmid, "P-CNN: pose-based CNN features for action recognition," *CoRR*, vol. abs/1506.03607, 2015.

[139] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," June 2015.
[140] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *PAMI*, pp. 1915–1929, 2013.
[141] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv:1301.3572*, 2013.
[142] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *Proceedings of The 31st International Conference on Machine Learning*, 2014, pp. 82–90.
[143] B. Shuai, G. Wang, Z. Zuo, B. Wang, and L. Zhao, "Integrating parametric and non-parametric models for scene labeling," in *CVPR*, 2015, pp. 4249–4258.
[144] B. Shuai, Z. Zuo, and W. Gang, "Quaddirectional 2d-recurrent neural networks for image labeling," *Signal Processing Letters*, 2015.
[145] B. Shuai, Z. Zuo, G. Wang, and B. Wang, "Dag-recurrent neural networks for scene labeling," *arXiv preprint arXiv:1509.00552*, 2015.
[146] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," in *CVPR*, 2015.
[147] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.
[148] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.