REVIEW Communicated by Vincent Vanhoucke

Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review

Waseem Rawat
wrawat10@gmail.com
Zenghui Wang
wangz@unisa.ac.za
Department of Electrical and Mining Engineering, University of South Africa, Florida 1710, South Africa

Convolutional neural networks (CNNs) have been applied to visual tasks since the late 1980s. However, despite a few scattered applications, they were dormant until the mid-2000s when developments in computing power and the advent of large amounts of labeled data, supplemented by improved algorithms, contributed to their advancement and brought them to the forefront of a neural network renaissance that has seen rapid progression since 2012. In this review, which focuses on the application of CNNs to image classification tasks, we cover their development, from their predecessors up to recent state-of-the-art deep learning systems. Along the way, we analyze (1) their early successes, (2) their role in the deep learning renaissance, (3) selected symbolic works that have contributed to their recent popularity, and (4) several improvement attempts by reviewing contributions and challenges of over 300 publications. We also introduce some of their current trends and remaining challenges.

1 Introduction

Image classification, which can be defined as the task of categorizing images into one of several predefined classes, is a fundamental problem in computer vision. It forms the basis for other computer vision tasks such as localization, detection, and segmentation (Karpathy, 2016). Although the task can be considered second nature for humans, it is much more challenging for an automated system. Some of the complications encountered include viewpoint-dependent object variability and the high in-class variability of having many object types (Ciresan, Meier, Masci, Gambardella, & Schmidhuber, 2011). Traditionally, a dual-stage approach was used to solve the classification problem. Handcrafted features were first extracted from images using feature descriptors, and these served as input to a trainable classifier. The major hindrance of this approach was that the accuracy of the classification task was profoundly dependent on the design of the feature

Neural Computation **29, 2352–2449** (2017)
doi:10.1162/NECO_a_00990
© 2017 Massachusetts Institute of Technology

Open Rubric

Samsara EX1015
U.S. Patent No. 11,875,580

extraction stage, and this usually proved to be a formidable task (LeCun, Bottou, Bengio, & Haffner, 1998).

In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extraction and transformation as well as for pattern analysis and classification, have been shown to overcome these challenges. Among them, CNNs (LeCun, Boser, Denker, Henderson, Hubbard, & Jackel, 1989a, 1989b) have become the leading architecture for most image recognition, classification, and detection tasks (LeCun, Bengio, & Hinton, 2015). Despite some early successes (LeCun et al., 1989a, 1989b; LeCun et al. 1998; Simard, Steinkraus, & Platt 2003), deep CNNs (DCNNs) were brought into the limelight as a result of the deep learning renaissance (Hinton, Osindero, & Teh, 2006; Hinton & Salakhutdinov, 2006; Bengio, Lamblin, Popovici, & Larochelle, 2006), which was fueled by GPUs, larger data sets, and better algorithms (Krizhevsky, Sutskever, & Hinton, 2012; Deng & Yu, 2014; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014). Several advances such as the first GPU implementation (Chellapilla, Puri, & Simard, 2006) and the first application of maximum pooling (max pooling) for DCNNs (Ranzato, Huang, Boureau, & LeCun, 2007) have all contributed to their recent popularity.

The most significant advance, which has captured intense interest in DCNNs, especially for image classification tasks, was achieved in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 (Russakovsky et al., 2015), when the winning entry, by Krizhevsky et al. (2012), used a DCNN to classify approximately 1.2 million images into 1000 classes, with record-breaking results. Since then, DCNNs have dominated subsequent versions of the ILSVRC and, more specifically, its image classification component (Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015).

In addition, selected representative examples of other improvement attempts related to the following different aspects of DCNNs—(1) network architecture (Lin, Chen, & Yan, 2013; Zeiler & Fergus, 2013; Gong, Wang, Guo, & Lazebnik, 2014; Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015); (2) nonlinear activation functions (He, Zhang, Ren, & Sun, 2015a; Xu, Wang, Chen, & Li, 2015); (3) supervision components (Tang, 2013; Zhao & Griffin, 2016); (4) regularization mechanisms (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012; Zeiler & Fergus, 2013); and (5) optimization techniques (Glorot & Bengio, 2010; Krizhevsky et al., 2012)—have also been implemented in recent years. Moreover, some of their open challenges, like their variance to geometric distortions (Gong, Wang, et al., 2014), the fact that their models are often large and slow to compute (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014), and the intriguing discovery of adversarial examples (Szegedy et al., 2014), have led to even more research focusing on image classification with DCNNs.

Previously, several generic deep learning reviews (Bengio, 2009; Schmidhuber, 2015; Deng, 2014; LeCun et al., 2015), reviews that deal with deep

learning for visual understanding (Guo et al., 2016), reviews covering recent advances in CNNs (Gu et al., 2015), and a taxonomy of DCNNs for computer vision tasks (Srinivas et al., 2016) have been published. However, given the surge in the popularity of DCNNs for image classification tasks and the subsequent plethora of related papers, we feel the time is right to review them for this specific and momentous problem. With this in mind, this review is intended for those who want to understand the development of CNN technology and architecture, specifically for image classification, from their predecessors up to modern state-of-the-art deep learning systems. It also asserts brief insights into their future and provides several interesting imminent directions making it suitable for researchers in the field.

The remainder of this review is organized as follows: Section 2 briefly introduces CNNs and acquaints readers with the key building blocks of their architecture. Section 3 covers the early development of CNNs. Among other highlights, it briefly touches on the first applications of backpropagation and max pooling, as well as the introduction of the famous MNIST data set (LeCun et al., 1998). In section 4, we deal with the role of DCNNs in the deep learning renaissance, and this is followed by discussions on selected representative works that have contributed to their popularity for image classification tasks. Section 5 deals with several DCNN improvement attempts in various aspects, including network architecture, nonlinear activation functions, supervision components, regularization mechanisms, optimization techniques, and computational cost developments. Section 6 concludes the review by introducing some of the remaining challenges and current trends.

2 Overview of CNN architecture

CNNs are feedforward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells (Hubel & Wiesel, 1959, 1962), motivates their architecture. CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feedforward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model. Figure 1 illustrates typical CNN architecture for a toy image classification task. An image is input directly to the network, and this is followed by several stages of convolution and pooling. Thereafter, representations from these operations feed one or more fully connected layers. Finally, the last fully connected layer outputs the class label. Despite this being the most popular base architecture found in the literature, several architecture changes have been proposed in recent years with the objective of improving image classification accuracy or

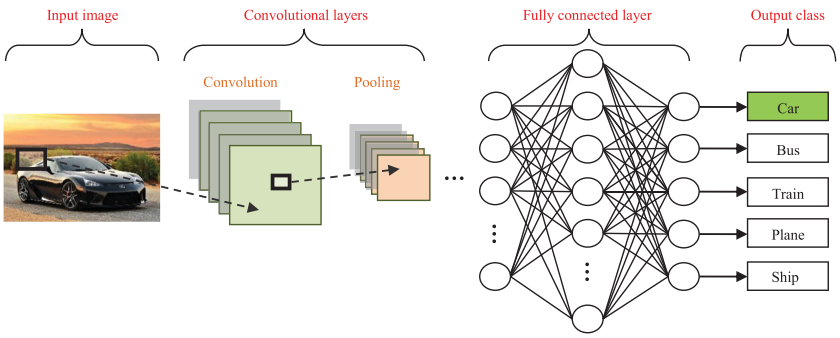

Figure 1: CNN image classification pipeline.

reducing computation costs. Although for the remainder of this section, we merely fleetingly introduce standard CNN architecture, in section 5 we deal with several architectural design changes that have facilitated enhanced image classification performance.

2.1 Convolutional Layers. The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter bank (LeCun et al., 2015). Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function. All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location (LeCun et al., 1998; LeCun et al., 2015). More formally, the kth output feature map Y_k can be computed as

$$Y_k = f(W_k * x) \tag{2.1}$$

where the input image is denoted by x; the convolutional filter related to the kth feature map is denoted by W_k; the multiplication sign in this context refers to the 2D convolutional operator, which is used to calculate the inner product of the filter model at each location of the input image; and $f(\cdot)$ represents the nonlinear activation function (Yu, Wang, Chen, & Wei, 2014). Nonlinear activation functions allow for the extraction of nonlinear features. Traditionally, the sigmoid and hyperbolic tangent functions were used; recently, rectified linear units (ReLUs; Nair & Hinton, 2010) have become popular (LeCun et al., 2015). Their popularity and success have

opened up an area of research that focuses on the development and application of novel DCNN activation functions to improve several characteristics of DCNN performance. Thus, in section 5.2, we formally introduce the ReLU and discuss the motivations that led to their development, before elaborating on the performance of several rectification-based and alternative activation functions.

2.2 Pooling Layers. The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations (LeCun et al., 1989a, 1989b; LeCun et al., 1998, 2015; Ranzato et al., 2007). Initially, it was common practice to use average pooling aggregation layers to propagate the average of all the input values, of a small neighborhood of an image to the next layer (LeCun et al., 1989a, 1989b; LeCun et al., 1998). However, in more recent models (Ciresan et al., 2011; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015; Xu et al., 2015), max pooling aggregation layers propagate the maximum value within a receptive field to the next layer (Ranzato et al., 2007). Formally, max pooling selects the largest element within each receptive field such that

$$Y_{kij} = \max_{(p,q)\in\Re_{ij}} x_{kpq}, \tag{2.2}$$

where the output of the pooling operation, associated with the kth feature map, is denoted by Y_{kij}, x_{kpq} denotes the element at location (p, q) contained by the pooling region $\Re_{ij}$, which embodies a receptive field around the position (i, j) (Yu et al., 2014). Figure 2 illustrates the difference between max pooling and average pooling. Given an input image of size 4×4, if a 2×2 filter and stride of two is applied, max pooling outputs the maximum value of each 2×2 region, while average pooling outputs the average rounded integer value of each subsampled region. While the motivations behind the migration toward max pooling are addressed in section 4.2.3, there are also several concerns with max pooling, which have led to the development of other pooling schemes. These are introduced in section 5.1.2.

2.3 Fully Connected Layers. Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and perform the function of high-level reasoning (Hinton et al., 2012; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014). For classification problems, it is standard to use the softmax operator (see sections 5.3.1 and 5.3.5) on top of a DCNN (Krizhevsky et al., 2012; Lin et al., 2013; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015; Xu et al., 2015). While early

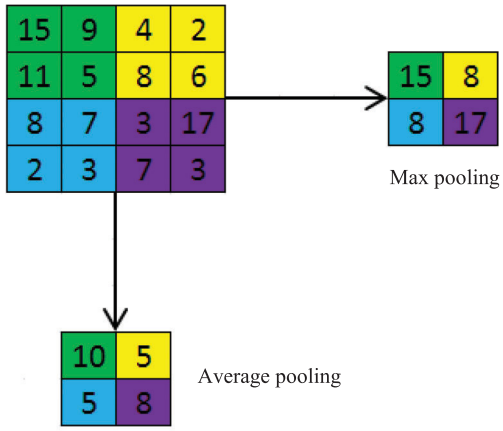

Figure 2: Average versus max pooling.

success was enjoyed by using radial basis functions (RBFs), as the classifier on top of the convolutional towers (LeCun et al., 1998), Tang (2013) found that replacing the softmax operator with a support vector machine (SVM) leads to improved classification accuracy (see section 5.3.4 for further details). Moreover, given that computation in the fully connected layers is often challenged by their compute-to-data ratio, a global average-pooling layer (see section 5.1.1.1 for further details), which feeds into a simple linear classifier, can be used as an alternative (Lin et al. 2013). Notwithstanding these attempts, comparing the performance of different classifiers on top of DCNNs still requires further investigation and thus makes for an interesting research direction (see section 6 for other intrinsic DCNN trends).

2.4 Training. CNNs, and ANNs in general use learning algorithms to adjust their free parameters (i.e., the biases and weights) in order to attain the desired network output. The most common algorithm used for this purpose is backpropagation (LeCun, 1989; LeCun et al., 1998; Bengio, 2009; Deng & Yu, 2014; Deng, 2014; Srinivas et al., 2016). Backpropagation computes the gradient of an objective (also referred to as a cost/loss/performance) function to determine how to adjust a network's parameters in order to minimize errors that affect performance. A commonly experienced problem with training CNNs, and in particular DCNNs, is overfitting, which is poor performance on a held-out test set after the network is trained on a small or even large training set. This affects the model's ability to generalize on unseen data and is a major challenge for DCNNs that can be assuaged by regularization, which is surveyed in section 5.4.

2.5 Discussion. This section briefly highlighted some of the fundamental aspects related to the basic building blocks of CNNs. Further detailed explanations on the convolution function and its variants and the convolutional and pooling layers, can be found in Goodfellow, Bengio, and Courville (2016). Furthermore, for convolutional and pooling arithmetic, reader's are referred to Dumoulin and Visin (2016). Detailed explanations on the backpropagtion algorithm and general training protocols for deep neural networks (DNNs) are available in LeCun et al. (1998) and Goodfellow et al. (2016), while LeCun et al. (2015) provides a concise summary of the algorithm and supervised learning (one of the major machine learning paradigms, together with unsupervised and reinforcement learning) in general. A brief history on the development of this popular algorithm, specifically for CNNs, is provided in section 3.2. Finally, some of the DCNN theoretical considerations, many of which are concisely summarized by Koushik (2016), are introduced in section 6.1.

3 Early CNN Development

In this section, we cover the early developments and significant advancements of CNNs, from their predecessors up to successful applications prior to the deep learning renaissance (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio, Lamblin, Popovici, & Larochelle, 2006).

3.1 CNN Predecessors Inspired by Neuroscience. Biology has inspired several artificial intelligence techniques such as ANNs, evolutionary algorithms, and cellular automata (Floreano & Mattiussi, 2008). However, perhaps the greatest success story among them are CNNs (Goodfellow et al., 2016). Their history began with the neurobiological experiments conducted by Hubel and Wiesel (1959, 1962) from as early as 1959. The main contribution of their work was the discovery that neurons in different stages of the visual system, responded strongly to specific stimulus patterns while ignoring others. More specifically, they found that neurons in the early stages of the primary visual cortex responded strongly to precisely oriented patterns of light, such as bars, but ignored more complex patterns of the input stimulus that resulted in strong responses from neurons in later stages. They also found that the visual cortex consisted of simple cells, which had local receptive fields, and complex cells, which were invariant to shifted or distorted inputs, arranged in a hierarchical fashion. These works provided the early inspiration to model our automated vision systems based on characteristics of the central nervous system.

In 1979, a novel multilayered neural network model, nicknamed the neocognitron, was proposed (Fukushima, 1979). Modeled based on the findings of Hubel and Wiesel (1959, 1962), it also consisted of simple and complex cells, cascaded together in a hierarchical manner. With this architecture, the network proved successful at recognizing simple input patterns

irrespective of a shift in the position or considerable distortion in the shape of the input pattern (Fukushima, 1980; Fukushima & Miyake, 1982). Significantly, the neocognitron laid the groundwork for the development of CNNs. In fact, CNNs were derived from the neocognitron, and hence they have a similar architecture (LeCun et al., 2015).

3.2 Brief History of Backpropagation and the First Application to CNNs. Backpropagation was derived in the 1960s. In particular, S. E. Dreyfus (1962) derived a simplified version of the algorithm that used the chain rule alone. Nevertheless, the early versions of backpropagation were inefficient since they backpropagated derivative information from one layer to the preceding layer without openly addressing direct links across layers. Furthermore, they did not consider potential efficiency gains due to network sparseness (Schmidhuber, 2015). The modern efficient form of the algorithm that addressed these issues was derived in 1970 (Linnainmaa, 1970); however, there was no mention of its use for ANNs. Preliminary discussions for its use for ANNs date back to 1974 (Werbos, 1974); however, the first known application of efficient backpropagation, specifically for ANNs, was described in 1981 (Werbos, 1982), but this remained relatively unknown. Nevertheless, it was "significantly popularized" (Schmidhuber, 2015) due to a seminal paper in 1986 by D. E. Rumelhart et al. (1986), which demonstrated that by using the backpropagation learning algorithm, the internal hidden neurons of an ANN could be trained to represent important features of the task domain.

In 1989, LeCun et al. (1989a, 1989b) proposed the first multilayered CNNs and successfully applied these large-scale networks, to real (handwritten digits and zip codes) image classification problems. These initial CNNs were reminiscent of the neocognitron (Fukushima, 1979, 1980; Fukushima & Miyake, 1982). However, the key difference was that they were trained in a fully supervised fashion using backpropagation, which was in contrast to the unsupervised reinforcement scheme used by their predecessor. This allowed them to rely more profoundly on automatic learning rather than hand-designed preprocessing for feature extraction (LeCun et al., 1989a, 1989b; LeCun, 1989), which previously proved to be extremely challenging; hence, they form an essential component of many recent competition-winning DCNNs (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015).

3.3 Introduction of the MNIST Data Set. In 1998, the CNNs described earlier (LeCun et al., 1989a, 1989b), were improved on and used for the task of individual character classification in a document recognition application. This work was published in a detailed seminal paper (LeCun et al., 1998) that highlighted the main advantages of CNNs when compared to traditional ANNs: they require fewer free parameters (because of weight sharing), and they consider the spatial topology of the input data, thereby

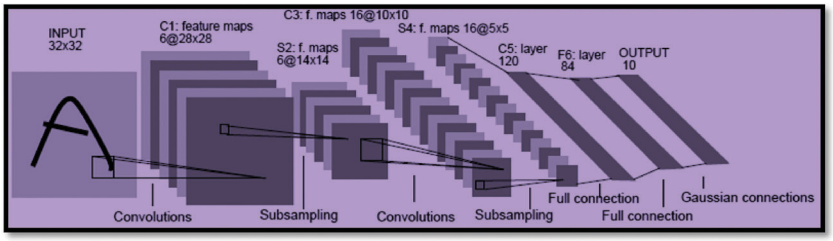

Figure 3: Architecture of LeNet-5 (LeCun et al., 1998).

allowing them to deal with the variability of 2D shapes. In addition to the proposed CNNs, LeCun et al. (1998) introduced the popular Modified National Institute of Standards and Technology (MNIST) data set of 70,000 handwritten digits, which has since been used extensively for several computer vision tasks and, in particular, for image classification and recognition problems. Figure 3 illustrates the architecture of the CNN, called LeNet-5, proposed by LeCun et al. (1998). The diagram clearly illustrates the design of LeNet-5, which consists of alternating convolutional and subsampling layers, followed by a single fully connected layer.

3.4 Early CNN Successes Despite Perceived Issues with Gradient Descent. In the late 1990s and early 2000s, neural network research had diminished (Simard et al., 2003; LeCun et al., 2015). It was little used for machine learning tasks, and computer vision and speech recognition tasks overlooked them. It was widely believed that learning useful multistage feature extractors, with little prior knowledge, was infeasible due to issues with the popular optimization algorithm, gradient descent. Specifically, it was thought that basic gradient descent would not recover from poor weight configurations that inhibited the reduction of the average backpropagated error, a phenomenon known as poor local minima (LeCun et al., 2015). In contrast, other statistical methods and, in particular, SVMs, became popular due to their successes (Decoste & Schölkopf, 2002). Contrary to this trend, a CNN was proposed for the application of visual document analysis in 2003 (Simard et al., 2003).

At a time when CNNs were not popular in the engineering community, Simard et al. (2003) were able to achieve the best-known classification result on the MNIST data set (LeCun et al., 1998), improving on the previous best results obtained by the SVMs of Decoste and Schölkopf (2002). Citing the advantages that were mentioned by LeCun et al. (1998), utilizing CNNs for visual tasks, they expanded the size and quality of the MNIST data set and proposed the use of simple software loops for the convolutional operation. These loops exploited the property of backpropagation that allows an ANN to be expressed in a modular fashion, and this allowed

for modular software debugging. Although LeCun et al. (1998) had already hypostasized and proved that by increasing the size of the data set, using artificially generated affine transformations, the network's performance will improve, Simard et al. (2003) improved the quality of the increased portion of the data set to further improve performance. This was accomplished by using elastic image deformations. This work formed part of a series of several optical character recognition applications that used CNNs. In particular, Microsoft used them for English handwritten digits (Simard et al., 2003; Chellapilla, Shilman, & Simard, 2006), Arabic handwriting recognition (Abdulkader, 2006) and East Asian handwritten character recognition (Chellapilla & Simard, 2006). Thus, these applications, together with the work described by LeCun et al. (1989a, 1989b, 1998), represent some of the early image classification successes enjoyed by CNNs. The background to the next section highlights several other successes.

4 The Deep Learning Renaissance and the Rise of DCNNs

This section briefly introduces the deep learning renaissance and focuses on the significant contributions of DCNNs to the current surge in deep learning research. It also covers a seminal paper and several representative works that have led to their recent ascendancy over other image classification techniques.

4.1 Background to the Deep Learning Renaissance. The first feedforward multilayered neural networks were trained in 1965 (Ivakhnenko & Lapa, 1966), and although they did not use backpropagation, they were perhaps the first deep learning systems (Schmidhuber, 2015). Although deep learning–like algorithms have a long history, the term *deep learning* became a catchphrase around 2006, when deep belief networks (DBNs) and autoencoders trained in an unsupervised fashion were used to initialize DNNs, trained using backpropagation (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2006). Prior to this, it was taught that deep multilayered networks (including DCNNs) were too hard to train due to issues with gradient descent and thus were not popular (Bengio et al., 2006; Bengio, 2009; Deng & Yu, 2014; Schmidhuber, 2015; Goodfellow et al., 2016). Conversely, CNNs were a notable exception and proved easier to train when compared to fully connected networks (Simard et al., 2003, Bengio, 2009; LeCun et al., 2015; Goodfellow et al., 2016). In addition to the successes discussed in section 3.3, some of the other successful applications that incorporated CNNs for their image classification component prior to the resurgence of neural networks in 2006 include medical image segmentation (Ning et al., 2005); facial recognition, detection, and verification (Lawrence, Giles, Tsoi, & Back, 1997; Garcia & Delakis, 2002; Chopra, Hadsell, & LeCun, 2005); off-road obstacle avoidance (Muller, Ben, Cosatto, Flepp, &

LeCun, 2005); and generic object classification (LeCun, Huang, & Bottou, 2004; Huang & LeCun, 2006).

However, since neural network research had slowed in the late 1990s and early 2000s (Simard et al., 2003; LeCun et al., 2015), CNN development was also hindered, but it revived around 2006. Using an energy-based model to extract sparse features, which has several applications that include classification and segmentation, and then using the resultant output to initialize the first layer of a DCNN, Ranzato, Poultney, Chopra, and LeCun (2006) slightly improved the previous best-reported classification result (Simard et al., 2003) on the MNIST data set (LeCun et al., 1998). Citing Hinton et al. (2006), their DCNN model, which had a similar architecture to that of LeCun et al. (1998) but used a considerably larger number of feature maps to produce sparse features, was pretrained in an unsupervised fashion and consisted of three essential components. An encoder interrogated the input image and computed a code vector of the image, which was then transformed into a sparse code vector by a nonlinear-sparsifying logistic module. A decoder that computed a restored version of the input image deciphered the sparse code vector, and its output was used to initialize the first-layer weights of the CNN. This work was the first to use DCNNs initialized by unsupervised training techniques during the period of the deep learning renaissance and led to several other unsupervised pretraining attempts between 2006 and 2011, as the next section shows.

4.2 The Deep Learning Renaissance Fueled by GPUs and Improved Algorithms.

4.2.1 Unsupervised Pretraining. Inspired by the speed and accuracy advantages of unsupervised pretraining (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2006; Ranzato et al., 2006), Ranzato et al. (2007) used a DCNN-like architecture trained in an unsupervised manner to learn hierarchical sparse features that were locally invariant to small shifts and distortions. Their approach, which introduced max pooling (see sections 2.2 and 4.2.3), achieved results very close to the state-of-the-art for the MNIST (LeCun et al., 1998; Ranzato et al., 2006) and the California Institute of Technology (CALTECH-101—Fei-Fei, Fergus, & Perona, 2006; Zhang, Berg, Maire, & Malik, 2006) benchmarks. Despite this early success, DCNNs are still not immune to large-scale shifts and distortions; this is still an open area of research (see section 6.2).

Asserting that the pretraining methods that Hinton et al. (2006), Bengio et al. (2006), and Ranzato et al. (2007), used were complicated and restricted, Weston, Ratle, Mobahi, and Collobert (2008) presented a simpler way to perform deep learning by fusing nonlinear embedding algorithms with deep multilayered architectures (including DCNNs), trained in a supervised fashion. The resulting semisupervised deep learning scheme was inspired by the Laplacian SVMs presented by Belkin, Niyogi, and

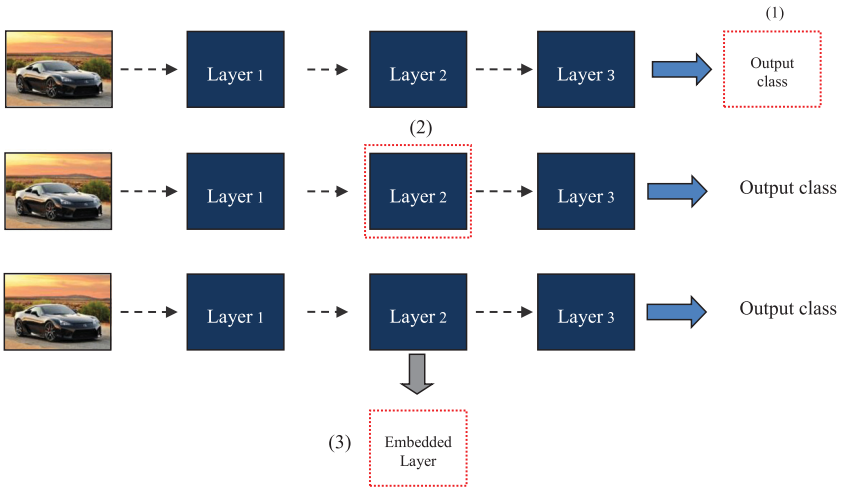

Figure 4: Different modes of embedding regularizes into deep architectures.

Sindhwani (2006) and brought about competitive error rates on the MNIST data set (LeCun et al., 1998), when compared to other shallow semisupervised techniques (Belkin, Niyogi, & Sindhwani, 2006; Collobert, Sinz, Weston, & Bottou, 2006) and the existing deep learning approaches of the time (Hinton et al., 2006; Ranzato et al., 2007; Salakhutdinov & Hinton, 2007). Figure 4 shows how the embedding algorithms were added to regularize either the entire network output, the hidden layers, or an auxiliary network that had the same initial layers of the original network but a new final set of weights. In the figure, the broken red lines illustrate the point at which the embedding algorithms were incorporated.

Along the lines of unsupervised DCNN pretraining (Ranzato et al., 2006, 2007) and semi-supervised embedding (Weston et al., 2008), Ahmed, Yu, Xu, Gong, and Xing (2008) first performed a set of pseudo-tasks on data in an unsupervised fashion and then transferred the resultant knowledge to DCNNs via transfer learning. All the layers of the DCNN, including the final classification layer, were trained with backpropagation. Their results inferred that knowledge transfer followed by supervised training improved DCNN performance and could be applied to a range of visual tasks, including object, gender, and ethnicity recognition. Further details are available in the original paper (Ahmed et al., 2008); an overview on the different forms of knowledge transfer and some of its early successes is provided by Fei-Fei (2006). Recently, the features extracted by DCNNs have been shown to provide an astounding baseline for various computer vision tasks, including scene recognition, fine-grained recognition, attribute detection, image retrieval, and, most significant, image classification (Razavian, Azizpour,

Sullivan, & Sarlsson, 2014). The obvious advantage for computer vision systems that use knowledge transferred from DCNNs is that their exorbitant training times can be eliminated, thus reducing the development and deployment times of such schemes.

A detailed study that investigated the effect of the nonlinearities that follow convolutional filters in DCNNs; the performance of supervised, unsupervised, and randomly learned convolutional filters; and the advantages (if any) of using two stages of feature extraction compared to one was undertaken by Jarrett, Kavukcuoglu, and LeCun (2009), and LeCun, Kavukcuoglu, and Farabet (2010). They found that nonlinearities that comprise rectification and local contrast normalization were key to good accuracy on the MNIST (LeCun et al., 1998), CALTECH-101 (Fei-Fei et al., 2006), and NYU Object Recognition Benchmark (NORB—LeCun et al., 2004) data sets, and that better classification accuracy was obtained from two stages of feature extraction rather than one. In particular, they set a new record on the unmodified MNIST data set, improving on the previous best performance (Ranzato et al., 2006) by following unsupervised pretraining, using a method called predicative sparse decomposition (PSD; Kavukcuoglu, Ranzato, & LeCun, 2010), with supervised reinforcement. The PSD technique, like the work proposed by Ranzato et al. (2006), is based on an encoder-decoder architecture that enforces sparse constraints on the feature vector by utilizing a basic feedforward regressor that is trained to estimate a sparse solution for all the vectorized patches or their stacks in a prescribed training set. Although sparse coding algorithms are generally computationally exorbitant, since the PSD technique approximates the sparse codes, it is computationally cheaper, making it very fast relative to other sparse coding schemes.

Unsupervised (including semisupervised) pretraining, followed by supervised refinement, discussed in this section, was made popular by the deep belief networks proposed at the rise of the deep learning renaissance (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2006). The most common unsupervised schemes used contrastive divergence (Hinton, 2002) methods (see Lee, Grosse, Ranganath, & Ng, 2009), sparse constraints (Ranzato et al., 2006, 2007), or PSD (Kavukcuoglu et al., 2010; LeCun et al., 2010). In general, for these techniques, the feature extraction filters are trained such that representations at a particular stage can be reconstructed from representations of a preceding stage. The major hindrance of this approach is that the feature learning process is independent of the task, although Bengio et al. (2006), Mairal, Bach, Ponce, Sapiro, and Zisserman (2008), and Ranzato and Szummer (2008) attempted to alleviate this by assimilating supervised criteria with unsupervised techniques.

Furthermore, despite the initial promising results obtained from unsupervised pretraining (see Erhan et al., 2010, for a detailed analysis), in recent years, supervised learning has become the leading paradigm for training DCNNs (see section 5.3). However, semisupervised learning is more

biologically plausible. For example, consider how children learn about their environments or, more specifically, how they learn to recognize or classify objects. They are usually supplied a few examples by their caregivers, analogous to semi- or weakly supervised learning, and they use this to generalize on unseen objects. Thus, to align our current heavily supervised models closer to nature, it is envisaged that future DCNNs will go back to using semisupervised schemes, similar to those introduced in this section. These schemes will incorporate, at least initially, supervised criteria to overcome the known issues with their unsupervised counterparts. Such progress will eventually lead to independent, unsupervised systems to tackle the increasingly immense expanses of unannotated data currently available (see section 6.6 for further insight).

4.2.2 GPUs Stimulate Research into DCNNs. Even though the deep learning algorithms that work currently have been available since the 1980s (LeCun et al., 1989a, 1989b), they were taught to be too computationally expensive to allow a great deal of research on the hardware available prior to 2006 (Goodfellow et al., 2016). Furthermore, during program execution, convolution operations are computationally costly and thus make DCNNs significantly slower to evaluate when compared to standard ANNs of the same magnitude. To overcome these constraints, Chellapilla, Puri, and Simard (2006) proposed three novel methods to speed up DCNNs: unrolling convolution, using basic linear algebra software subroutines, and using GPUs. Although GPUs had already been applied to ANNs (Oh & Jung, 2004; Steinkrau, Simard, & Buck, 2005), this work was significant since it was the first implementation of a DCNN using GPUs. Over time, this has become a momentous facet of most award-winning or state-of-the-art DCNNs (Ciresan et al., 2011; Krizhevsky et al., 2012; Hinton et al., 2012; Zeiler & Fergus, 2013, 2014; Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2015a). Although the development of enhanced hardware to facilitate DCNN computation is still an open area of research, it has become largely commercialized in recent years. With this trend, much of the academic focus has been on either the application of this commercially available hardware or algorithmic development to aid swifter processing. Although this is not envisaged to change in the near future, there is an expectation that imminent hardware and software advances will focus on the deployment of DCNNs to mobile devices (see section 6.3).

4.2.3 Max Pooling Leads to Improved Generalization. In 2007, backpropagation was applied for the first time to a DCNN-like architecture that used max pooling (Ranzato et al., 2007). In 2010, Scherer, Müller, and Behnke (2010) showed empirically that the max pooling operation was vastly superior for capturing invariance in image-like data and could lead to improved generalization and faster convergence when compared to a subsampling operation. They demonstrated this by achieving the best published results

on the normalized-uniform NORB data set (LeCun et al., 2004), improving on the previous best (Nair & Hinton, 2009) by over a half percent. Continuing with the empirical work, Jarrett et al. (2009) showed that max pooling alleviated the need for a rectification layer, which is not usually part of DCNN architecture; however, they found that average pooling does not enjoy the same benefit and thus suffers from cancellation effects between neighboring filter outputs.

A detailed theoretical analysis of max pooling and average pooling, supplemented by empirical evaluations, was conducted by Boureau, Ponce, and LeCun (2010). They concluded that the performance of either max or average pooling was dependant on the data and its features, and that for a given classification problem, using either pooling strategy alone may not be optimal. Since max pooling was designed only for feedforward networks, Lee, Gross, Ranganath, and Ng (2009) introduced and applied probabilistic max pooling to convolutional DBNs with the aim of scaling DBNs (Hinton et al., 2006) to full-sized, high-dimensional images. Their resultant translation invariant hierarchical generative model performed well on several classification benchmarks, including MNIST (LeCun et al., 1998) and CALTECH-101 (Fei-Fei et al., 2006). Although it is well known that max pooling leads to a certain degree of invariance to distortions and translations, it accomplishes this by discarding spatial information (Ranzato et al., 2007; Scherer et al., 2010; Szegedy, Liu, et al., 2015). Despite this, it continues to be a key component of several state-of-the-art DCNNs (Ciresan et al., 2011; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy, Liu, et al., 2015). For some of the issues associated with max and average pooling and their proposed solutions, refer to section 5.1.2.

4.3 The Changing Point in the Application of DCNNs for Computer Vision Tasks. The deep learning renaissance of 2006 (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2006), spurred on several successful applications of DCNNs to a wide variety of tasks. These included image and object classification and recognition (Chellapilla, Puri, & Simard, 2006; Ranzato et al., 2007; Weston et al., 2008; Jarrett et al., 2009; Lee et al., 2009; LeCun et al., 2010; Scherer et al., 2010; Boureau et al., 2010; Masci, Meier, Ciresan, & Schmidhuber, 2011), face detection (Nasse, Thurau, & Fink, 2009), and image segmentation (Turaga et al., 2010). Furthermore, they also found interesting applications in scene parsing (Farabet, Couprie, Najman, & LeCun, 2012), vision for autonomous off-road driving (Hadsell et al., 2009), and hand gesture recognition (Nagi et al., 2011). Despite these accomplishments, they were still largely discarded by the mainstream computer vision and machine learning communities (LeCun et al., 2015). This changed after the ILSVRC 2012 (Russakovsky et al., 2015), when a fully supervised DCNN achieved record-breaking classification results on a subset of the ImageNet data set (Krizhevsky et al., 2012). This work has revolutionized the field of computer vision, and as a result, DCNNs have

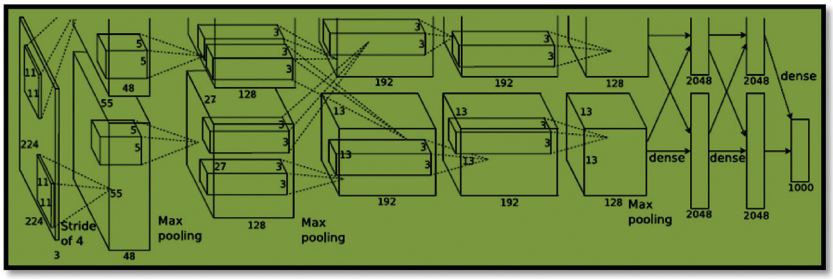

Figure 5: DCNN architecture split over two GPUs (Krizhevsky et al., 2012).

since become the leading architecture for most visual tasks, in particular, for image classification–related applications, as the rest of this review shows.

Central to their success, they implemented several novel and unusual techniques. Rather than using traditional sigmoid or hyperbolic tangent activation functions, they were inspired by Jarrett et al. (2009) and used ReLU (Nair & Hinton, 2010) activations, which allowed much faster training times (see section 5.2.1 for further details). Since their network was too big to fit into one GPU, they spread it over two GPUs arranged in a parallel configuration, which was similar to the multicolumn DCNNs proposed by Ciresan, Meier, and Schmidhuber (2012). Inspired by the local contrast normalization of Jarrett et al. (2009), they applied local response normalization. Denoted mathematically, if a kernel i, at position $(x; y)$ is used to compute the activity of a neuron denoted by $a^i_{x,y}$ and the ReLU nonlinearity is than applied, the response-normalized activity $b^i_{x,y}$ can be expressed as

$$b^i_{x,y} = a^i_{x,y} \Bigg/ \left(k + \alpha \sum_{j=max(0,\, i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}, \tag{4.1}$$

where N is the total number of kernels in the layer and the sum runs over n "adjacent" kernel maps at the same spatial position. This scheme aided generalization and reduced their network classification error rates. They further reduced the classification error by overlapping the network's max pooling layers. Figure 5 illustrates the revolutionary architecture presented by Krizhevsky et al. (2012). It consisted of five convolutional layers, three of which were followed by max pooling layers, and three fully connected layers. The various layer parts in the top half of the figure ran on one GPU, while the layer parts at the bottom ran on the second GPU. The GPUs interacted with each other only at specific layers.

To overcome overfitting, the authors employed a regularization technique known as Dropout (Hinton et al., 2012). Specifically, when each training case was presented to the network during the training phase, each hidden neuron was randomly omitted from the network with a probability of 0.5. Thus, hidden neurons could not rely on other hidden neurons being present, and this prevented complex coadaptations of features on the training data. At test time, all of the hidden neurons were used, but their outputs were multiplied by 0.5 to compensate for the fact that double the number of neurons were now active. The result of this was a strong regularization effect that significantly reduced overfitting (Krizhevsky et al., 2012; Hinton et al., 2012; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Figure 11 (in section 5.4.2) shows the effect of Dropout on a standard feedforward network, with two hidden layers, while section 5.4.1 gives a more formal description of the technique and introduces several of its variants.

Overfitting was further reduced by applying data augmentation, a popular procedure to artificially enlarge a data set (LeCun et al., 1998; Simard et al., 2003; Ciresan et al., 2011, 2012). In particular, they created more images by applying translations and horizontal reflections to the training images, altering the intensities of their color channels, and performing principal component analysis (PCA) on their pixel values, which led to improved classification performance. Krizhevsky et al.'s (2012) model has been used extensively for various purposes since its development. Vast amounts of research have used it to benchmark their models against or as a base model to test new algorithms. Furthermore, their model has inspired DCNN work and has become one of the major contributors to the recent rise in DCNN technology for image classification–related applications.

Notably, preceding the pioneering work of Krizhevsky et al. (2012) was the series of work proposed by Ciresan et al. (2011, 2012). They presented deep hierarchical CNNs, trained in a fully supervised fashion, that achieved the best published results on the NORB (LeCun et al., 1998; Krizhevsky, 2009; Coates, Lee, & Ng, 2011) and MNIST (LeCun et al., 1998; Ciresan, Meier, Gambardella, & Schmidhuber, 2010) classification benchmarks (Ciresan et al., 2011). By stacking these DCNNs into columns, they further improved the state of the art for these benchmarks and, in particular, reached human-level performance on the MNIST data set. Furthermore, on the German traffic sign recognition benchmark (GTSRB; Stallkamp, Schlipsing, Salmen, & Igel, 2011), they surpassed human performance by a factor of two.

Table 1 summarizes the key attributes of the image classification data sets introduced thus far. Although other classification data sets exist (see sections 5.1.2.3 and 5.3.2), these are the most commonly used for DCNN evaluation and benchmarking. Among them, the MNIST data set (LeCun et al., 1998) has stood the test of time and become the most popular,

Table 1: Popular DCNN Image Classification Benchmarks.

Data Set	Image Description	Image Size	Number of Images	Number of Classes	Reference
MNIST	Handwritten digits	28×28	70,000	10	LeCun et al., 1998
CALTECH-101	Color images	$\pm 300 \times 200$	9146	101	Fei-Fei et al., 2006
CALTECH-256	Color images	$\pm 300 \times 200$	30,607	256	Griffin, Holub, & Perona, 2007
Normalized-uniform NORB	Stereo images with uniform background	96×96	48,600	5	LeCun et al., 2004
Jittered and cluttered NORB	Stereo jittered images with clutter	96×96	349,920	6	LeCun et al., 2004
CIFAR-10	Natural color images	32×32	60,000	10	Krizhevsky, 2009
CIFAR-100	Natural color images	32×32	60,000	100	Krizhevsky, 2009
ILSVRC	High-resolution color images	Variable size	>1.2 million	1000	Russakovsky et al., 2015

although modern classification systems are judged by their success on the ILSVRC (Russakovsky et al., 2015), as the next section shows.

4.4 Representative Improvements Exemplify DCNN Dominance. Since the groundbreaking work of Krizhevsky et al. (2012), DCNNs have dominated image classification tasks, in particular the ILSVRC (Russakovsky et al., 2015). In fact, they were triumphant in every ImageNet classification challenge since 2012 (Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015; He, Zhang, Ren, & Sun, 2015b). In an attempt to understand them and derive ways to improve their performance, Zeiler and Fergus (2014) introduced a new visualization technique using a multilayered deconvolutional network (Zeiler, Taylor, & Fergus, 2011) that provided vision into the intermediate feature extraction layers of the network. They used this in a diagnostic role to improve the DCNN architecture and performance of Krizhevsky et al. (2012). Thus, when compared to Krizhevsky et al. (2012), their model achieved better results on the ImageNet classification benchmark, and multiple models were averaged to win the ILSVRC 2013 (Russakovsky et al., 2015). Furthermore, their model generalized excellently, and they demonstrated this by achieving the best published results on the CALTECH-101 (Fei-Fei et al., 2006) and CALTECH-256 data sets (Griffin et al., 2007). Although their visualization technique worked well on relatively higher-dimensional color images, it would be interesting to test its applicability on the popular MNIST data set (LeCun et al., 1998), since it is conceivable that the deconvolutional network may not be able to reproduce the lower-dimensional gray-scale MNIST images with the same accuracy. Another interesting direction from using the extracted features in a diagnostic role will be to investigate its applicability to solve some of the remaining DCNN challenges, specifically like those mentioned in sections 6.2 and 6.4.

Szegedy, Liu, et al. (2015) introduced a DCNN architecture that they called the Inception model. A particular incarnation of this model, GoogLeNet, produced outstanding image classification and object detection results, winning both the ImageNet classification and detection challenges in 2014 (Russakovsky et al., 2015). Their success was brought about by using a very large network, consisting of 22 layers. Since the cost of this is a larger number of parameters, which makes the network more prone to overfitting and a considerably larger computational burden, they used a carefully engineered design, based on Hebbian principles, that allowed them to move from a fully to sparsely connected convolutional architecture, which was motivated by the findings of Arora, Bhaskara, Ge, and Ma (2014). Specifically, their architecture used 1×1 convolutions heavily, inspired by Lin et al. (2013), to perform two functions. Most significant, they served as dimension-reduction blocks prior to the more computationally costly 3×3 and 5×5 convolutions, and they included the use of rectified linear activations (Nair & Hinton, 2010), thus making them dual purpose. Subsequently,

they were able to increase the depth and width of their network, while only marginally increasing the computational cost. Figure 8 (in section 5.1.1.1) illustrates an Inception module, which incorporates the dimension reduction 1×1 convolution filters, illustrated by the bevel in the diagram. These modules are the building blocks of the Inception model, which has since been improved several times, as discussed in section 5.1.1.2.

Similar to Szegedy, Liu, et al. (2015), Simonyan and Zisserman (2014), the runners-up in the same ILSVRC 2014 classification contest (Russakovsky et al., 2015), also used a very deep DCNN, which consisted of 19 layers compared to the 22 of their competitors. However, asserting that the Inception model was too complex, they kept all the parameters of their DCNN architecture constant and steadily increased the depth alone. This was made feasible by using smaller-sized convolutional (3×3) filters throughout the network, which was inspired by Ciresan et al. (2011), who already used smaller kernels, albeit for shallower networks applied to simpler tasks.

The winners of the ILSVRC 2015 (Russakovsky et al., 2015), He et al. (2015b), used an even deeper DCNN, when compared to Simonyan and Zisserman (2014) and Szegedy, Liu, et al. (2015). In fact, their model was ultra-deep in that it consisted of 152 layers. Since deeper models are harder to train and suffer from degradation (of training and thus test accuracy) (He et al., 2015b; He & Sun, 2015; Srivastava, Greff, & Schmidhuber, 2015a, 2015b), they introduced a new residual learning framework.[1] They reformulated the layers of the network and forced them to learn residual functions with reference to their preceding layer inputs rather than learning unreferenced functions. This allowed errors to be propagated directly to the preceding units, and thus made these networks easier to optimize and, although they were ultra-deep, easier to train. They tried different residual module configurations and network architectures and found that optimized residual modules worked more optimally compared to their initial modules. Figure 6 compares the difference between the original residual module and its optimized successor, which resulted in faster computation. As illustrated, ReLUs (Nair & Hinton, 2010) feature heavily in both versions; however, optimized residual connections make use of the dimension-reducing 1×1 filters to cushion computation. A more formal description of the original residual learning technique, as well its improvements, is discussed in section 5.5.4.

Table 2 illustrates the performance of DCNNs in the ILSVRC since its inception. Significantly, the table highlights DCNN domination over earlier methods that used feature extraction and compression, followed by classification with a shallow classifier (Perronnin, Sánchez, & Mensink, 2010;

[1] Degradation is caused by the poor propagation of activations and gradients because of stacking several nonlinear transformations on top of each other.

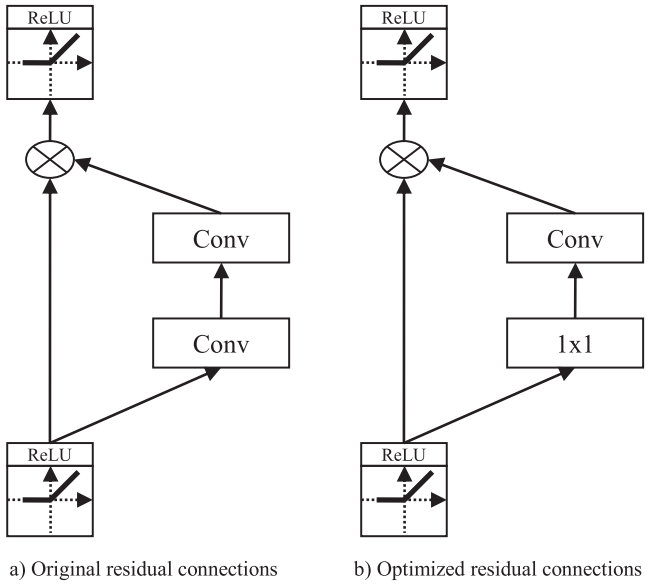

Figure 6: Residual versus improved residual modules.

Lin et al., 2011; Sánchez & Perronnin, 2011), as well as an approximate correlation between classification performance and network depth (see section 5.5.4). Further results can be found in Table 4.

Notwithstanding degradation (He et al., 2015b; He & Sun, 2015; Srivastava et al., 2015a, 2015b), deeper models are generally more accurate and thus produce better empirical results; however, as depth increases, so do computational costs. With this in mind, the representative work discussed here has led to several attempts to improve the classification accuracy of DCNNs by modifying their architecture for improved performance without losing sight of the computational burden imposed on such models. In particular, the models of Szegedy, Liu, et al. (2015), Simonyan and Zisserman (2014), and He et al. (2015b) all focused on deeper or wider networks for improved accuracy, with several tricks, ranging from dimension reduction to residual learning, to handle the associated computational strain placed on deeper networks. This has led to a classic engineering dilemma between deeper models, which are more accurate but computationally expensive, and shallower models, which are easier and cheaper to train but do not produce the same classification accuracy. Thus, although there have been several attempts to address this, maintaining accuracy with reduced computational expenditures remains an open challenge for DCNNs. To this end, section 5.5.4 deals with the swifter processing of deep models, while the

Table 2: ILSVRC Image Classification Results from 2010.

Year	Team	Number of Layers	General Contribution	Position	References
2010	NEC	Shallow	Fast feature extraction, data compression, SVM classifier	First	Lin et al., 2011
2011	XRCE	Shallow	High-dimensional image signatures, data compression, SVM classifier	First	Perronnin et al., 2010; Sánchez & Perronnin, 2011
2012	SuperVision	8	Efficient GPU-based DCNN, with Dropout and several other innovations	First	Krizhevsky et al., 2012
2013	Clarifai	8	DCNN architecture based on deconvolutional visualization technique	First	Zeiler & Fergus, 2014; Zeiler et al., 2011
2014	GoogLeNet	22	DCNN architectural design based on Hebbian principle and multiscale ideas	First	Szegedy, Vanhoucke et al., 2015
2014	VGG	19	Improvements to DCNN convolutional layers, increased network depth	Second	Simonyan & Zisserman, 2014
2015	MSRA	152	Introduction of deep residual learning for ultra DCNNs	First	He et al., 2015b

latest developments, trends, and recommendations in this regard are introduced in section 6.3.

5 A Deep Array of Further Improvements and Recent Advancements

In addition to the revolutionary work of Krizhevsky et al. (2012) and the further symbolic improvements described in the preceding section (Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015; He et al., 2015b), several other improvement attempts related to network architecture, nonlinear activation functions, supervision components, regularization mechanisms, optimization techniques, and swifter processing of DCNNs have supplemented the popularity of DCNNs. In the sections that follow, we survey these improvements in detail, focusing on their employment to image classification applications. Along the way, we compare and contrast the different methodologies and techniques used to design these improvements. Toward the end of the section, we empirically

summarize their classification results on several popular image classification benchmarks.

5.1 Network Architecture. This section first introduces the improvements made to the convolutional layers of DCNNs, followed by discussions that deliberate several pooling schemes, including the latest advancements in this regard.

5.1.1 Convolutional Layers. The convolutional layers learn the feature representations of their input images, and this makes them the main building block of DCNNs. Thus, it is natural to try to improve this aspect of DCNN architecture. Here we introduce the motivations behind some of the key innovations in this area.

5.1.1.1 Network in network. Since the convolutional layers use linear filters, which are more suited to learning latent features (hidden properties of an image) that are linearly separable, they are not cable of extracting abstract representations from images.[2] Thus, Lin et al. (2013) proposed replacing them with universal function approximators. Specifically, they replaced the conventional local convolutional filters with multilayered perceptrons (MLPs), which are compatible with the architecture and training procedures of DCNNs, to convolve over the input resulting in a MLP convolutional layer. The computation performed by this layer, when the ReLU (Nair & Hinton, 2010) is used as the activation function, can be expressed mathematically as

$$f^{n}_{i,j,k_n} = \max\left(w^{nT}_{k_n} f^{n-1}_{i,j} + b_{k_n}, 0\right) \tag{5.1}$$

where the pixel index of the feature map is denoted by (i, j); the input patch, centered at location (i, j), is denoted by $x_{i,j}$; the feature map channels are indexed by k; and n represents the number of layers in the MLP.

The proposed method demonstrated that these MLP convolutional layers model local image patches better than standard convolutional layers. When combined with a novel global average pooling technique, which spatially averaged the feature maps of the final layer, was used to replace the standard fully connected layer, they produced state-of-the-art results on two versions of the CIFAR-10 (Krizhevsky, 2009; Wan, Zeiler, Zhang, LeCun, & Fergus, 2013; Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013) and CIFAR-100 benchmarks (Krizhevsky, 2009; Srivastava & Salakhutdinov, 2013), and very close to the state of the art on the MNIST

[2] *Abstract* in this contents relates to features that are invariant to features of an equivalent concept (Bengio, Courville, & Vincent, 2013).

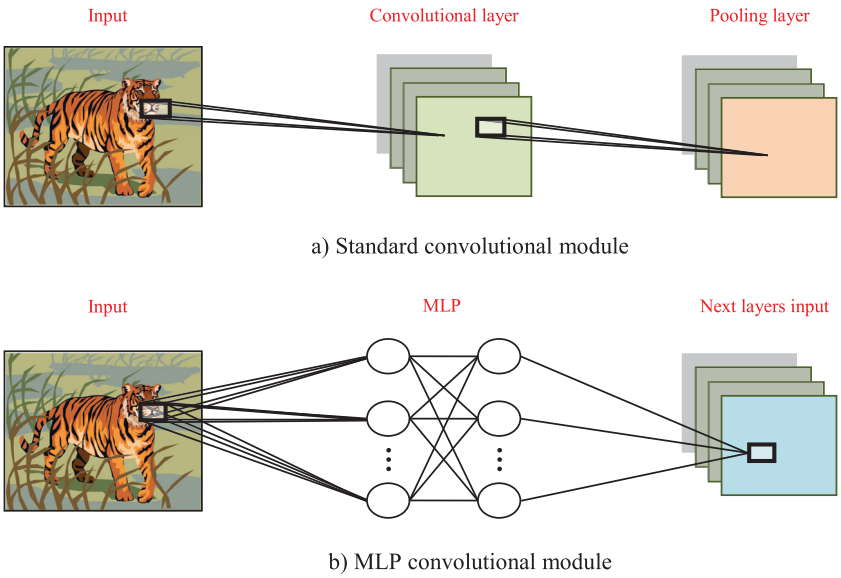

Figure 7: Convolutional versus MLP convolutional layers.

data set (LeCun et al., 1998; Goodfellow et al., 2013). Although the proposed global average pooling technique, which has fewer parameters and thus cheaper computational costs compared to fully connected layers, contributed to a reduction in overfitting for the relatively small MNIST (LeCun et al., 1998) and CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) data sets, a study into overfitting using this type of layer to replace the conventional fully connected layers of other DCNN models is still at large for larger data sets like ImageNet (Russakovsky et al., 2015). Figure 7 illustrates the difference between a conventional convolutional module and an MLP convolutional module, which is the main building block of the network in network (NIN) model. While both variants map the local receptive field representing the hidden input features to a succeeding layer, panel b uses a micronetwork for enhanced representation.

5.1.1.2 Inception and improved Inception models. The Inception model (Szegedy, Liu, et al., 2015), inspired by Lin et al. (2013) and discussed in section 4.4, used a dimension-reduction (1×1 convolutional filters) technique to lessen the computational burden of the expensive convolutional operation. In order to scale up and further improve DCNN classification accuracy in a computationally efficient manner, the Inception model was later enhanced by using factorized convolutions (see section 6.6) and aggressive dimension reductions within the network. While the original Inception module still used 5×5 convolutions, the improved version replaced this

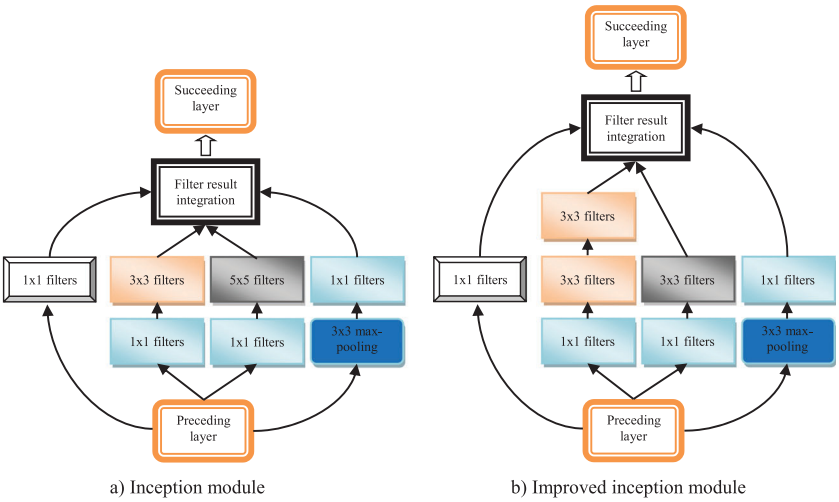

Figure 8: Inception versus improved Inception modules.

with two computationally cheaper 3×3 convolutions (Szegedy, Vanhoucke et al., 2015). Figure 8 illustrates the differences between the two modules.

Inspired by the image classification accuracy accomplished by residual networks (He et al., 2015b), discussed in sections 4.4 and 5.5.4, the Inception architecture (Szegedy, Liu, et al., 2015; Szegedy, Vanhoucke et al., 2015) was further refined and combined with residual connections to form residual Inception networks (Szegedy, Ioffe, & Vanhoucke, 2016). The paper provided clear evidence advocating that training with residual connections significantly accelerated the training of Inception networks. Although they tested several Inception-only and residual Inception architectures, they found that a hybrid residual Inception architecture yielded the best single-model classification accuracy, albeit at a higher computation cost when compared to the improved Inception architecture described by Szegedy, Vanhoucke et al. (2015). Furthermore, when they combined a new, improved Inception model, which had a simpler architecture and more Inception modules compared to their earlier model (Szegedy, Vanhoucke et al., 2015), into an ensemble with three residual Inception networks, they achieved the best published results on the challenging ImageNet image classification benchmark (Russakovsky et al., 2015; He et al., 2015b). Despite this success, further work is required in order to reduce the computational burden imposed on the hybrid architecture.

5.1.1.3 Doubly convolution. Motivated by intuition, followed by a theoretical analysis, which advocated that several of the learned filters of well-trained DCNNS are slightly translated versions of each other, Zhai, Cheng,

Lu, and Zhang (2016) newly proposed doubly convolutional neural networks, which make use of a double convolution operation in the convolutional layers. This allows them to learn clusters of filters, where filters within each cluster are translated forms of each other. In order to accomplish this, a set of meta-filters is allocated to a doubly convolutional layer. The sizes of these meta-filters are larger than the effective filter size, which are extracted from each of them. This corresponds to convolving the meta-filters with an identity kernel. By concatenating the extracted filters and then convolving it with the input, the technique attains double convolution. This technique is also complementary to Maxout (Goodfellow et al., 2013), which we introduce in section 5.2.8, since there is an opportunity to pool along the activations generated by the same meta-filter. They outperformed the NIN model (Lin et al., 2013), which also altered the standard convolutional layer for improved classification accuracy, on the CIFAR-10 and CIFAR-100 data sets (Krizhevsky, 2009); furthermore, since the architecture of the doubly convolutional networks can be amenably varied, they are parameter efficient, thus reducing their storage space requirements without a loss in accuracy. The downside of such an approach is that the double convolution operation will incur additional computational costs in comparison to a standard convolutional layer.

5.1.1.4 Analysis and outlook. The convolutional filters, the workhorses of DCNNs, are generalized linear models of the underlying image patches that they convolve, and although they work well for extracting features that have a low level of abstraction, they are challenged when they need to extract highly nonlinear functions of our input images. This advocated the need for more effective nonlinear feature extractors, starting with the NIN model. The architecture introduced by Lin et al. (2013) led to a series of other improvements that also focused on the convolutional layers. At the heart of these approaches were the Inception (Szegedy, Liu, et al., 2015) and improved Inception models (Szegedy, Vanhoucke et al., 2015; Szegedy et al., 2016), which were meticulously engineered to mitigate any computational constraints, and this facilitated increased network size (width and depth) for enhanced classification accuracy. However, notwithstanding the promising empirical results of these models, a theoretical justification for their successes is still lacking. Furthermore, their complex and highly optimized architectures do not warrant modification without possible performance constraints, thus the need to exercise caution when adopting them. Future work should attempt to justify the reasons for the empirical successes of the innovative convolutional layers discussed here, and this should be supplemented by novel convolutional-related modifications that address the concerns associated with our current models, such as the computational encumbrance imposed by the convolutional operation, their inability to extract potent features, and the complexity of the current models that mitigate these concerns. This will not only advance classification

accuracy and robustness and promote swifter computation but will lead to models that are easily adaptable to different tasks. Some of the promising recent developments along these lines include double convolution (see section 5.1.1.3), tiled convolution (Ngiam et al., 2010; Wang & Oates, 2015), and, in particular, dilated convolution (Yu & Koltun, 2015), which has shown promising results for diverse tasks such as speech recognition and synthesis (Sercu & Goel, 2016; Oord et al., 2016), machine translation (Kalchbrenner et al., 2016), and scene segmentation (Yu & Koltun, 2015).

5.1.2 Pooling Layers. After the convolutional layers, the pooling layers are perhaps the most important. They recapitulate the responses of neighboring neurons from the same kernel map and thus reduce the dimensions of their input representations. Significantly, they provide DCNNs with their spatial invariance (Krizhevsky et al., 2012; LeCun et al., 2015). In addition to average and max pooling, introduced in sections 2.2 and 4.2.3, we survey some of the other successful pooling techniques mentioned in the literature.

5.1.2.1 L_p pooling. Although the use of max pooling has resulted in excellent empirical results (Ciresan et al., 2011; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy, Liu, et al., 2015; Szegedy, Vanhoucke et al., 2015), it can overfit the training data and does not guarantee generalization on test data. Average pooling, on the other hand, considers all the elements in the pooling region and thus areas of low activation may lessen the effect of areas of high activation (Zeiler & Fergus, 2013; Sainath, Kingsbury, Mohamed et al., 2013). To address these issues, a viable alternative is the biologically inspired L_p pooling, which is modeled on complex cells (Simoncelli & Heeger, 1998; Hyvärinen & Köster, 2007). In a given pooling region R_j, it takes the weighted average of the activations a_i, as illustrated by equation 5.1:

$$s_j = \left(\sum_{i \epsilon R_j} a_i^p \right)^{1/p} \tag{5.2}$$

Notably, when $p = 1$ the equation corresponds to average pooling, while $p = \infty$ translates to max pooling. For values of $1 < p < \infty$, L_p pooling can be seen as a trade-off between average and max pooling (Sainath, Kingsbury, Mohamed et al., 2013). Although L_p pooling has been applied previously (Yang, Yu, Gong, & Huang, 2009; Kavukcuoglu, Ranzato, Fergus, & LeCun, 2009), when it was combined with DCNNs (Sermanet, Chintala, & LeCun, 2012), it resulted in exceptional image classification results and a new state of the art on the Street View House Numbers (SVHN) classification benchmark, beating the previous best set by Netzer et al. (2011).

Moreover, theoretical analysis conducted by Boureau et al. (2010), Bruna, Szlam, and LeCun (2013), and Gulcehre, Cho, Pascanu, and Bengio (2014) suggests that it provides better generalization when compared to max pooling.

5.1.2.2 Stochastic and fractional max pooling. Motivated by the problems with average and max pooling and the regularization effect of Dropout (Turaga et al., 2010; Hinton et al., 2012), Zeiler and Fergus (2013) introduced stochastic pooling to replace the deterministic average and max pooling techniques. Specifically, in stochastic pooling, by normalizing the activations within each region j, the probabilities p for the region are first computed by

$$p_i = \frac{a_i}{\sum_{k \epsilon R_j} a_k} \tag{5.3}$$

Then, based on p, a sample is taken from the multinomial distribution, formed from the activations of each pooling region, in order to pick a location l within the region. Thus, the pooled activation is simply a_l:

$$s_j = a_l \text{ where } l \sim P(p_1, \ldots, p_{|R_j|}). \tag{5.4}$$

Although stochastic pooling has the same benefits as max pooling, its stochastic nature helps it prevent overfitting, thus making it an effective network regularization technique that can be combined with other approaches such as Dropout (Hinton et al., 2012; Srivastava et al., 2014) and data augmentation (LeCun et al., 1998; Simard et al., 2003; Ciresan et al., 2011, 2012; Montavon, Orr, & Müller, 2012). When applied to image classification tasks, stochastic pooling outperformed average and max pooling on the MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), and SVHN (Netzer et al., 2011) benchmarks.

Similar to stochastic pooling, fractional max pooling (Graham, 2014), also introduces stochastic attributes to the pooling process. However, divergent from stochastic pooling, the selection of the pooling regions, rather than the pooling operations within them, has a stochastic nature. More specifically, while stochastic and traditional max pooling use $\beta \times \beta$ max pooling, where $\beta = 2$ (see section 2.2), fractional max pooling introduces a fractional factor β (e.g., $\sqrt{2}$), which is selected either randomly or pseudorandomly from the range $1 < \beta < 2$, to reduce the spatial dimensions of the pooling input. They obtained state-of-the-art results on the CIFAR data sets (Krizhevsky, 2009; see Table 6); however, their observations lacked suitable motivation and the technique still needs to be tested on other architectures such as Inception (Szegedy, Liu, et al., 2015) and Residual networks (He et al. 2015b).

5.1.2.3 Mixed pooling. Inspired by the stochastic nature of the pooling technique described by Zeiler and Fergus (2013) and other successful stochastic regularization techniques such as Dropout (Hinton et al., 2012; Srivastava et al., 2014) and DropConnect (Wan et al., 2013) (see section 5.4.2), Yu, Wang, Chen, and Wei (2014) introduced a novel mixed pooling technique to further boost the regularization abilities of DCNNs and address the known issues associated with average and max pooling (Zeiler & Fergus, 2013; D. Yu et al., 2014; Sainath, Kingsbury, Mohamed et al., 2013). They also employed a stochastic procedure to utilize, randomly, max or average pooling during DCNN training. Expressed mathematically, the mixed-pool output y_{kij} in relation to the kth feature map is computed by:

$$y_{kij} = \lambda \cdot \max_{(p,q)\in\Re_{ij}} x_{kpq} + (1-\lambda) \cdot \frac{1}{|\Re_{ij}|} \sum_{(p,q)\in\Re_{ij}} x_{kpq}, \tag{5.5}$$

where the element at location (p, q), within the pooling region $\Re_{ij}$ with size $|\Re_{ij}|$, is represented by x_{kpq}, and either max or average pooling is selected by λ, which has a random value of either one or zero. Similar to stochastic pooling, the combination of mixed pooling with other regularization techniques is possible. Comparatively, when tested on the SVHN (Netzer et al., 2011) classification challenge, this method proved superior to average, max, stochastic, and L_p pooling (Sermanet et al., 2012). Additionally, when tested on the CIFAR-10 and CIFAR-100 benchmarks (Krizhevsky, 2009), it also provided enhanced classification performance over average, max, and stochastic pooling (Zeiler & Fergus, 2013), and the parameterized scheme introduced by Malinowski and Fritz (2013).

5.1.2.4 Mixed, gated, and tree pooling. Experiments conducted by Lee, Gallagher, and Tu (2016) support the findings of Boureau et al. (2010), who also found that there were instances where either max or average pooling performed better than the other did. Thus, they explored learning the pooling function by combining max and average pooling using a responsive (achieved via a gate) and unresponsive strategy. The output of the gated max-average method can be computed by

$$f_{gate}(\mathrm{x}) = \sigma(w^{\mathrm{T}}\mathrm{x}) f_{\max}(\mathrm{x}) + (1-\sigma(w^{\mathrm{T}}\mathrm{x})) f_{avg}(\mathrm{x}), \tag{5.6}$$

where the values in the pooling region are denoted by x and the values of the gating mask are denoted by w. Furthermore, inspired by Bulo and Kontschieder (2014), who incorporated MLPs with decision trees, Lee et al. (2016) used a binary decision tree to learn a combination of previously learned individual pooling filters. A particular incarnation of their approach, which combined their tree and max-average methods, achieved state-of-the-art results on several benchmarks. In particular, they

outperformed several high-performing convolutional networks such as NIN (Lin et al., 2013), stochastic pooling DCNNs (Zeiler & Fergus, 2013), the DCNNs presented by Jarrett et al. (2009), Maxout networks (Goodfellow et al., 2013), and DropConnect networks (Wan et al., 2013) on various image classification benchmarks, including the MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), and SVHN (Netzer et al., 2011) data sets. Notably, despite their successes, RCNNs (Liang & Hu, 2015) outperformed them on the CIFAR-100 data set. Furthermore, for future DCNNs to readily incorporate decision analysis tools such as decision trees into their architectures, further work on reducing the computational costs and exorbitant number of model parameters required by such models is still required.

5.1.2.5 Spectral pooling. Rather than focus on the computational speed gains of moving the convolutional operation out of the spatial domain, similar to the work described by Mathieu, Henaff, and LeCun (2013), Rippel, Snoek, and Adams (2015) proposed learning the convolutional filters of DCNNs directly in the frequency domain. More significant, the authors proposed spectral pooling, which projected spectral representations into the frequency domain and then truncated these representations as an unconventional dimensionality-reduction technique, when compared to the popular max pooling. More precisely, spectral pooling first computes the discrete Fourier transform (DFT) of an input feature map $\mathrm{x} \in \mathbb{R}^{M \times N}$ and then crops the frequency representation by maintaining only the central $H \times W$ sub-matrix of frequencies that are governed by the dimensions of the desired output feature map $H \times W$. Finally, the inverse DFT maps the truncated representation back to the spatial domain. Their method provided a viable solution to the loss of spatial information associated with max pooling (Ranzato et al., 2007; Scherer et al., 2010; Szegedy, Liu, et al., 2015; Rippel et al., 2015) and a new form of stochastic regularization similar to a Dropout (Hinton et al., 2012; Srivastava et al., 2014) variant, known as Nested Dropout (Rippel, Gelbart, & Adams, 2014). On the CIFAR-10 and CIFAR-100 benchmarks (Krizhevsky, 2009), they outperformed several of the works already introduced in this review (Lin et al., 2013; Zeiler & Fergus, 2013; Goodfellow et al., 2013; Liang & Hu, 2015), as well as the deeply supervised DCNNs proposed by Lee, Xie, Gallagher, Zhang, and Tu (2015), but not the combined tree-mixed pooling technique of Lee et al. (2016). The successes described by this work advocate the need for further research into hybrid DCNNs that make use of digital signal processing fundamentals to improve the accuracy of our current classification systems.

5.1.2.6 Spatial pyramid pooling. DCNNs are restricted in that they can only handle a fixed input image size (e.g., 96×96). In order to make them more flexible and thus handle images of different sizes, scales, and aspect ratios, inspired by the spatial pyramid matching described in papers by

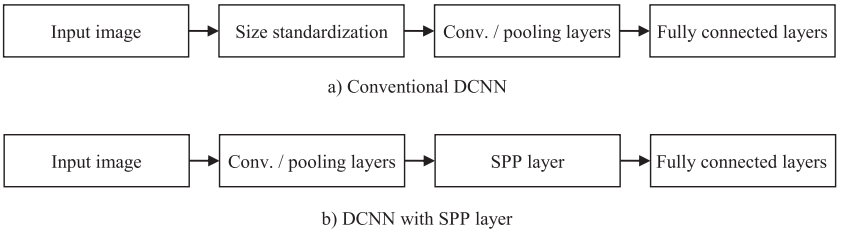

Figure 9: Conventional versus SPP DCNNs.

Grauman and Darrell (2005), Lazebnik, Schmid, and Ponce (2006), and Yang et al. (2009), He, Zhang, Ren, and Sun (2014) proposed spatial pyramid pooling (SPP). They used multilevel spatial bins, which have sizes proportional to the image size, and this allowed them to generate a fixed-length representation, irrespective of the image size or scale. The SPP layer was integrated into DCNN architecture between the final convolutional/pooling layer and the first fully connected layer (see Figure 9) and thus performed information aggregation deep in the network to prevent fixing the size (via cropping or warping) of the image at the input. Unlike stochastic (Zeiler & Fergus, 2013) and L_p pooling (Sermanet et al., 2012), SPP is designed to work with max pooling layers rather than replace them. Among other successes, they set a new record on the CALTECH-101 data set (Fei-Fei et al., 2006), beating the previous best set by Chatfield, Simonyan, Vedaldi, and Zisserman (2014), and they came in third in the classification component of the ILSVRC 2014 (Russakovsky et al., 2015), behind Simonyan and Zisserman (2014) and Szegedy, Liu, et al. (2015). Further work along these lines is required to facilitate commercial DCNN deployment on a variety of portable devices, since this will relax the constraints placed on the image capturing system. Furthermore, this work has shown that tried and tested computer vision-based techniques need not be forsaken in the face of deep learning and that room for this type of traditional computer vision integration is still available.

5.1.2.7 Multiscale orderless pooling. Inspired by Lazebnik et al. (2006), Gong, Wang, et al. (2014) attempted to make DCNNs more robust to invariance without compromising their discriminative power. Asserting that max pooling may not provide invariance to large-scale global deformations, they proposed multiscale orderless pooling (MOP), which extracts patches at multiple scales, beginning with the complete image and then pools each scale disregarding spatial information. Specifically, they extract deep activation features from the whole image, to preserve global spatial layout, and from local patches, to capture fine-grained details. Next, the fine-grained details are aggregated via VLAD encoding (Jegou et al., 2012), which has

an orderless nature and thus contributes to a more invariant representation. Finally, the initial global deep activations and the VLAD encoded features are concatenated to form a new image representation. Their method proved successful at a wide variety of applications, including scene classification, data retrieval, and, most significant, image classification producing competitive results on the ILSVRC 2012/2013 (Russakovsky et al., 2015). With the ever rising size of image data sets (see Table 1), further investigation into the merge of feature compression techniques, such as VLAD encoding and DCNN technology, is warranted.

5.1.2.8 Transformation invariant pooling. Since the features extracted by DCNNs lack invariance to known nuisance variations in data, inspired by max pooling (Boureau et al., 2010) and multiple instance learning (Wu, Yu, Huang, & Yu, 2015), Laptev, Savinov, Buhmann, and Pollefeys (2016) introduced a new pooling technique to generate transformation-invariant features. Given an input image x, they formulate new features $g_k(x)$ from a predefined set of possible transformations ϕ, such that the new features are independent of any known nuisance variations of the input. Formally, these features are formulated in the following manner:

$$g_k(x) = \max_{\emptyset \in \Phi} f_k(\emptyset(x)). \tag{5.7}$$

Subsequently, they refer to this max pooling over transformations as transformation-invariant pooling (TI pooling). By applying the maximum operator, learned features are less dependent on the known nuisance variations. Furthermore, for specific transformation sets, they theoretically prove complete transformation invariance. Similar to the SPP integration into DCNN architecture (He et al., 2014), the authors proposed integrating TI pooling at the same point in the network; however, they used parallel Siamese architectures—two or more identical subnetworks that share the same weights (Bromley et al., 1993)—and applied TI pooling at their outputs prior to the fully connected layers. On two variations of the MNIST data set (Larochelle, Erhan, Courville, Bergstra, & Bengio, 2007; Jaderberg, Simonyan, & Zisserman, 2015), which were designed to benchmark rotation-invariant algorithms, their method obtained results comparable to or better than other state-of-the-art DCNNs, with the added advantage of requiring fewer model parameters since they did not use data augmentation, a popular technique for invariant tasks (Van Dyk & Meng, 2012).

5.1.2.9 Analysis and outlook. Pooling is imperative to diminish the computational burden of the expensive convolutional layers; however, despite the initial successes of average pooling and the contribution of max pooling to the recent rise of DCNNs, inadequacies associated with them (see sections 4.2.3 and 5.1.2.1) have led researchers to investigate other

pooling strategies. Although L_p pooling is biologically plausible and there is theoretical evidence to show it results in better generalization compared to max pooling, the latter continues to enjoy greater popularity for image classification tasks, probably because of its known ability to capture invariance from visual data. The stochastic nature of stochastic pooling (Zeiler & Fergus, 2013) and mixed pooling (D. Yu et al., 2014) gives them an advantage over max pooling with relation to their intrinsic ability to avoid overfitting. The downside is that their inherent probability computations put them at a disadvantage concerning their computational burden when compared to other deterministic techniques such as max or average pooling. The tree-based scheme of Lee et al. (2016) produced exceptional classification performance, but utilizing decision analysis tools adds complexity and computational strain. Although SPP (He et al., 2014) addresses variations in image properties and is fast and effective, architectures that use it cannot be trained in an end-to-end manner. While TI pooling (Laptev et al., 2016) advances DCNN invariance, making DCNNs invariant to geometric transformations and translations, and in particular, large-scale variances, is still an open area of research requiring further efforts, some of which are touched on in section 6.2.

From the analysis, we can conclude that the different pooling strategies have various pros and cons, and thus a particular superlative and generic strategy cannot be singled out. Although max pooling is probably the most established, the choice of strategy will depend largely on the requirements of a particular classification task and the resources available to accomplish it. Some of the key factors to consider here are system complexity, since it is possible to incorporate techniques from digital signal processing (Rippel et al., 2015), decision analysis (Lee et al., 2016), and traditional computer vision (He et al., 2014; Gong, Wang, et al., 2014), required classification accuracy, the consequences of overfitting, and the available computational resources. Future pooling innovations should focus on harmonizing these conflicting requirements, while not losing cognizance of the need for them to be biologically conceivable, so that we are able to both improve our models and understand more about our current vision system.

5.2 Nonlinear Activations. The choice of activation function affects network training time, and this has a significant influence on the performance of large DCNNs on large data sets (Krizhevsky et al., 2012). Introduced by Nair and Hinton (2010) for deep Boltzmann machines, ReLUs were made popular for DCNNs by Krizhevsky et al. (2012), although Glorot, Bordes, and Bengio (2011) had already shown that they lead to faster training times in fully supervised networks without the need for unsupervised pretraining. Figure 10 compares the training times of ReLUs (solid line) to hyperbolic tangent (dashed line) activations for a four-layer DCNN (Krizhevsky et al., 2012), trained on the CIFAR-10 data set (Krizhevsky, 2009). The DCNN with ReLUs was trained six times faster than an equivalent

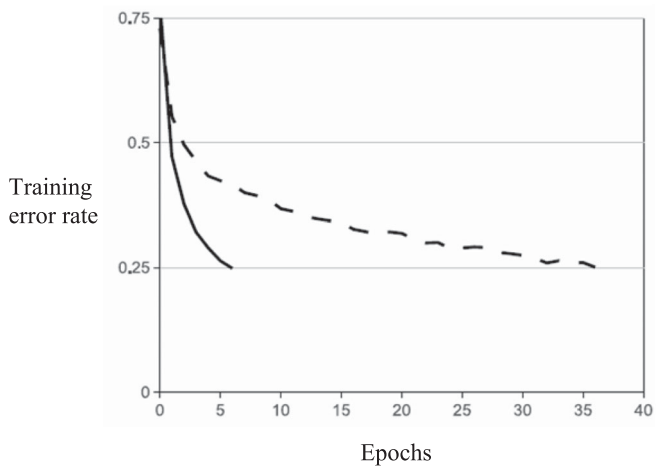

Figure 10: Training times of ReLUs versus tanh activations (Krizhevsky et al., 2012).

network that used hyperbolic tangent activations (Krizhevsky et al., 2012). We next briefly introduce this nonsaturating activation function and discuss the motivations that have led to several of their successors.

5.2.1 ReLU Activations. Traditional activation functions, such as the sigmoid or hyperbolic tangent are given by $f(x) = 1/(1 + e - x)$ and $f(x) = \tanh(x)$, respectively, where f is the neuron's output as a function of its input x (the same notation is used for the remainder of the activation functions that follow). The ReLU (Nair & Hinton, 2010), a piecewise linear function, has the simplified form $f(x) = \max(x, 0)$. The ReLU retains only the positive part of the activation, by reducing the negative part to zero, while the integrated maximum operator promotes faster computation. The ReLU has been used in several state-of-the-art image classification systems (Zeiler & Fergus, 2013, 2014; Lin et al., 2013; Gong, Wang, et al., 2014; Simonyan & Zisserman, 2014; Szegedy, Vanhoucke et al., 2015; Szegedy, Liu et al., 2015). An in-depth discussion and further motivations on them can be found in the work presented by Glorot et al. (2011).

5.2.2 LReLU Activations. Even though ReLUs (Nair & Hinton, 2010) lead to faster convergence (Nair & Hinton, 2010; Glorot et al., 2011; Krizhevsky et al., 2012; Maas, Hannun, & Ng, 2013) and do not suffer from the vanishing gradient problem, in which the lower layers have gradients near zero because high layers are almost saturated (Bengio, Simard, & Frasconi, 1994), they are at a possible disadvantage during optimization since the gradient is zero when the unit is not active (Glorot et al., 2011; Maas et al., 2013). This

may lead to cases where units never get activated, since popular gradient descent optimization algorithms fine-tune only the weights of units previously activated. Thus, similar to the vanishing gradient problem, ReLUs suffer from slow convergence when training networks with constant zero gradients. To compensate for this, Maas et al. (2013) introduced leaky rectified linear units (LReLU), which allow for small nonzero gradients when the unit is not active yet is saturated. Mathematically, the LReLU is given by

$$f(x) = \max(x, 0) + \lambda \min(x, 0), \tag{5.8}$$

where λ is a predefined parameter within the range $(0, 1)$. LReLUs were initially applied to acoustic models (Maas et al., 2013); however, Xu et al. (2015) found that they perform slightly better than ReLU for image classification tasks after conducting an empirical evaluation on the CIFAR-10 and CIFAR-100 data sets (Krizhevsky, 2009). A measure on the modern ImageNet (Russakovsky et al., 2015) will facilitate the comparison of this rectification-based nonlinearity to other similar activations.

5.2.3 PReLU Activations. While LReLUs (Maas et al., 2013) rely on a predefined parameter to compress the negative part of the activation signal, He et al. (2015a) proposed a parametric rectified linear unit (PReLU) to adaptively learn the parameters of the activation units during backpropagation. Mathematically, the PReLU is the same as the LReLU, except that λ is replaced with the learnable λ_k, which is allowed to vary for different input channels, denoted by k. Thus, the PReLU can be expressed as

$$f(x_k) = \max(x_k, 0) + \lambda_k \min(x_k, 0). \tag{5.9}$$

Utilizing a previously designed DCNN model and training implementation (He & Sun, 2015), He et al. (2015a) compared the performance of ReLUs (Nair & Hinton, 2010) to PReLUs and found a greater than 1% performance increase on the ILSVC data set (Russakovsky et al., 2015). Furthermore, when this method was combined with a robust weight initialization method that specifically considered the rectified nonlinearities, they surpassed human-level performance for the first time on this challenging benchmark (Russakovsky et al., 2015). At the time, their results were the state of the art on this data set, and although outside of the yearly competition, they beat the winning entry from 2014 (Simonyan & Zisserman, 2014). Despite this, Xu et al. (2015) found that the PReLU always performed better than other rectified units, such as the ReLU (Nair & Hinton, 2010) and LReLU (Zeiler & Fergus, 2014), on the training set, thus alerting to the fact that they suffer from a severe overfitting problem on smaller data sets.

5.2.4 APL Activations. Similar to PReLUs (He et al., 2015a), Agostinelli, Hoffman, Sadowski, and Baldi (2014) concurrently proposed adaptive piecewise linear (APL) activation functions, which are parameterized, learn independently for every neuron by using conventional gradient descent, and can represent both convex and nonconvex functions of the input. Mathematically, the APL is expressed as the sum of hinge-shaped functions,

$$h_i(x) = \max(0, x) + \sum_{s}^{S=1} a_i^s \max(0, -x + b_i^s), \tag{5.10}$$

where S is the number of hinges, which is a hyperparameter set in advance, and the variables a_i^s, b_i^s for $i \in 1, \ldots, S$ are learned during training. In equation 5.7, the a_i^s variables control the slopes of the linear segments, while the location of the hinges is determined by the b_i^s variables. Although they obtained new state-of-the-art results on the CIFAR-10 and CIFAR-100 data sets (Krizhevsky, 2009), beating the high-performing NIN (Lin et al., 2013), unlike PReLUs (He et al., 2015a), their image classification experimentation did not include the challenging ILSVRC data set (Russakovsky et al., 2015). Thus, the relative performance between these similar techniques cannot be evaluated.

5.2.5 RReLU Activations. In order to address the overfitting problem associated with the PReLU (He et al., 2015a), the randomized rectified linear unit (RReLU) was proposed in a Kaggle National Data Science Bowl Competition (National Data Science Bowl | Kaggle, 2016). For RReLUs, the negative components of the activation function are randomly selected from a uniform distribution during training. During testing, they are averaged, similar to the Dropout technique (Hinton et al., 2012; Srivastava et al., 2014) before being fixed, thus allowing them to obtain a deterministic result (Xu et al., 2015). Mathematically, the PReLU can be expressed as

$$f(x_k^{(n)}) = \max(x_k^{(n)}, 0) + \lambda_k^{(n)} \min(x_k^{(n)}, 0), \tag{5.11}$$

where $\lambda_k^{(n)}$ denotes the randomized sampled parameter on the kth channel of the nth example. On the CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), and a private plankton classification data set (National Data Science Bowl | Kaggle, 2016), their classification accuracy outperformed the ReLU (Nair & Hinton, 2010), LReLU (Maas et al., 2013), and PReLU activations (He et al., 2015a).

5.2.6 ELU Activations. While ReLUs (Nair & Hinton, 2010), LReLUs (Maas et al., 2013), and PReLUs (He et al., 2015a) are all nonsaturating and thus lessen the vanishing gradient problem (Bengio et al., 1994), only ReLUs

ensure a noise-robust deactivation state (Nair & Hinton, 2010; Clevert, Unterthiner, & Hochreiter, 2016); however, they are nonnegative and thus have a mean activation larger than zero. To deal with this, Clevert et al. (2016) proposed the exponential linear unit (ELU), which has negative values to allow for activations near zero, but also saturates to a negative value with smaller arguments. Since the saturation decreases the variation of the units when deactivated, the precise deactivation argument becomes less relevant, thereby making ELUs robust against noise. Formally:

$$f(x) = \max(x, 0) + \min(\lambda(e^{x-1}), 0), \tag{5.12}$$

where λ is a predetermined parameter that controls the amount an ELU will saturate for negative inputs. ELUs sped up DCNN learning and led to higher classification accuracy when compared to other activation functions such as ReLUs. In particular, among other successes, they set a new record on the CIFAR-100 data set (Krizhevsky, 2009), beating the previous best obtained by the fractional max pooling DCNNs proposed by Graham (2014), and they obtained encouraging convergence speeds on the stimulating ImageNet (Russakovsky et al., 2015). Even though these activations provide promising image classification results and considerably reduce the computational strain on DCNNs, further experimentation using them, in particular with different architectures, is required.

5.2.7 SReLU Activations. Despite the successes of ReLUs (Nair & Hinton, 2010), LReLUs (Maas et al., 2013), and PReLUs (He et al., 2015a), they all have a limited ability to learn nonlinear transformations. Specifically, since all of these activations are convex, they are not able to learn nonconvex functions. Although the APL activation can approximate convex functions, it does so with inappropriate constraints that undermine its representation ability. To alleviate these concerns, taking inspiration from psychophysics and neural sciences, Jin et al. (2015) proposed a new type of activation, called the S-shaped rectified linear unit (SReLU). This activation combines three linear functions and performs a $\mathbb{R} \to \mathbb{R}$ mapping with the following mathematical expression:

$$f(x_i) = \begin{cases} t_i^r + a_i^r(x_i - t_i^r), & x_i \geq t_i^r \\ x_i, & t_i^r > x_i > t_i^l \\ t_i^l + a_i^l(x_i - t_i^l), & x_i \geq t_i^l \end{cases}, \tag{5.13}$$

where $\{t_i^r, a_i^r, t_i^l, a_i^l\}$ are the learnable parameters used to model each individual SReLU activation unit, and the subscript i indicates that SReLUs are allowed to vary on different input channels. Briefly, in the positive direction, when the inputs exceed the threshold t_i^r, the slope of the right line of the activation curve is given by a_i^r, while t_i^l represents a symmetrical

threshold in the negative direction. For inputs smaller than t_i^l, the left line of the activation curve calculates the outputs. For inputs within the range (t_i^l, t_i^r), the outputs are linear functions with a slope of one and no bias. SReLUs were incorporated into the state-of-the-art models of Lin et al. (2013) and Szegedy, Liu, et al. (2015), and the empirical results attained, after several image classification experiments on the MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), and ILSVRC (Russakovsky et al., 2015) data sets, illustrated their superiority over several other high-performing activations (Nair & Hinton, 2010; Goodfellow et al., 2013; Maas et al., 2013; He et al., 2015a; Xu et al., 2015). With this promising outcome, it is interesting to think about whether future deep learning advances will also rely on further inspiration from psychophysics and neural sciences.

5.2.8 Maxout and Probout Activations. Goodfellow et al. (2013) proposed an alternative to the several rectification-based activation units called Maxout, which are activations that output the maximum value from a set of inputs. For a given input $x \in \mathbb{R}^d$, a hidden layer in a Maxout network implements the following activation function,

$$f(x_i) = \max_{j \in [1,k]} x^T W_{\cdots ij} + b_{ij}, \tag{5.14}$$

where $W \in \mathbb{R}^{d \times m \times k}$ and $b \in \mathbb{R}^{m \times k}$ are both learnable parameters. Specifically, for a DCNN, a Maxout feature map can be attained by taking the maximum across k affine feature maps, which corresponds to a subspace pooling across channels in additional spatial locations. In addition to achieving state-of-the-art image classification results on several popular benchmarks, the authors provided empirical proof that Maxout was well suited for DCNN training with Dropout (Hinton et al., 2012; Srivastava et al., 2014), and that it aided in model averaging and DCNN optimization. Despite this, it suffers from the same faith as ReLUs (Nair & Hinton, 2010), LReLUs (Maas et al., 2013), and PReLUs (He et al., 2015a) concerning their inability to learn nonconvex functions; furthermore, it requires a large number of extra parameters, which increases the storage and memory costs and requires a significantly longer training time (Jin et al., 2015).

As mentioned above, Maxout (Goodfellow et al., 2013) performs a subspace pooling operation over a group of linear transformations, and this makes it partially invariant to variations in the input. In order to improve this invariance property and maintain the desirable properties of Maxout units, Springenberg and Riedmiller (2013) proposed a probabilistic variant of Maxout, called Probout, in which the maximum operator is replaced with a probabilisti sampling technique. On the CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) and SVHN (Netzer et al., 2011) data sets, Probout activations achieved better classification results compared to Maxout activations; however, in comparison to their competitors, they have a greater

computational burden due to their inherent yet computationally expensive probability calculations.

5.2.9 Analysis and Outlook. Using the correct nonlinear activation for a specific task improves the classification accuracy and computational performance of a DCNN. Although our initial models used sigmoid activations, they saturate during backpropagation, which exterminates the local gradient, and they negatively affect network dynamics during gradient descent, since their outputs are nonzero centred. This led to the development of the popular ReLU (Nair & Hinton, 2010) activation, which significantly accelerates network convergence. However, when large gradients pass through them during network training, they may irreversibly perish, and this led to other innovations such as the ReLU (Maas et al., 2013), PReLU (He et al., 2015a), and APL activations (Agostinelli et al., 2014). Despite the promising empirical results of these methods, further investigation to gauge their reliability on diverse classification tasks still needs to be conducted. ELU Clevert et al. (2016) and SReLU (Jin et al., 2015) activations have addressed other shortcomings of ReLUs, such as their positive mean activation and inability to deal with nonconvex functions, but their consistency is also relatively uncertified. Furthermore, activations such as Maxout and Probout appear particularly well suited to training DCNNs with Dropout (Hinton et al., 2012), but they require a high number of parameters and can be computationally exorbitant, therefore advocating the need for further innovations in this area.

From this analysis, we can conclude there is no clear-cut solution for which activation to use for a specific task, but a trial-and-error approach starting with ReLUs and progressing to the other activations and that monitors their shortfalls against required performance can be adopted. While an interesting future direction is to combine the use of different activation functions within the same DCNN model for their diverse benefits, a detailed theoretical analysis of why our current rectification-based activations succeed empirically is also of paramount importance. To supplement this, although it has been proven by Baldi and Sadowski (2013, 2014) that any continuous, twice-differentiable piece-wise activation function, of which ReLUs are a special case, can be used in conjunction with the Dropout model averaging technique (Hinton et al., 2012; Goodfellow et al., 2013; Srivastava et al., 2014; see sections 5.4.1 and 5.4.3), the effect of the different activations on the generalization characteristics of Dropout or even other regularization techniques still needs to be fundamentally analyzed, thus opening up the door for stimulating future work.

5.3 Supervision Component. After the innovative work of Krizhevsky et al. (2012), the preceding unsupervised DCNN pretraining methods (Ranzato et al., 2006, 2007; Weston et al., 2008; Ahmed et al., 2008; Jarrett et al., 2009; Lee et al., 2009; LeCun et al., 2010) were largely abandoned for fully

supervised training. In general, learning in DCNNs is achieved by minimizing a specific loss function, with the most common classification loss being the softmax loss (Krizhevsky et al., 2012; Lin et al., 2013; Goodfellow et al., 2013; Zeiler & Fergus, 2013, 2014; Chatfield et al., 2014; Simonyan & Zisserman, 2014; Szegedy, Liu, et al., 2015; Szegedy, Vanhoucke, et al., 2015; He et al., 2015a, 2015b). In this section, we briefly introduce this loss and deal with some motivations for using alternative losses in DCNNs.

5.3.1 Softmax Loss. The softmax activation function is widely used in the last fully connected layer of DCNNs, owing to its simplicity and probabilistic interpretation. When this activation function is combined with the cross-entropy loss (or multinomial logistic regression) in the last fully connected layer of a DCNN, they form the extensively used softmax loss. Formally, for the ith input feature x_i that has a corresponding label y_i, the softmax loss can be written as

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right), \tag{5.15}$$

where the jth element ($j \in [1, K]$, K is the number of classes) of the vector of class scores f is represented by f_j, and N is the amount of training data. For this loss, f is typically the activations of a fully connected layer W; thus, f_{y_i}, can be denoted as $f_{y_i} = W_{y_i}^T x_i$ in which W_{y_i} is the y_ith column of W (Liu, Wen, Scut, Yu, & Yang, 2016).

5.3.2 Contrastive and Triplet Losses. In order to enforce further intraclass compactness and interclass separability, and thus reinforce DCNNs with more discriminative information, the contrastive loss, also called the margin-based loss (Hadsell, Chopra, & LeCun, 2006), and the triplet loss (Schroff, Kalenichenko, & Philbin, 2015), were independently proposed (Liu, Wen et al., 2016). The contrastive loss was first implemented in a Siamese DCNN to reduce the dimensionality of data by learning mappings that are invariant to geometric distortions (Hadsell et al., 2006), while the embedded DCNNs described by Weston et al. (2008), combined it with the hinge loss for image classification and semantic role labeling tasks. Other image classification–related applications that have used the contrastive loss as part of DCNN architecture include face representation (Sun, Chen, Wang, & Tang, 2014) and visual similarity for visual search (Bell & Bala, 2015), where the contrastive loss was used in combination with the softmax loss. Furthermore, it was also used for image instance retrieval (Lin, Morere, Chandrasekhar, Veillard, & Goh, 2015), in which the authors proposed a double-margin loss. For the contrastive loss, the loss function runs over pairs of samples, which is dissimilar to conservative systems, where it is the run over individual samples. Formally, as introduced by Hadsell et al.

(2006), for a pair of input vectors $\overrightarrow{X_1}, \overrightarrow{X_2} \in I$, with binary label Y(*if* $Y = 0$ $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$ are deemed similar and $Y = 1$ if dissimilar), the general form of the contrastive loss is

$$L = \sum_{i=1}^{P} L(W, (Y, \overrightarrow{X_1}, \overrightarrow{X_2})^i) = (1 - Y)L_S(D_W^i) + YL_D(D_W^i), \tag{5.16}$$

where $D_W(\overrightarrow{X_1}, \overrightarrow{X_2})$ is written as D_W (to shorten notation), $(Y, \overrightarrow{X_1}, \overrightarrow{X_2})^i$ is the ith labeled sample pair, the partial loss function for a pair of similar points and dissimilar points is represented by L_S and L_D, respectively, and P represents the number of training pairs.

The triplet loss for DCNNs (Schroff et al., 2015), which was previously used for large-margin nearest-neighbor classification (Weinberger, Blitzer, & Saul, 2005), requires training samples in multiples of three. It minimizes the distance between a shared identity anchor sample and a positive sample while maximizing the distance between the anchor sample and a negative sample that has a different identity. Formally, for face classification, the minimized loss L is then

$$L = \sum_{i}^{N} [\| f(x_i^a) - f(x_i^a)\|_2^2 - \| f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+, \tag{5.17}$$

where x_i^a, is an anchor image of a specific person, x_i^p are positive images of the same person, negative images of any other person are denoted by x_i^n, α is the enforced margin between positive and negative pairs, and N is the cardinality of all the possible triplets in the training set. For the triplet loss, the anchor image needs to be closer to all other positive images of the same person than it is to any negative image of any other person. By combining the triplet loss with embedded image mappings, optimized by a DCNN, to a compact Euclidean space in which distance corresponds directly to face similarity, Schroff et al. (2015) achieved the best published results on the popular Labelled Faces in the Wild (LFW; Huang, Ramesh, Berg, & Learned-Miller, 2007) and YouTube Faces (Wolf, Hassner, & Maoz, 2011) databases. These results were a significant improvement compared to the previous best error rates reported in the literature (Sun, Wang, & Tang, 2015).

The LWF data set (Huang et al., 2007) was introduced in 2007 and has since become the de facto academic standard for face verification and identification (Sun, Chen et al., 2014; Taigman, Yang, Ranzato, & Wolf, 2014; Schroff et al., 2015; Zhou, Cao, & Yin, 2015). Initially, most face verification and identification attempts on this data set used individual or combined feature extractors, with the leading systems (Barkan, Weill, Wolf, & Aronowitz, 2013; Cao, Wipf, Wen, Duan, & Sun, 2013; Chen, Cao, Wen, & Sun, 2013) using greater than 10,000 image descriptors. However, more

recently, fully supervised DCNNs have been at the heart of most of the top-achieving systems, as illustrated by Table 3. The table compares the accuracy of the highest-performing DCNNs against human-level performance (HLP; Kumar, Berg, Belhumeur, & Naya, 2009). For significance, only the results published in academic papers are included; further results, including non-DCNN techniques, are deliberated in the data sets–related survey paper (Learned-Miller, Huang, RoyChowdhury, Li, & Hua, 2016). Despite these successes, the mechanisms used by the human brain to be able to easily identify and recognize faces, in a short period of time, are still at large. Interestingly, it may be possible that the central nervous system has evolved to process faces in a different fashion when compared to objects (Leibo, Mutch, & Poggio, 2011), and thus future face classification and recognition DCNN models may need to incorporate this type of evidence.

5.3.3 Large Margin Loss. Asserting that a larger angular similarity will lead to larger angular seperability between learned features, which in turn will result in the generation of more discriminative features, Liu, Wen et al. (2016) introduced an angular margin between the input feature vector and the weight matrix for a more general, large-margin softmax loss, which they called the large-margin softmax (L-Softmax). Formally, the L-Softmax is defined as

$$L\text{-}Softmax = -\log\left(\frac{e^{\|W_{y_i}\|\|x_i\|\psi(\theta_{Y_i})}}{e^{\|W_{y_i}\|\|x_i\|\psi(\theta_{Y_i})+\sum_{j\neq y_i} e^{\|W_j\|\|x_i\|\cos(\theta_j)}}}\right), \tag{5.18}$$

in which

$$\psi(\theta) = \begin{cases} \cos(m\theta), & 0 \leq \theta \leq \frac{\pi}{2} \\ D(\theta), & \dfrac{\pi}{m} \leq \theta \leq \pi \end{cases}, \tag{5.19}$$

where θ_j is the angular margin, $a^{(i)}$ is the input vector, ω_j is the jth column of the weight matrix, and m regulates the margin among classes. In addition to fashioning more discriminative features, other desirable advantages of the L-Softmax include the fact that its geometric interpretation is very clear and that it partially avoids overfitting. When applied to image classification tasks, it outperformed the original softmax loss (for the same architecture) and achieved results on par with the state of the art for the MNIST data set. It also accomplished new state-of-the-art results on the CIFAR-10 and CIFAR-100 data sets (LeCun et al., 1998; Krizhevsky, 2009; Bulo & Kontschieder, 2014; Liang & Hu, 2015; Liu, Wen, et al., 2016).

While others have proposed using a specially designed hybrid loss function that combines the negative logarithm of a normalized value with weighted errors of varying order to improve the adversarial robustness

Table 3: DCNN Performance on LFW Data Set.

Model	Description	Number of Layers	Number of Models	Accuracy	H.P	Reference
Canonical view DCNN	DCNN, PCA, SVM	7	60	96.45 ± 0.25	97.53	Zhu, Luo, Wang, & Tang, 2014
DeepFace	3D Face modelling, DCNN	8	4	97.35 ± 0.25	97.53	Taigman et al., 2014
DeepID	DCNN, PCA	7	60	97.45 ± 0.26	97.53	Sun, Wang, & Tang, 2014
DeepID2	DCNN, PCA, combined loss	7	25	*99.15 ± 0.15*	97.53	Sun, Chen et al., 2014
DeepID2+	DCNN, PCA, loss at multiple layers	7	25	*99.47 ± 0.12*	97.53	Sun, Wang, & Tang, 2015
Face++	DCNN, PCA	10	1	*99.50 ± 0.36*	97.53	Zhou et al., 2015
DeepID3	DCNN, Inception layers, PCA	10–15	25	*99.53 ± 0.10*	97.53	Sun, Liang et al., 2015; Szegedy, Vanhoucke et al., 2015
FaceNet	DCNN, triplet loss, embedded images	22	1	*99.63 ± 0.09*	97.53	Schroff et al., 2015

Note: The accuracy of the models in italics has surpassed human-level performances.

(see section 6.4) of DCNNs (Zhao & Griffin, 2016), the application of the L-Softmax loss to this specific challenge is still at large. Specifically, since this loss results in the generation of more discriminative features, its application to the challenge of adversarial examples will contribute to understanding if holistic changes to the way we train DCNNs are required to address their remaining challenges.

5.3.4 L2-SVM Loss. While SVMs have previously been used in combination with CNNs (i.e., by replacing the softmax layer with an SVM) for improved classification performance (Huang & LeCun, 2006; Lee et al., 2009; Coates et al., 2011), the downside is that the lower-level features of the CNN are not learned with respect to the SVM's objective. To address this issue, Collobert and Bengio (2004) and Nagi, Di Caro, Giusti, Nagi, and Gambardella (2012) proposed joint training at the lower levels by, respectively, introducing new cost functions to integrate SVMs with MLPs and CNNs. Inspired by this, Tang (2013) also proposed integrating SVMs with DCNNs but they replaced the standard SVM hinge loss (L1-SVM) with the L2-SVM loss (Hinton, 1989). When compared to the L1-SVM loss, the L2-SVM loss is differentiable and penalizes errors more profoundly. SVMs were initially formulated for binary classification. Therefore, given training samples and their corresponding labels (x_n, y_n), $n = 1, \ldots, N$, $x_n \in \mathbb{R}^D$, $t_n \in \{-1, +1\}$, the L2-SVM minimizes the squared hinge loss, denoted formally by the following unconstrained optimization problem,

$$\min_{w} \frac{1}{2} w^T W + C \sum_{n=1}^{N} \max\left(1 - w^T x_n t_n, 0\right)^2, \tag{5.20}$$

where W is the weight connecting the penultimate layer to the softmax layer. The class label for test data x can be predicted by $arg\ \max_t (w^T x)t$, while for a multiclass SVM (Vapnik, 1995), where the output of the kth SVM is denoted as $a_k(x) = w^T x$, the predicted class is $arg \max_k a_k(x)$. When compared to the conventional softmax loss, for the same DCNN architecture, the L2-SVM loss showed improved classification performance on the CIFAR-10 data set (Krizhevsky, 2009), obtaining results comparable with the current (at the time) state of the art, which utilized a much more complex model that included contrast normalization layers and Bayesian parameter fine tuning (Snoek, Larochelle, & Adams, 2012).

5.3.5 Analysis and Outlook. The softmax loss is an extremely popular choice for CNNs owing to its simplicity, probabilistic elucidation, and the intuitive output it produces. However, to furnish CNNs with the ability to extract more discriminative features, other losses, such as the contrastive loss (Hadsell et al., 2006) and the triplet loss (Schroff et al., 2015), were suggested. Although these losses encourage discriminative learning, a consequent problem is that in theory, the number of required training pairs or

triplets can go up to $O(N^2)$, where N is the total number of training samples. Moreover, for a large data set, like the data set used for the ILSVRC (Russakovsky et al., 2015), which consists of over 1 million images, the subset of training samples will require careful online or offline selection for both of these losses. This led to the recently proposed coupled clusters loss (see Liu, Tian, Yang, Pang, & Huang, 2016), which accelerated network convergence and stabilized the training process. Although it produced promising vehicle reidentification results, more traditional classification tasks are yet to be tested. Despite its benefits and well-established and acceptable performance, the softmax loss does not explicitly encourage intraclass compactness and interclass separability. It uses the cosine distance between classes for its classification score; thus, the predication of a label for a given input is determined predominantly by the angular similarity to each class. This inspired the proposal of the L-Softmax loss (Liu, Tian et al., 2016), which still needs to be commissioned on DCNN's open issues (see sections 5.3.3 and 6.4). Integrating the L2-SVM loss, traditionally associated with SVMs, facilitated improved classification accuracy, but like the coupled clusters loss, its consistency across diverse tasks remains unknown.

Despite the innovations summarized above, the softmax loss remains a reputable choice for traditional academic benchmarks such as MNIST (LeCun et al., 1998) and ImageNet (Russakovsky et al., 2015), or other tasks where a single output class (label) per image is required. For real-world tasks requiring multiple classes per image, per class multiple logistic regression is recommended as a starting point. Based on the requirements of the task, experimentation with the other losses mentioned in this section can be explored. For example, fine-grained classification could benefit particularly by employing the coupled clusters loss, while for face verification or other verification tasks not constrained by computational resources, the triplet loss could produce excellent verification performance. In closing, it is recommended that future work should challenge the development of novel loss functions that address DCNN's open issues, supporting the work proposed by Zhao and Griffin (2016), while the use of other multiclass SVM formulations, or even other classifiers such as RBFs, which further investigate the performance improvements presented by LeCun et al. (1998) and Tang (2013), should also be explored.

5.4 Regularization Mechanisms. DCNNs are very expressive models, capable of learning exceptionally complicated relationships between their inputs and outputs. However, with limited training data, even for larger data sets (Krizhevsky et al., 2012), many of these complicated mappings are due to sampling noise. Thus they exist in the training set rather than in the test set, irrespective of whether they are drawn from the same data distribution. This leads to overfitting, which can be mitigated by regularization. Although the easiest and most common method to reduce overfitting is data augmentation (LeCun et al., 1998; Simard et al., 2003; Ciresan et al., 2011, 2012; Krizhevsky et al., 2012; Montavon et al., 2012; Chatfield et al.,

2014), it requires a larger memory footprint and comes at a higher computational cost (Szegedy, Liu, et al., 2015). Furthermore, despite the regularization effects of several other diverse methods, including L_1 and L_2 regularization, stopping training early, stochastic pooling (Zeiler & Fergus, 2013), unique activation functions (He et al., 2015a; Xu et al., 2015), model averaging (Goodfellow et al., 2013; Srivastava et al., 2014), novel loss functions (Liu, Wen et al., 2016), and soft-weight sharing (Nowlan & Hinton, 1992), the successful application of Dropout (Hinton et al., 2012; Srivastava et al., 2014) to DCNNs (Krizhevsky et al., 2012) has led to its extensive use and inspired numerous improvements. We next provide a formal description of Dropout and discuss several of its variants. We also introduce some of the latest regularization developments that can be used in conjunction with Dropout.

5.4.1 Dropout. In Dropout (Hinton et al., 2012; Srivastava et al., 2014), each unit of a layer's output is retained with probability p; else, it is set to zero with probability $1 - p$, with 0.5 being a common value of p (Krizhevsky et al., 2012; Hinton et al., 2012). When Dropout is applied to a fully connected layer of a DCNN (or any DNN), the output of the layer $r = [r_1, r_{2,}, \ldots, r_d]^T$, can be expressed as

$$r = m \star a(W_v), \tag{5.21}$$

where $\star$ denotes the element-wise product between a binary mask vector m and the matrix product between the input vector $v = [v_1, v_{2,}, \ldots, v_n]^T$ and the weight matrix W (with dimensions $d \times n$), followed by a nonlinear activation function, a. In equation 5.16, the binary mask vector has size d, and each element j is drawn independently from a $Bernoulli(p)$ distribution m_j, while the biases are included in W and fixed to one for simplicity (Wan et al., 2013). The primary benefit of Dropout is its proven ability to significantly reduce overfitting by effectively preventing feature coadaptation (Hinton et al., 2012); it is also capable of attaining model averaging (Goodfellow et al., 2013; Srivastava et al., 2014). Further, to the various improvements and Dropout variants discussed below, Wager, Wang, and Liang (2013) highlighted its adaptive regularization characteristics; its efficiency and ensemble learning characteristics were examined by Warde-Farley, Goodfellow, Courville, and Bengio (2013), while Baldi and Sadowski (2013, 2014) provided a detailed mathematical analysis of its static and dynamic properties and characterized its averaging properties for DNNs by formal recursive equations.

5.4.1.1 Fast dropout. Despite the highlighted advantages of Dropout (Hinton et al., 2012; Srivastava et al., 2014), the actual sampling or training of multiple models makes training slower. Furthermore, in the case of

nonredundant data, depending on the how the data are sampled, training efficiency may be further reduced. To assuage these concerns yet still achieve the advantages of Dropout training without actually sampling, and thus use all the data efficiently, Wang and Manning (2013) proposed Fast Dropout. Fast Dropout training is accomplished by sampling from or integrating with a gaussian approximation, which is justified by the central limit theorem and empirical evidence. Specifically, when Fast Dropout is integrated with the commonly used softmax loss, the loss can be computed by the following loss function:

$$L = E_{s \sim N(\mu, \sum)} \left[\sum_{i=1}^{|y|} t_i \log(softmax(S)_i) \right], \tag{5.22}$$

where samples are taken directly from an input gaussian approximation, with $S \in \mathbb{R}^{|y|}$, and the set y represents all the possible predications. Fast Dropout can also be integrated with the hinge loss traditionally associated with SVMs (see section 5.3.4) and the Maxout technique (Goodfellow et al., 2013) and has produced promising results on regression, document classification, and, most significant, noteworthy speed gains on image classification tasks benchmarked on the CIFAR-10 (Krizhevsky, 2009) and MNIST data sets (LeCun et al., 1998). Although training with backpropagation is possible with certain limitations (see section 5.4.3), further research to assert its benefits and shortcomings when applied to different DCNN architectures is still required.

5.4.1.2 Adaptive dropout (standout). Since Dropout (Hinton et al., 2012) uses a constant probability to randomly drop units, it is conceivable that even units that can make confident predictions for the presence or absence of a feature will be dropped 50% (if $p = 0.5$) of the time. Motivated to improve this, Ba and Frey (2013) presented a Dropout variation, called Standout, in which a binary belief network that shares parameters with a deep network computes the Dropout probability for each hidden unit. More specifically, the dropout probability is adaptive, and unlike in standard Dropout, where the unit activity is masked by a $Bernoulli(p)$ distribution m_j, with probability 0.5, in Standout it depends on the input activities

$$P(m_j = 1 | \{a_i : i < j\}) = f\left(\sum_{i:i<j} \pi_{j,i} a_{,i} \right), \tag{5.23}$$

where the weight from unit i to unit j in the adaptive dropout network is denoted by $\pi_{j,i}$ and $f(\cdot)$ is a sigmoidal function, with $f : \mathbb{R} \to [0, 1]$. With this method, units that make confident predications for the presence of

features have a higher probability of being retained, and vice versa. The paper's empirical work on popular classification benchmarks did not include experiments on DCNNs; however, since Standout is designed to work with backpropagation, using stochastic gradient descent, it seems plausible to incorporate it into DCNN architecture for image classification–related applications.

5.4.1.3 Multinomial dropout and evolutional dropout. Asserting that standard Dropout (Hinton et al., 2012) resulted in suboptimal convergence and that it was more logical to use nonuniform multinomial sampling probabilities for different neurons and their associated features, Li, Gong, and Yang (2016) freshly proposed multinomial dropout. More specifically, to determine the optimal Dropout probabilities rather than the original technique that determined them independently and identically and to justify the application of multinomial sampling to shallow learning systems, they formally established a risk bound for stochastic optimization with multinomial dropout. This allowed them to attain, by minimizing a sampling reliant factor from the risk bound, a distribution-dependent dropout. This distribution-dependent dropout demanded sampling probabilities that were based on the second-order statistics of the data distribution. Based on this multinomial distribution-dependent dropout, they proposed an efficient yet adaptive Dropout version called Evolutional Dropout, with the objective of solving the deep learning issue of internal covariate shift, which is discussed further in section 5.5.3.

The dropout probabilities for Evolutional Dropout can be computed by the following expression,

$$p_i^l = \frac{\sqrt{\frac{1}{m}\sum_{j=1}^{m}[\mathrm{X}_j^l]_i^2]}}{\sum_{i'}^{d}\sqrt{\frac{1}{m}\sum_{j=1}^{m}[\mathrm{X}_j^l]_{i'}^2]}}, i = 1, \ldots, d, \tag{5.24}$$

where the probabilities p_i^l evolve as the output layers' distribution evolves (hence, the name *Evolution*), and the outputs of the lth layer, for a minibatch of m examples is represented by $X^l = (X^l, \ldots, X_m^l)$. On popular image classification benchmarks such as MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), and SVHN (Netzer et al., 2011), they provided empirical proof that these latest Dropout additions lead to faster convergence and smaller testing error when compared to vanilla Dropout, advocating the need for further investigation.

5.4.1.4 Spatial dropout. In an object localization application that used a DCNN, the authors found that applying regular Dropout before a 1×1 convolution layer (see the paper for detailed architecture) increased the training time but did not prevent overfitting. Thus, they proposed Spatial Dropout

(Tompson, Goroshin, Jain, LeCun, & Bregler, 2015). Specifically, for a given convolutional feature tensor, with dimensions $n_{feats} \times height \times width$, they performed only n_{feats} Dropout and used the entire feature map to extend the Dropout value. Subsequently, adjacent pixels in a dropped-out feature map were either all zero (omitted) or all active. Initial results indicate that Spatial Dropout is well suited to a data set that has a small number of training samples, thus making it a good candidate to reduce overfitting for smaller data sets, where generalization is usually an issue. Furthermore, although the method demonstrated promising results in human pose and joint motion estimation, further work is warranted for classification-specific tasks. In particular, its application to fine-grained classification seems conceivable, since the technique recovers information lost during the pooling operation without losing the computational gains achieved by pooling.

5.4.1.5 Nested dropout. To learn ordered representations of data in which different dimensions have different degrees of importance, such that the information contained in each dimension of the representation decreases as a function of the dimension index according to a predefined decay function, Rippel et al. (2014) proposed Nested Dropout. Nested Dropout randomly draws unit indices from a geometric distribution. Rather than independently drop units with a predefined probability, as in standard Dropout (Hinton et al., 2012) it omits all the units that follow the drawn number. More specifically, for a representation space with dimension K, a distribution $pB(\cdot)$ defined over the representation index subset $S_b = \{1, \ldots, b\}, b = 1, \ldots, K$ has the characteristic that if the jth unit appears in a particular mask, then all the preceding units $1, \ldots, j-1$, do so as well, thereby allowing the jth unit to rely on them. Thus, while Dropout enforces a distribution over each individual unit in a model, Nested Dropout assigns a distribution over nested subsets of representation units. Inspired by its application to unsupervised autoencoders (Rippel et al., 2014), Finn et al. (2015) used it to train, by standard backpropagation, compact DCNNs that adapt to different tasks and data complexity.

5.4.1.6 Max pooling dropout. Dropout was initially designed to work on the fully connected layers of deep architectures (Krizhevsky et al., 2012; Hinton et al., 2012; Wan et al., 2013), with little attention paid to the other layers. Motivated by this, empirical work by Wu and Gu (2015) found that the effect of Dropout on the max pooling layers of DCNNs is equivalent to randomly picking an activation based on a multinomial distribution at training time, which is similar in nature to stochastic pooling (Zeiler & Fergus, 2013). Thus, to get a more accurate approximation of averaging all possible Dropout units, they proposed a probabilistic weighted pooling scheme instead of the commonly used max pooling. For the proposed scheme, the pooled activity of all activations in each region is computed by

$$a_j^{(l+1)} = \sum_{i=0}^{n} p_i a_i^{\prime(l)} = \sum_{i=1}^{n} p_i a_i^{\prime(l)}, i \in R_j^{(l)}, \tag{5.25}$$

where the pooling region j at layer l is represented by $R_j^{(l)}$, and p_i is the probability computed by

$$\Pr(a_j^{(l+1)} = a_i^{\prime(l)} = p_i = pq^{n-i}, (i = 1, 2, \ldots, n), \tag{5.26}$$

in which p is the retaining property, $q = 1 - p$ is the dropout probability, and i is an index in the multinomial distribution. The proposed scheme is capable of regularization and model averaging similar to the effect of Dropout (Srivastava et al., 2014) and Maxout (Goodfellow et al., 2013). For classifications tasks on the MNIST (LeCun et al., 1998) and CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) data sets, their method outperformed max pooling and scaled max pooling, while they also found that Dropout on the max pooling layers outperformed the stochastic pooling technique (Zeiler & Fergus, 2013), discussed in section 5.1.2.2.

5.4.2 DropConnect. Another popular generalization of Dropout (Hinton et al., 2012) is DropConnect (Wan et al., 2013), which, rather than randomly dropping a subset of activations, as in traditional Dropout, randomly drops a subset of the weights with probability $1 - p$. As with Dropout, DropConnect is suitable for the fully connected layers of DNNs (DCNNs included) only. Formerly, using the same notation as equation 5.16, the output of a DropConnect layer can be expressed as

$$r = a((M \star W)_v), \tag{5.27}$$

where M is a binary mask matrix that encodes the connection information of the weights, drawn from a *Bernoulli* (p) distribution m_{ij}. During the training process, each element of the mask is drawn independently for each sample, thus resulting in a different connectivity for each example observed. Furthermore, during this process, the biases are also masked out. During inference, samples are drawn from a 1D gaussian approximation through moment matching and are averaged and presented to the next layer after being passed through the activation layer.

The DropConnect paper compared the image classification performance of Dropout and DropConnect on several popular classification benchmarks and found that DropConnect outperformed Dropout on the MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky, 2009), and SVHN (Netzer et al., 2011) data sets, while on the NORB data set (LeCun et al., 2004), Dropout produced better results. Moreover, when they combined various DCNNs trained with DropConnect into ensembles, motivated by the

voting scheme presented by Ciresan et al. (2012), they achieved new state-of-the-art results on the SVHN, MNIST and NORB, and CIFAR-10 data sets, respectively, beating the previous best results of Zeiler and Fergus (2013), Ciresan et al. (2012), and Snoek et al. (2012). Although there have been several high-performing DCNNs since (see Table 5), the DropConnect model, which used data augmentation, still holds the record for the lowest classification error on the famous MNIST benchmark. Since the empirical work of Wan et al. (2013) was conducted only on small data sets, Smirnov, Timoshenko, and Andrianov (2014) extended the comparison and found that Dropout provided better regularization than DropConnect on the much larger ILSVRC 2013 (Russakovsky et al., 2015). Figure 11 illustrates the difference between a feedforward fully connected network without Dropout, with Dropout, and with DropConnect. As illustrated, in a DropConnect network, the connections with their associated weights are randomly dropped rather than the nodes.

5.4.3 Recent Regularization Advances. Further to the model regularization techniques mentioned in the introduction to this section, another mostly uncharted alternative is to regularize a DCNN's output distribution. One of the signs of overfitting is when a model assigns all class probabilities to a single class from the training set. These confident estimates usually resemble low-entropy output distributions. To deal with this, Szegedy, Vanhoucke et al. (2015) introduced label smoothing regularization (LSR), which maintains a realistic ratio between the unnormalized log probabilities (logits) of erroneous classes by estimating, during training, the marginalized consequence of label dropout. This averts the model from allocating a complete likelihood for each training case. The LSR technique can be considered the equivalent of replacing a single cross-entropy loss with a pair of losses, the second of which looks at a prior distribution and penalizes the deviation of the predicted label relative to it. Inspired by the regularization effect of LSR, confident output distributions are also penalized by Pereyra, Tucker, Chorowski, Kaiser, and Hinton (2017), who present a confidence penalty based on maximum entropy supplemented by uniform and unigram label smoothing. Their technique improves several state-of-the-art models on a wide variety of tasks, which include image classification. Recently another output regularization technique, which added noise to the output layer, was also proposed (Xie, Wang, Wei, Wang, & Tian, 2016), hinting at a possible new trend to tackle overfitting.

5.4.4 Analysis and Outlook. A suitable way to regularize a model is to average the results from several different networks; however, for large DCNNs, the computational resources required to do this will be astronomical. This led to the presentation of Dropout (Hinton et al., 2012), which provided a means to roughly merge an exponential number of DCNNs in an effective manner (Hinton et al., 2012; Goodfellow et al., 2013; Srivastava et al., 2014), and this contributed to numerous empirical successes that stimulated

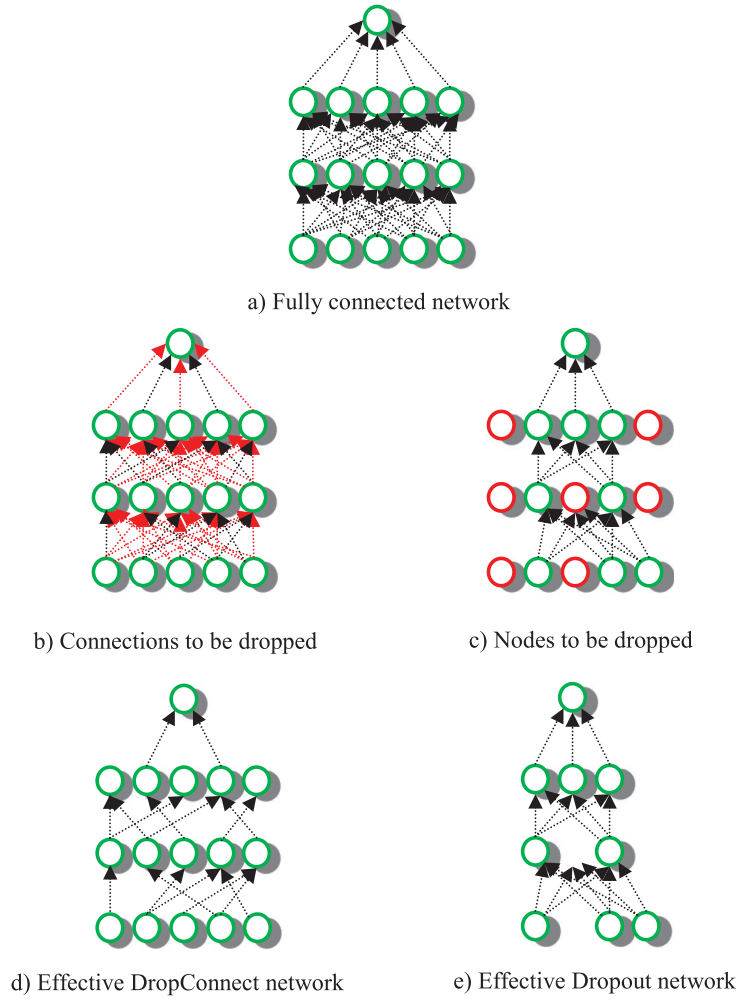

Figure 11: Difference between fully connected network layers without Dropout (a), with Dropout (b, d), and with DropConnect (c, e).

researchers to advance the technique further and investigate and mitigate its inadequacies.

To deal with the training inefficiencies associated with standard Dropout, a Fast Dropout (Wang & Manning, 2013) method capable of providing DCNNs with significant speed gains during training and inference, together with more stability, was proposed. However, the downside of this approach is that training during the backward pass of backpropagation is more complicated, while the forward pass remains straightforward. A possible solution to mitigate this is to train a DCNN with standard Dropout

for simplicity and use Fast Dropout only during inference for computational speed gains; further work to simplify the process is still required. Standout (Ba & Frey, 2013) was implemented to reduce the risk of dropping units that presented assertive predications of features, but as with Fast Dropout and Spatial Dropout (Tompson et al., 2015), although they have shown promising results for other architectures and tasks, further investigation focusing on their application to image classification tasks tackled by DCNNs is still required. In particular, using them for discriminative fine-tuning, after knowledge transfer is an interesting avenue requiring further investigation. Evolutional Dropout (Li et al., 2016), which has adaptive characteristics similar to Standout, can improve the convergence characteristics and improve the classification performance of DCNNs; however, as with Standout and Max pooling Dropout (Wu & Gu, 2015), the complementary probability computations add to the computational burden of the systems that use them. Whilst DropConnect (Wan et al., 2013) allows for the training of large models without overfitting, it is slower than models that utilize Dropout or no Dropout.

Thus, from this analysis, it can be concluded that despite the promising empirical results of the several dropout variants described in this section, further research to firmly establish them is still required. In particular, the technique can benefit the most from further innovations that focus on reducing the computational costs of systems that utilize it. Moreover, given that real intelligence is highly adaptive in nature, it is predicted that future work will incorporate adaptive physiognomies, similar to Standout and Evolutional Dropout. On the theoretical side, to supplement the work by Wager et al. (2013), Wade-Farley et al. (2013) and, most significant, Baldi and Sadowski (2013, 2014), and further promote the application of Dropout, further theoretical analysis justifying the reasons for its successes is still required. In particular, its generalization properties are yet to be proven with acceptable mathematical precision. In fact, all its variants will benefit from such scrutiny. Another promising avenue regarding further analysis, briefly touched on by Baldi and Sadowski (2014), is to investigate the duality and connections between spiking or stochastic neurons and Dropout, since there is a possibility that these can be used during learning to accomplish the same objectives of this game-changing regularization technique.

Although output regularization techniques are still in their infancy for DCNNs, the initial results are encouraging, and given that the techniques introduced are generally suitable and orthogonal to many other regularization approaches such as Dropout, further work along these lines is encouraged.

5.5 Optimization Techniques. In this section, we evaluate some of the important optimization techniques, after first examining their requirements.

5.5.1 Gradient-Based Learning. In fully supervised DCNNs, the loss function, which is in most cases the softmax loss (Krizhevsky et al., 2012; Goodfellow et al., 2013; Lin et al., 2013; Zeiler & Fergus, 2013, 2014; Simonyan & Zisserman, 2014; Chatfield et al., 2014; Szegedy, Liu, et al., 2015; Szegedy, Vanhoucke et al., 2015; He et al., 2015a, 2015b), is usually minimized using some form of stochastic gradient descent (SGD; Bottou, 1998, 2010). For this technique, the gradient is evaluated using the popular backpropagation algorithm (Ciresan et al., 2011; Krizhevsky et al., 2012; Wan et al., 2013; Goodfellow et al., 2013; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Vanhoucke et al., 2015; Szegedy, Liu, et al., 2015; He et al., 2015b; Srivastava et al., 2015a; Choromanska, Henaff, Mathieu, Arous, & LeCun, 2015). While gradient descent, made popular by Rumelhart et al. (1986), was used in many of the early CNNs (LeCun et al., 1989a, 1989b; LeCun et al., 1998), increases in data size—consider MNIST (LeCun et al., 1998) versus ILSVRC (Russakovsky et al., 2015)—and its related computational complexity have led to the popularization of SGD, which is a tremendous simplification of the traditional method (Bottou, 2010). Instead of precisely computing the gradient, a single randomly selected sample (in practice, a mini-batch of samples) is used to estimate it for each iteration, thereby making the process naturally stochastic. Significantly, SGD can process examples online (or on the fly) since it does not need to recall which examples were observed in past iterations (Choromanska et al., 2015). Furthermore, standard SGD can be implemented in parallel, across multiple GPUs, for further optimization and improved processing speeds, particularly for large-scale machine learning applications (Zinkevich, Weimer, Li, & Smola, 2010; Recht, Re, Wright, & Niu, 2011; Dean et al., 2012; Zhuang, Chin, Juan, & Lin, 2013; Bengio, 2013; Paine, Jin, Yang, Lin, & Huang, 2013). While we have given a brief introduction here, further details can be gleaned from the abundance of literature available on this optimization technique. In particular, Bottou (1998, 2010) presents a detailed analysis on SGD; Qian (1999), Zeiler (2012), Duchi, Hazan, and Singer (2011), and Kingma and Ba (2014) present other gradient descent–based optimization algorithms, while several of these alternatives are surveyed and compared by Sutskever, Martens, Dahl, and Hinton (2013). Despite the heavy use of gradient-based optimization techniques for DCNNs and image classification in general, the question remains whether these algorithms are intrinsically flawed, leading to some of the known challenges with DCNNs (see section 6). Thus, further work to understand the inner workings of our models and, in particular, our optimization techniques still needs to be conducted.

5.5.2 Enhanced Initialization Schemes. Poor initialization of DCNN parameters, which are typically in the millions (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Taigman et al., 2014; Szegedy, Liu, et al., 2015), and in particular their weights, can hamper the training process because of the vanishing/exploding gradient problem (Bengio et al., 1994), and hinder

convergence. Thus, their initialization is extremely critical (Sutskever et al., 2013; Simonyan & Zisserman, 2014; He et al., 2015a; Mishkin & Matas, 2016). Here we briefly introduce selected representative initialization schemes. Saxe, McClelland, and Ganguli (2013); Sussillo and Abbott (2014), Hinton, Vinyals, and Dean (2015), Romero et al. (2015), and Srivastava (2015a, 2015b) can be referred to for other appropriate techniques.

5.5.2.1 Xavier initialization. Glorot and Bengio (2010) evaluated how backpropagated gradients and activations varied across different layers; based on these considerations, they proposed a normalized initialization scheme that essentially adopted a balanced uniform distribution for weight initialization (He et al., 2015a). For this initialization scheme, the initial weights are drawn from a uniform or gaussian distribution, with a zero-mean and precise variance. As Glorot and Bengio (2010) recommended, the following variance can be used:

$$\mathrm{Var}(W_Init) = \frac{2}{n_x + n_y}, \tag{5.28}$$

where W_Init represents a specific neuron's distribution at initialization, n_x is the number of neurons feeding into the variance, and n_y represents the number of neurons furnished by its output. Thus, for the original technique, the number of input and output neurons controls the degree of initialization. Later, the scheme was referred to as "Xavier" initialization and simplified for easier implementation by Jia et al. (2014), in which the variance is drawn from a distribution with zero mean, and variance computed by the following expression:

$$\mathrm{Var}(W_Init) = \frac{1}{n_x}. \tag{5.29}$$

Xavier initialization promotes the propagation of signals deep into DNNs (DCNNs included) and has been shown to lead to substantively faster convergence (Glorot & Bengio, 2010). Its main limitation is that its derivation is based on the assumption that activations are linear, thus making it inappropriate for ReLU (Nair & Hinton, 2010) and PReLU activations (He et al., 2015a).

5.5.2.2 Theoretically derived adaptable initialization. To circumvent this, He et al. (2015a) derived a theoretically sound initialization that considered these nonlinear activations. Specifically, their derivation, which closely followed Glorot and Bengio (2010), led to the initialization of weights from a zero-mean gaussian distribution whose standard deviation is $\sqrt{2/n_l}$, where n is the number of connections of the response and l is the layer index.

Furthermore, they initialize the biases to zero. They showed empirical proof that this initialization scheme was suited to training extremely deep models, while Xavier initialization was not.

5.5.2.3 Standard fixed initialization. Another method made popular by Krizhevsky et al. (2012) is to initialize the weights for each layer from a zero-mean gaussian distribution, with a fixed standard deviation—0.01 in Krizhevsky et al. (2012)—and set the biases of different layers to either a constant one or zero. However, while Krizhevsky et al. (2012) found that this initialization scheme complimented ReLU activations (Nair & Hinton, 2010) and accelerated learning, others have found that for very deep models, it hinders convergence due to the magnitude of the gradients or the activations in the final layers (Simonyan & Zisserman, 2014; He et al., 2015a; Mishkin & Matas, 2016).

5.5.2.4 Layer sequential unit variance initialization. After its inception, the Xavier initialization method (Glorot & Bengio, 2010) was generalized only to ReLUS (Nair & Hinton, 2010) by He et al. (2015a), but not to other nonlinear activations like the hyperbolic tangent or Maxout (Goodfellow et al., 2013) activations. Moreover, the initial theoretical derivation did not cover the entire spectrum of DCNN layers such as max pooling (Ranzato et al., 2007) and local response normalization (Krizhevsky et al., 2012). However, instead of deriving a new theoretical formulation to cover all the remaining activation and DCNN layers, Mishkin and Matas (2016) presented a data-driven weight initialization scheme called layer-sequential unit-variance (LSUV) initialization. Briefly, they preinitialize the weights of the convolutional and inner product layers by filling the weights with gaussian noise, with unit variance. Next, they use QR or single value decomposition (SVD) to decompose the weights to an orthonormal basis, motivated by Saxe et al. (2013), and then replace them with one of the components. Thereafter, they estimate the output variance of each of the affected layers, from the first to the final layer, normalizing the weights so that the variance is equal to one. On ImageNet (Russakovsky et al., 2015), CIFAR-10 (Krizhevsky, 2009), and MNIST (LeCun et al., 1998), their fast initialization technique facilitated the training of DCNNs and led to results comparable to the state of the art, outperforming other sophisticated systems (Srivastava et al., 2015a; Romero et al., 2015) also designed for optimization.

5.5.2.5 Analysis and outlook. To escape the vanishing/exploding gradient problem (Bengio et al., 1994) and thus promote network convergence and to mitigate the challenges of training DCNNs usually imposed with nonconvex loss functions (Choromanska et al., 2015), a considerable amount of research has gone into ensuring adequate network initialization. Xavier initialization (Glorot & Bengio, 2010), which is commonly employed to initialize DNNs, facilitates fast convergence and is known to work well in many

applications (Mishkin & Matas, 2016). However, since Xavier initialization is inappropriate for rectification-based nonlinear activations, it is not well suited to modern DCNNs, which utilize them exhaustively. Furthermore, it does not promote the convergence of extremely deep networks. These deficiencies were addressed by the more robust scheme presented by He et al. (2015a); however, in extremely deep networks, although they showed improved convergence characteristics, they could not demonstrate that their initialization method led to improved accuracy, probably due to degradation (He et al., 2015b; He & Sun, 2015; Srivastava et al., 2015a, 2015b), which also encumbers standard fixed initialization. To counter this, Simonyan and Zisserman (2014) pretrain a DCNN shallow enough to have its weights randomly initialized and use this network to train deeper architectures; however, this naturally requires more training time, is more expensive computationally, and may even lead to poor convergence, advocating the need for other alternatives. The recently proposed LSUV initialization technique (Mishkin & Matas, 2016) produced promising empirical results; however, despite the practicality of the proposed data-driven approach, it still necessitates computing batch statistics, is not established on large data sets, and entails intricate procedures.

Thus, from this we can conclude that the key factors to consider when selecting an initialization scheme are the activation function to be used, the network depth, which could hamper classification accuracy due to degradation, the computational budget available, the size of the data set, and the tolerable complexity of the required solution. If the last two factors do not impose system constraints, LSUV is appealing (see section 5.5.3.2), while standard fixed initialization remains a popular choice for shallower networks (by today's standards). For extremely deep networks, experimentation with other schemes, such as those presented by Simonyan and Zisserman (2014) and He et al. (2015a), which specifically monitor for degradation, can be conducted. Upcoming work should focus on the design of generic initialization schemes that not only speed up training times and maintain accuracy by tackling degradation but are also adaptable to different models, irrespective of their depth and tasks irrespective of their complexity, while a theoretical analysis of how our current schemes optimize our modern models is also advocated. Furthermore, notwithstanding the computational implications, the optimization effect of unsupervised pretraining, supporting the work conducted by Saxe et al. (2013) and Simonyan and Zisserman (2014), also necessitates further experimentation.

5.5.3 Batch Normalization. In addition to having a large number of parameters, the training of DCNNs is convoluted by a phenomenon known as internal covariate shift, which is caused by changes to the distribution of each layer's inputs because of parameter changes in the previous layer. This phenomenon has severe consequences, which include slower training due to lower learning rates, the need for careful parameter initializations,

and complexities when training DCNNs with saturating nonlinear activations. To reduce the consequences of internal covariate shift, Ioffe and Szegedy (2015) proposed a technique known as batch normalization (BN). This technique introduces a normalization step, which is simply a nonlinear transform applied to each activation, that fixes the means and variances of layer inputs. To allow integration with SGD (Bottou, 1998, 2010), which also uses mini-batches during training, BN computes the mean and variance estimates after mini-batches rather than over the entire training set.

Specifically, for a mini-batch $B = \{x_{1,\ldots,n}\}$, with activation x and dimension n, the mini-batch mean and variances are first computed by $\mu_B \leftarrow \frac{1}{n}\sum_{j=1}^{n} x_j$ and $\sigma_B^2 \leftarrow \frac{1}{n}\sum_{j=1}^{n} (x_j - \mu_B)^2$, respectively. The jth dimension is then normalized by the following expression,

$$\hat{x}_j \leftarrow \frac{x_j - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{5.30}$$

where ϵ is a constant, introduced for arithmetical stability. The normalized values $\hat{x}$ are then scaled and shifted for enhanced representation by the following expression:

$$Y_j \leftarrow \zeta \hat{x}_j + \beta \equiv BN_{\zeta,\beta}(x_j), \tag{5.31}$$

where ζ and β are learnable parameters. The result of the second transformation Y is propagated to other layers of the network.

5.5.3.1 Application to other DCNN models and empirical summary. Batch normalization allows for higher learning rates, which traditionally resulted in exploding or vanishing gradient issues (Bengio et al., 1994), and this significantly accelerates training times. Furthermore, it has a regularization effect similar to Dropout (Hinton et al., 2012; Srivastava et al., 2014), and when combined with the Inception model (Szegedy, Liu, et al., 2015), there were significant training speed gains without an increase in overfittting. When combined into an ensemble, batch-normalized DCNNs achieved the best reported results on the ImageNet data set (Russakovsky et al., 2015), passing the previous best by He et al. (2015a), which was already considered better than human-level performance; however, this was later superseded by the improved Inception model (Szegedy, Vanhoucke et al., 2015), the residual networks of He et al. (2015b), and the batch normalized Inception-residual networks of Szegedy et al. (2016).

In fact, this is clearly illustrated in Table 4, which compares the performance of batch normalization to selected representative DCNNs on the ImageNet data set (Russakovsky et al., 2015), starting from the revolutionary work of Krizhevsky et al. (2012) up to the current state of the art (Szegedy

Table 4: DCNN Performance on the ImageNet Data Set.

Code Name and Reference	Specific Contributions	Top 5 Error (%)
AlexNet (Krizhevsky et al., 2012)	DCNN model across parallel GPUS, innovations including Dropout, data augmentation, ReLUs, and local response normalization	15.3
Z&F-net (Zeiler & Fergus, 2014)	Novel visualization method; larger convolutional layers compared to Krizhevsky et al., 2012	*11.7*
SPP-net (He et al., 2014)	Spatial pyramid pooling to allow for flexible image size	8.06
VGG-net (Simonyan & Zisserman, 2014)	Increased depth, more convolutional layers, 3×3 convolutional filter	*7.32*
GoogLeNet (Szegedy, Liu, et al., 2015)	Novel inception architecture, very large network, dimensionality reduction	*6.67*
PReLU-net (He et al., 2015a)	PReLU activation functions and a robust initialization scheme	4.94
BN-Inception (Ioffe & Szegedy, 2015)	Batch normalization combined with inception architecture	4.82
Inception-V3 (Szegedy, Vanhoucke et al., 2015)	Factorized convolutions, aggressive dimensionality reduction	3.58
ResNets (He et al., 2015b)	Residual functions/blocks integrated into DCNN layers	*3.57*
BN-Inception-ResNet (Szegedy et al., 2016)	**BN, Inception architecture integrated with residual functions**	**3.08**

Notes: The entry in bold represents the state of the art or lowest known published error rates for the data set. The italicized entries represent results obtained in competition.

et al., 2016). For the ILSVRC, there are three reporting measures, the top 1 and top 5 error rates, and hierarchical criteria; however, since the top 5 metric is the simplest and is best suited to the benchmark, it has been used exclusively since 2012. The italicized numbers represent results obtained in competition, while Russakovsky et al. (2015) discuss further results, especially from early successes, together with a detailed analysis on this challenging classification benchmark in their comprehensive survey.

Furthermore, BN was also combined with the NIN model (Lin et al., 2013) and Maxout activations (Goodfellow et al., 2013) to form a complex batch-normalized Maxout network in network (MIN) module and network, which is illustrated by the top and bottom halves of Figure 12, respectively (Chang & Chen, 2015). Despite its sophistication, it outperformed the high-performing RCNNs and achieved state-of-the-art results (for no data augmentation), on the MNIST and CIFAR-100 benchmarks and a new state of the art (for with and without data augmentation) on the CIFAR-10 data set (LeCun et al., 1998; Krizhevsky, 2009; Liang & Hu, 2015). The NIN model

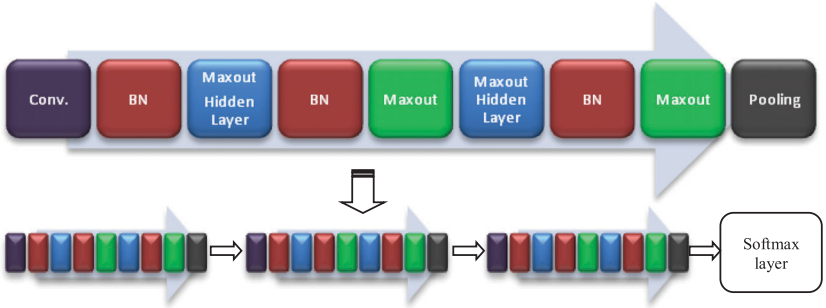

Figure 12: Batch-normalized MIN (Chang & Chen, 2015).

(Lin et al., 2013) was also extended to incorporate batch-normalized (Ioffe & Szegedy, 2015) activations prior to Maxout activations (Goodfellow et al., 2013), resulting in the Maxout network in Maxout network (MIM) model (Liao & Carneiro, 2016), which is similar in architecture and complexity to the MIN model presented by Chang and Chen (2015). The paper showed empirical proof that using BN led to model preconditioning, allowing for the use of larger learning rates, and thus faster convergence, while still maintaining accuracy.

Table 5 compares the performances of several state-of-the-art or high-performing DCNNs on the MNIST data set (LeCun et al., 1998). The results obtained in the clear blocks were by single models that did not use data augmentation, while the results in red were the state-of-the-art performance for this configuration at the time of their respective papers. As illustrated, for this configuration, MIN obtained state-of-the-art results, which were very close to the lowest error rates achieved by ensemble-based models supplemented with data augmentation (shaded blocks). Despite several DCNN successes since the DropConnect paper (Wan et al., 2013), this model still holds the record for the lowest known published error rate on this popular benchmark.

5.5.3.2 Batch renormalization. Despite the effectiveness of BN in accelerating the training of DCNNs, its efficiency is challenged for small mini-batches or mini-batches that do not contain independent samples. Asserting that these shortcomings are due to the activations being computed differently during training and inference, Ioffe (2017) replaced BN with batch renormalization, which ensures that the outputs computed by a model are reliant not on the entire mini-batch but on the individual examples throughout the course of training and inference. Specifically, for models that have batch-normalized layers, Ioffe (2017) augments the model by applying per dimension affine transformations, of which they keep the parameters fixed,

Table 5: DCNN Performance on the MNIST Data Set.

Model and Reference	Brief Description	Error (%)
Pretrained DCNN (Ranzato et al., 2006)	Energy-based unsupervised pretraining, followed by DCNN	*0.60*
DCNN-NN (Jarrett et al., 2009)	Dual CNN feature extractor followed by dual neural network	*0.53*
FitNets (Romero et al., 2015)	Thin, deep networks with intermediate-level hints to guide training	0.51
Stochastic pooling (Zeiler & Fergus, 2013)	Stochastic rather than deterministic pooling procedure	*0.47*
NIN (Lin et al., 2013)	MLP integrated into DCNN architectur	0.47
Maxout networks (Goodfellow et al., 2013)	Maxout activation functions	*0.45*
Highway Networks (Srivastava et al., 2015b)	Learning gate mechanism for regulating DCNN information flow	0.45
Deeply supervised nets (Lee et al., 2015)	Companion objective function, feature quality feedback	0.39
MIM (Liao & Carneiro, 2016)	Maxout network in Maxout network	*0.35*
RCNN (Liang & Hu, 2015)	Recurrent connections in convolutional layer	*0.31*
Tree+Max-Avg pooling (Lee et al., 2016)	Tree pooling followed by gated average max pooling	*0.31*
Batch-normalized MIN (Chang & Chen, 2015)	BN, Maxout activations, NIN architecture	*0.24*
Multicolumn DCNN (Ciresan et al., 2012)[a]	Multicolumn DCNNs, with data augmentation (elastic distortions)	*0.23*
DropConnect (Wan et al., 2013)[a]	**Ensemble of DropConnect networks, with data augmentation (no elastic distortions)**	*0.21*

Notes: The italicized entries represent the state of the art for this configuration at the time of their publication data set.
[a]Ensemble-based models that used data augementation.

to the already normalized network activations, thereby allowing the following layers to observe the precise activations that would be generated by the inference model. Thus, this extension of BN enforces a per dimension correction to ensure correlation between the activations of the training and inference models, and this can completely eliminate overfitting for image data sets that have a biased label distribution.

5.5.3.3 Analysis and outlook. Internal covariate shift, which imposes lower learning rates and cautious parameter initializations, is a major

problem in DNNs, and DCNNs are not immune to it. BN was conceptualized to address this. It can significantly reduce training times by reducing the total number of iterations required for convergence; during inference, since the means and variances can be doubled into the convolutional layer, the effect of the overhead generated is eliminated. Various researchers have confirmed its ability and incorporated it into their models, as illustrated in section 5.5.3.1. However, notwithstanding this, it adds 30% computational overhead and requires an additional 25% of parameters per iteration (Ioffe & Szegedy, 2015). This has stimulated others to improve this aspect of the technique. The LSUV initialization scheme of Mishkin and Matas (2016) shares the same unit variance normalization procedure as BN and can be regarded as a weight initialization scheme supplemented by BN applied to the first mini-batch. The method efficiently decorrelates layer activations by performing orthonormalization of weights matrices, and this makes it more computationally efficient per iteration when compared to BN. However, for large data sets like ImageNet (Russakovsky et al., 2015), its results are erratic, and although BN is still the preferred choice, other attempts to reduce the computational encumber and additional parameter requirements per step imposed on systems that use BN are still necessary.

Furthermore the efficacy of BN diminishes for mini-batches constrained by size and devoid of independent samples, and although this can be mitigated by utilizing batch renormalization (Ioffe, 2017), which is easy to implement and significantly improves the training of constrained mini-batches, the method introduces extra hyperparameters that require tuning and is still new, requiring further investigation. While future work should also attempt to reduce the intricacy of the complex models (Chang & Chen, 2015; Liao & Carneiro, 2016) that use BN, this should be supplemented by a theoretical justification of why BN leads to faster convergence. Other attempts to solve the problem of internal covariate shift such as novel training procedures are also plausible. One such fresh attempt is Evolutional Dropout proposed by Li et al. (2016; see section 5.4.1.4), which adapts dropout sampling probabilities, computed on-the-fly from a mini-batch of examples, to the evolving distributions of layers outputs rather than using identical and independent probabilities as in standard Dropout (Hinton et al., 2012).

5.5.4 Skip Connections. Even though increasing network depth generally leads to increased performance (see Table 2), improving the classification accuracy of DCNNs is not as straightforward as simply adding layers (Srivastava et al., 2015a). As mentioned in section 4.4, some of the complications include overfitting, an increased computational burden and memory footprint, and degradation (Krizhevsky et al., 2012; Szegedy, Liu, et al., 2015; He et al., 2015b; He & Sun, 2015; Srivastava et al., 2015a; Romero et al., 2015). In particular, degradation remains a key challenge. Thus, overcoming it is imperative in order to investigate the benefits of very deep networks for

several applications; this has led to work that focuses on skip (or shortcut) connections within DCNNs.

5.5.4.1 Highway networks. Highway networks (Srivastava et al., 2015a, 2015b) use a learnable gating mechanism, inspired by the long short-Term memory (LSTM) recurrent neural networks (RNNs) presented by Hochreiter and Schmidhuber (1997), for regulating information flow across several layers without degradation. The output y of a highway network block is computed by

$$y = H(\mathrm{x}, \mathrm{W_H}) \cdot T(\mathrm{x}, \mathrm{W_T}) + \mathrm{x} \cdot C(\mathrm{x}, \mathrm{W_C}), \tag{5.32}$$

where H (parameterized by $\mathrm{W_H}$) is a nonlinear transformation on its associated input x, and T and C represent transform gates and carry gates, respectively. In the equation, the layer indices and biases have been excluded for simplification. These networks can have hundreds of layers, since their architecture enables optimization regardless of depth. On MNIST (LeCun et al., 1998) and CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), the authors obtained competitive results in comparison to the best-performing DCNNs, even though their networks had many fewer parameters compared to the thin, wide, and shallower (yet still deep) DCNN model compression networks of Romero et al. (2015). Following this, a similar grid-based LSTM model was proposed for a wider range of tasks (Kalchbrenner, Danihelka, & Graves, 2015), which included language translation, character predication, sequence memorization, and, most meaningful, image classification.

5.5.4.2 Residual networks. Residual networks (He et al., 2015b), introduced in section 4.4, also address the degradation problem, using skip connections. The main idea of residual networks is to learn an additive residual function with respect to an identity mapping that is based on the preceding layers inputs, accomplished by attaching an identity shortcut connection. Residual modules perform the following computation:

$$y_l = h(\mathrm{x}_l) + \mathcal{F}\big(\mathrm{x}_l, W_{l,k|1\le k\le K}\big), \tag{5.33}$$

$$\mathrm{x}_{l+1} = f(y_l), \tag{5.34}$$

where the input to the lth residual module is denoted by x_l, W_l represents its weights and biases, K is the number of layers in a module, $\mathcal{F}$ represents the residual function such as a stack of convolutional filters, f is the operation that follows element-wise addition (see Figure 6), and h is an identity mapping of the form $h(\mathrm{x}_l) = \mathrm{x}_l$. Residual networks have brought about some empirical success; in particular, they have performed exceptionally well on the demanding ImageNet challenge (Russakovsky et al., 2015), as illustrated by Tables 2 and 4.

5.5.4.3 Improved residual networks. To further enhance the residual learning framework, He, Zhang, Ren, and Sun (2016) found that it was possible to create a direct path for propagating information throughout the network in both the forward and backward pass rather than within residual units alone (He et al., 2015b). This propagation made training easier and was accomplished by using identity mappings as the shortcut connections coupled with after-addition activation. Considering equations 5.33 and 5.34, if f is also an identity mapping of the form $\mathrm{x}_{l+1} \equiv y_l$, by substituting equations 5.33 into 5.34, the input feature to the lth residual module can be computed by

$$\mathrm{x}_L = \mathrm{x}_l + \mathcal{F}(\mathrm{x}_l, W_l). \tag{5.35}$$

If equation 5.35 is solved recursively, this translates to

$$\mathrm{x}_L = \mathrm{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathrm{x}_i, W_i), \tag{5.36}$$

where L represents residual modules that are deeper than preceding shallower modules represented by l. Significantly, this result presents two key attributes. First, the features of deeper residual modules, denoted by x_L, can be represented as the features of any shallower module, denoted by x_l, supplemented by a residual function ($\sum_{i=l}^{L-1} \mathcal{F}$). Second, the features of deeper residual modules comprise the summation of all the previous residual functions. They empirically established their improved technique by easily training a network that had 1000 layers and demonstrated improved accuracy. Furthermore, they experimented with several other optimization techniques from Hinton et al. (2012), Srivastava et al. (2014), Szegedy, Liu, et al. (2015), and Srivastava et al. (2015a) and found that these had a negative effect on information propagation and hindered optimization. Notwithstanding this, they found that BN (Ioffe & Szegedy, 2015) and ReLU (Nair & Hinton, 2010) preactivations improved the performance of their previous generation residual networks (He et al., 2015b). Moreover, shortcut residual connections were also combined with the Inception architecture (Szegedy, Liu, et al., 2015; see section 5.1.1.2) and BN (Ioffe & Szegedy, 2015) for improved image classification performance (Szegedy et al., 2016).

Given that the extremely deep residual networks of He et al. (2016) were slow to train, their depth was reduced and width increased in a new variant called wide residual networks (WRNs; Zagoruyko & Komodakis, 2017). These WRNs were much shallower since they consisted of 16 layers compared to the 1000 of He et al. (2016), yet they outperformed all the previous residual models in terms of efficiency and accuracy and set new state-of-the-art results on the CIFAR-100 (Krizhevsky, 2009; Szegedy, Vanhoucke et al., 2015), and SVHN (Netzer et al., 2011; Lee et al., 2016) data sets.

However, these were later superseded by the shortcut technique introduced in section 5.5.4.4.

5.5.4.4 Densely connected convolutional networks. The recently proposed densely connected convolutional networks (Huang, Liu, Weinberger, & van der Maaten, 2016) extend the idea of skip connections by connecting in the usual feedforward modus each layer with every other layer in the network. Thus, the feature maps of all the previous layers are used as the inputs for each succeeding layer. Formerly, using notation similar to equation 5.35, the lth layer accepts the preceding layer's feature maps, $x_0, \ldots x_{l-1}$, as its input:

$$x_0 = H_l([x_0, x_1, x_2 \ldots x_{l-1}]), \tag{5.37}$$

where $[x_0, x_1, x_2 \ldots x_{l-1}]$ represents the feature map concatenation of the maps generated in layers $0, \ldots l-1$, and H_l, which follows the improved residual networks (He et al., 2016), is a compound function of ReLUs (Nair & Hinton, 2010), preceded by BN (Ioffe & Szegedy, 2015), and is followed by convolution. On popular image classification benchmarks, including the challenging ImageNet (Russakovsky et al., 2015), they obtained accuracy comparable to residual networks (He et al., 2016) but required significantly fewer parameters. Furthermore, they tackle degradation (He et al., 2015b; He & Sun, 2015; Srivastava et al., 2015a, 2015b) and the vanishing gradient problem (Bengio et al., 1994), while also promoting feature reuse to reduce computation.

Table 6 compares the performance of several DCNNs on the popular CIFAR (Krizhevsky, 2009) and SVHN (Netzer et al., 2011) data sets. The results reported in the CIF-10 (with DA) column are by DCNN models that used data augmentation. For CIFAR-100 and SVHN, where data augmentation is less popular, the results of the models that used data augmentation are in italics. As illustrated, WRNs (Zagoruyko & Komodakis, 2017) surpassed several other state-of-the-art DCNNs on these tasks (CIFAR-100 and SVHN), and when DCNNs were combined with ELUs (Clevert et al., 2016), they obtained the lowest classification error at the time on the CIFAR-10 data set without data augmentation. On CIFAR-10, with data augmentation, WRNs obtained results very close to the second lowest error rate obtained by the fractional max pooling method presented by Graham (2014; see section 5.1.2.2), even though they did not use the same extreme data augmentation techniques. The densely connected convolutional networks of Huang, Liu et al. (2016) superseded all of these results, illustrating the exceptional performance of the technique and the importance and empirical successes of shortcut connections within our current models.

5.5.4.5 Analysis and outlook. As researchers began experimenting with deeper networks for improved image classification performance, they

Table 6: DCNN Performance on CIFAR-10 and CIFAR-100, and SVHN Data Sets.

Model	Brief Description	CIF.-10	CIF.-10 (with DA)	CIF.-100	SV-HN
DropConnect (Wan et al., 2013)	Optimized DCNNs with dropped connections	18.7	9.32	-	*1.94*
Stochastic pooling (Zeiler & Fergus, 2013)	Stochastic rather than deterministic pooling procedure	15.13	-	42.51	2.80
Maxout networks (Goodfellow et al., 2013)	Maxout activation functions	11.68	9.38	38.57	2.47
Probout (Springenberg & Riedmiller, 2013)	Probabilistic activation functions	11.35	9.39	38.14	2.39
NIN (Lin et al., 2013)	MLP integrated into DCNN architecture	10.41	8.81	35.68	2.35
Deeply supervised nets (Lee et al., 2015)	Companion objective function, feature quality feedback	9.78	8.22	34.57	1.92
DCNN + APL (Agostinelli et al., 2014)	Adaptive piecewise linear activations	9.59	7.51	34.40	-
All-CNN (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2014)	DCNN with convolutional layers only	9.08	4.41	33.71	-
RCNN (Liang & Hu, 2015)	Recurrent connections in convolutional layers	8.69	7.09	31.75	1.77
Doubly convolutional nets	Double convolutional operation	8.58	7.24	30.35	-
MIM (Liao & Carneiro, 2016)	Maxout network in Maxout network	8.52		29.20	1.97
Batch-normalized MIN (Chang & Chen, 2015)	BN, Maxout activations, NIN	7.85	6.75	28.86	1.97
Tree+Max-Avg pooling (Lee et al., 2016)	Tree pooling followed by gated average max pooling	7.62	6.05	32.37	1.69
ELU (Clevert et al., 2016)	Exponential linear activation functions	6.55	-	24.28	-
Tree based priors (Srivastava & Salakhutdinov, 2013)	DCNN with learned tree priors	-		36.85	-
FitNet-LSUV (Mishkin & Matas, 2016)	FitNet architecture with LSUV initialization		6.06	*27.66*	

Table 6: Continued.

Model	Brief Description	CIF.-10	CIF.-10 (with DA)	CIF.-100	SV-HN
Deep Attention Selective Net (Stollenga, Masci, Gomez, & Schmidhuber, 2014)	DCNNs with feedback connections	-	9.22	*33.78*	-
FitNets (Romero et al., 2015)	Thin, deep networks with intermediate level hints for training	-	8.39	*34.04*	2.42
Highway networks (Srivastava et al., 2015b)	Learning gates for regulating data flow	-	7.6	*32.33*	-
Deep residual networks 1 (He et al., 2015b)	Residual functions / blocks integrated into DCNN layers	-	6.43	-	-
Deep residual networks 2 (He et al., 2016)	Residual blocks with identity mappings	-	4.62	*22.71*	-
Fractional max pooling (Graham, 2014)	Fractional stochastic version of max pooling	-	3.47	26.39	-
WRNs (Zagoruyko & Komodakis, 2017)	Residual blocks with increased width	-	4.17	*20.50*	1.64
Densely connected CNNs (Huang, Liu, et al., 2016)	**Interconnections between layers**	**5.19**	**3.46**	*17.18*	**1.59**

Note: Entries in italics for CIFAR-100 and SVHN are the results of the models that used data augmentation.

encountered what can still be regarded as an open challenge, commonly referred to degradation (see section 4.4). An early attempt to counter degradation was carried out by highway networks (Srivastava et al., 2015a, 2015b), which used gating shortcuts to effectively train very deep networks (more than 100 layers), optimized by SGD (Bottou, 1998, 2010). However, there are instances when the shortcuts are closed, preventing information flow; furthermore, the gated shortcuts are data dependent and require a number of parameters. Residual networks (He et al., 2015b, 2016) also alleviate the problem of degradation; however, in contrast to highway networks, they use identity-mapping shortcuts that are parameter free and are always open, allowing for continuous information flow. Residual networks, which can contain greater than 1000 layers without diminishing performance, have produced exceptional performance on localization, detection, and classification tasks. Despite their successes, others have found that there is significant redundancy in their extreme number of layers,

and this causes them to take a superfluous long time to train, advocating the need for other alternatives. Along these lines, WRNs (Zagoruyko & Komodakis, 2017) demonstrated that shallower residual networks can produce classification performance comparable to their extremely deep counterparts (He et al., 2015b, 2016) and are several times faster to train; however, their performance on the challenging ImageNet (Russakovsky et al., 2015) was made available only this year, encouraging the need for further experimentation with shallower yet wider convolutional networks.

Stochastic depth networks (Huang, Sun, Liu, Sedra, & Weinberger, 2016) also challenged the layer redundancies of residual networks by bypassing redundant layers with identity functions after stochastically dropping them. These networks accomplished reduced classification error and were faster compared to residual networks on the CIFAR-10 benchmark (Krizhevsky, 2009); however, they are yet to be tested on the challenging ImageNet, on which residual networks have significantly advanced the state of the art. Furthermore, they require hyperparameter tuning. Densely connected networks, which impose lower memory and computational constraints compared to residual networks, take shortcut connections to the extreme by introducing direct connections between layers, but unlike residual networks, they are yet to be tested on diverse computer vision tasks apart from image classification. Thus, although there has been progress toward solving degradation, the techniques available currently are yet to be firmly established. The empirical successes of residual networks on ImageNet make them the most appealing choice; however, when combined with Dropout (Hinton et al., 2012; Srivastava et al., 2014), they fail to converge to acceptable solutions (He et al., 2016). Thus, future work focusing on fundamentally changing the residual learning framework to work in conjunction with Dropout and other regularization techniques is required. To this end, WRNs show promising initial results.

Finally, despite the successful applications of the various shortcut connections discussed here and their promising empirical results reported thus far, a clear understanding of how they fundamentally improve the training of DCNNs is still devoid. Although some recent work has attempted to address diverse characteristics of this challenge (Hardt & Ma, 2016; Li, Jiao, Han, & Weissman, 2016; Littwin & Wolf, 2016), perhaps the most stimulating explanation behind their success is that they break intrinsic symmetries in the loss landscapes of DCNNs, resulting in considerably simplified landscapes, as observed by Orhan (2017). Specifically, for densely connected convolutional networks (see section 5.5.4.4), the paper found that shortcut connections cause discontinuities in the rescaling symmetry of the matrices that connect the different layers of the network, while for other deep models, they cause discontinuities in the permutation symmetries of the neurons at a specific layer. Furthermore, although other means of breaking symmetries also facilitate performance improvements when training deep models, they found that shortcut connections promote additional benefits further to

their symmetry-breaking physiognomies, and these include their ability to deal with the vanishing or exploding gradient problem (Bengio et al., 1994). Moreover, with regard to symmetry breaking, they also observed that differentiated classes of neurons, perhaps like those used by the human brain (Harris & Shepherd, 2015), are superior to undifferentiated neurons commonly used by artificial networks. Although this work provides a starting point, further research is required to better understand the role of shortcut connections in our models and to determine if the central nervous system uses mechanisms similar to them for visual tasks.

5.5.5 Computational Cost Developments. It is well known that larger data sets have contributed to the successes of deep learning (Krizhevsky et al., 2012; Deng & Yu 2014; Zeiler & Fergus, 2014). However, the downside, particularly during training, is a greater computational burden. Coupled to this is the fact that DCNN models have a tremendous number of parameters, which has a negative effect on their storage and memory requirements (Krizhevsky et al., 2012; Wan et al., 2013; Simonyan & Zisserman, 2014; Szegedy, Vanhoucke et al., 2015; Szegedy, Liu, et al., 2015; He et al., 2015b; Szegedy et al., 2016). As an example, the DCNN from Krizhevsky et al. (2012) had 60 million parameters and took six days to train on two GPUs, while the largest model presented by Simonyan and Zisserman (2014) consisted of 144 million parameters, trained on four GPUs in two to three weeks. Thus, a considerable amount of research has gone into reducing computational costs and storage space requirements of DCNNs. We next discuss some of the representative work in this regard.

5.5.5.1 Parallel computing. A considerable amount of effort (Zinkevich et al., 2010; Recht et al., 2011; Dean et al., 2012; Zhuang et al., 2013; Paine et al., 2013; Yadan, Adams, Taigman, & Ranzato, 2014; Krizhevsky, 2014) has gone in to parallelizing the training of DCNNs via model parallelism, which entails the use of GPUs, multiple GPUs, GPU and CPU clusters, and data parallelism, which incorporates improved optimization algorithms such as asynchronous SGD (ASGD; Recht et al., 2011; Dean et al., 2012) and BN (Ioffe & Szegedy, 2015). Yadan et al. (2014) utilized a hybrid parallelism strategy that considered both model and data parallelism. They used a configuration that shared the network's computation and image mini-batch split over four GPUs to reduce training time by greater than 2.2 times when compared to a single GPU. Dean et al. (2012) introduced a new framework for the parallel-distributed training of deep networks on a cluster of CPUs. Within the framework, they introduced a new form of ASGD, called Downpour SGD, to support training a large number of model replicas. This framework was also utilized by the top-performing system of Szegedy, Liu, et al. (2015), and although they used a CPU-based implementation, they forecast network convergence on several high-end GPUs as well, albeit with the estimated downside being higher memory use.

Paine et al. (2013) also exploited model and data parallelism by using GPUs for model parallelism and A-SGD (Recht et al., 2011; Paine et al., 2013) for data parallelism and accomplished 3.2 times speed gains with eight GPUs in comparison to a single unit; however, they lost significant accuracy. Krizhevsky (2014) proposed two variants of SGD descent that parallelize the training of DCNNs by utilizing heavy data parallelism in the convolutional layers and more model parallelism in the fully connected layers. On eight GPUs, the paper achieved 6.16 times speed gains with a negligible change in accuracy. Asserting that this scheme was too sophisticated, Simonyan and Zisserman (2014) achieved speed gains of 3.75 times on an off-the-shelf GPU system, consisting of four GPUs relative to a single GPU. However, even with this, it took two to three weeks to train a single network, thus advocating the need for other solutions. More recently, Dettmers (2016) proposed a large-scale GPU cluster, combined with improved parallelism algorithms for efficient DCNN parallelization (see section 6.3 for further details). Even though this system produced successful classification results, it lacks practicality for large-scale deployment.

5.5.5.2 Exploiting the convolution theorem and circular projections. By carrying out the convolutional operation as element-wise products in the Fourier domain, using fast Fourier transforms (FFTs) and recycling the same transformed feature map numerous times, Mathieu et al. (2013) achieved significant processing speed gains (up to two times) at the cost of a larger memory footprint. This technique can easily be integrated with spectral pooling (see section 5.1.2.5), and since the discrete Fourier transform (DFT) is performed for both methods, there are negligible additional computational costs (Rippel et al., 2015).

Independently, Cheng, Felix et al. (2015) and Cheng, Yu et al. (2015) also used FFTs to speed up computation. However, in contrast to Mathieu et al. (2013), who used the well-established convolution theorem to minimize the processing time in the convolutional layers, they focused on speeding up computation in the fully connected layers by imposing a circular rather than a linear structure on weight matrices. More formally, given a fully connected layer with d input and d output nodes, circular projections improve the time complexity from $O(d^2)$ to $O(d \log d)$. Furthermore, their method also benefits storage space and reduces the space complexity from $O(d^2)$ to $O(d)$. On MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky, 2009), and ImageNet (Russakovsky et al., 2015), they achieved significant computational cost and storage space reductions, with minimal increases in classification error. Following with the use of digital signal processing entrenched techniques, Wang, Xu, You, Tao, and Xu (2016) proposed compressing and reducing the computational costs of DCNNs by linearly uniting the convolution responses of discrete cosine transform (DCT) bases after first

treating the convolutional filters as images and then decomposing their representations in the frequency domain.

5.5.5.3 Matrix manipulations. Denil, Shakibi, Dinh, and de Freitas (2013) carried out an early attempt to alleviate the overparameterization of deep networks (Hinton et al., 2012; Denton, Zaremba, Bruna, LeCun, & Fergus, 2014; Kim et al., 2015). Their technique, aimed at reducing neural network's free parameters (weights and biases), was based on representing the weight matrix as a low-rank product of two smaller matrices. Therefore, by factoring and controlling the rank of the weight matrix, they were able to directly control the network's parameterization. On the CIFAR-10 benchmark (Krizhevsky, 2009), they found that by predicting and thus eliminating the computation and storage space of 75% of the parameters, there was a negligible effect on classification accuracy. Their method is complementary to other DCNN advances such as Dropout (Hinton et al., 2012; Srivastava et al., 2014) and Maxout activations (Goodfellow et al., 2013).

Spurred on by this, Denton et al. (2014) exploited the redundancy within the convolutional layers and derived approximations to minimize computation. Specifically, they used monochromatic approximations in the first convolutional layer and biclustering approximations with SVD in the second convolutional layer. In both layers, they report speed gains between 2 and 2.5 times, with a 1% drop in performance, relative to their baseline models. Furthermore, by applying truncated SVD, they reduced the memory overhead and storage requirements of the fully connected layers and reported up to 13.4 times reduction in weights with less than 1% loss in accuracy. Similar to this, Jaderberg, Vedaldi, and Zisserman (2014) used filter approximations to approximate the convolutional filters and followed this by using filter and data reconstruction techniques to reconstruct the approximations with minimal error. In a scene text classification application, they attained speed gains of 2.5 times with no loss in accuracy and 4.5 times with less than 1% drop in accuracy. Both Denton et al. (2014) and Jaderberg et al. (2014) employ low-rank matrix factorization to compress either a single or multiple layers; others who have employed related techniques include Sainath, Kingsbury, Sindhwani, Arisoy, and Ramabhadran (2013) and Lebedev, Ganin, Rakhuba, Oseledets, and Lempitsky (2014).

Inspired by the redundancies in neural networks parameters highlighted by Denil et al. (2013), Gong, Liu, Yang, and Bourdev (2014) proposed vector quantization as a higher-performing alternative to low-rank matrix factorization (Sainath, Kingsburg, Sindhwani et al., 2013; Denton et al., 2014; Jaderberg et al., 2014), for the compression of the matrices of the dense fully connected layers. They were able to obtain impressive compression rates (up to 24 times) without sacrificing more than 1% top 5 classification accuracy on the challenging ImageNet data set (Russakovsky et al., 2015). In

addition to Gong, Liu et al. (2014), various other methods based on vector quantization have been shown to achieve better compression than SVD. These include circular projections (Cheng, Felix et al., 2015; Cheng, Yu et al., 2015), hashing techniques (Chen, Wilson, Tyree, Weinberger, & Chen, 2015), and tensor train decomposition (Oseledets, 2011; Novikov, Podoprikhin, Osokin, & Vetrov, 2015).

5.5.5.4 Analysis and outlook. Our current state-of-the-art classification models are deeply dependent on supervised training; however, this regiment, has an inherent limitation in that it requires an exhaustive amount of training data that significantly lengthens the training process and inflicts space and memory complications. Although this can be mitigated by the brute force approach of using CPU or GPU clusters, access to such systems is restricted to large organizations. To reduce computation and memory footprints, different aspects of DCNN computation can be carried out using techniques firmly established in digital signal processing, such FFTs, DFTs, DCTs, and the convolutional theorem. For these techniques, the computation and memory footprints of either the convolutional operation or the fully connected layers can be reduced. Although moving between domains increases the complexity of our systems, given the successes described in section 5.5.4.2, further research into hybrid DCNNs that make use of digital signal processing–based fundamentals to improve not only computational costs and memory footprints but also classification accuracy (see section 5.1.2.5) is vindicated. Manipulation of the weight matrices of both the convolutional and fully connected layers is another alternative to improve computation efficiency and tackle the consequences of DCNN over parameterization; however, although further improvements forged from applied mathematics are encouraged, they do not tackle the root cause of the problem. In summary, moving operations to the frequency domain and manipulating matrices can lead to improved computational characteristics; however, all the techniques discussed in this section suffer from a loss in accuracy, even if it is just marginal. Parallel computing approaches can provide the required accuracy; however, this comes with financial implications preventing large-scale use and is not practical for large-scale adaptation. Thus, further research to address these challenges is required. While we have reviewed some of the early improvements in this section, the latest trends to improve DCNN computation, supplemented by future recommendations, are elaborated on in section 6.3.

6 Selected Open Challenges and Trends

Despite the promising image classification results obtained by DCNNs, there are still challenges that need to be addressed. In this final section, we address some of these, together with selected trends in recent work.

6.1 Theoretical Justification and Internal Understanding. Despite the empirical successes of DCNNs, the theoretical proof of why they succeed is lacking. To this end, Mallat (2012) proved translation invariance and deformation stability of the features extracted by scattering convolutional networks. Wiatowski and Bölcskei (2015) persisted with the mathematical analysis of the features extracted by DCNNs, and they theoretically established deformation stability and vertical translation invariance. Continuing with the theoretical analysis, Basu et al. (2016) considered image classification data sets where texture plays a significant role (Lazebnik, Schmid, & Ponce, 2005; Filho, Luiz, Oliveira, & Britto, 2009; Oxholm, Bariya, & Nishino, 2012), and they provided theoretical bounds on the use of DCNNs for textural classification. Specifically, they used the theory of Vapnik-Chervonenkis (VC) dimension (see Vapnik & Chervonenkis, 1971, for details and Sontag, 1998, for the first application to ANNs) to demonstrate that hand-crafted feature extraction creates low-dimensional representations. As a corollary to this, they derived the upper bounds on the VC dimension of DCNNs. Furthermore, there has also been investigation into the internal operation and performance of DCNNs, such as the commonly cited feature visualization technique presented by Zeiler and Fergus (2014). Other visualization techniques, all with the intention of understanding the internal mechanisms of DCNNs, have also been proposed (Girshick, Donahue, Darrell, & Malik, 2014; Yu, Yang, Bai, Yao, & Rui, 2014a, 2014b). Further progress is dependent on both sound theoretical proof and practical investigations that lead to improved understanding and performance.

6.2 Geometric Invariance. Although DCNNs are robust against small-scale deformations (Lee et al., 2009), their final representations are not geometrically invariant (Ciresan et al., 2011; Gong, Wang, et al., 2014; Razavian et al., 2014). Specifically, they are sensitive to global translations, rotations, and scaling (Gong, Wang, et al., 2014). To address translation variances, Lee et al. (2009) proposed probabilistic max pooling, while the MOP scheme of Gong, Wang, et al. (2014) was shown to be robust against several geometric variances. Recently, the TI pooling scheme presented by Laptev et al. (2016) efficiently handled rotations and scale changes and thus built transformation invariance into DCNN architecture, while the spatial transformer module proposed by Jaderberg et al. (2015) learned translation, scale, rotation, and warping invariance. Thus, an interesting direction is to investigate if further fundamental changes to DCNN architectures are required to improve their universal robustness. Furthermore, despite the large number of images available in modern data sets like ImageNet (Russakovsky et al., 2015), it is still conceivable that our current data sets are not suitable for the invariance tasks we now face. Thus, another promising direction is to collect or generate new training data that do not necessarily lead to larger data sets, similar to the trend we have seen over the last few years, but will facilitate the learning of DCNN features that contribute to more robust models.

6.3 Toward Mobile Deployment. Despite several computational cost, memory footprint, and storage reductions (see section 5.5.4), improving DCNNs in this regard continues to be an open area of research, especially with the aim of deploying them on FPGAs, embedded systems, mobile devices, and other devices with limited memory and battery constraints. Some of the very recent developments in this regard include DCNN compression and weight quantization, fast algorithms, GPU clusters with truncated representations, and FPGA accelerated advances.

By using network pruning, weight quantization to enforce weight sharing, and Huffman encoding for greater compression, Han, Mao, and Dally (2016), introduced deep compression, which reduced the storage requirements of the model proposed by Krizhevsky et al. (2012) from 240 MB to 6.9 MB without any loss in accuracy. When benchmarked on CPUs, GPUs, and mobile GPUs, they also obtained speed gains of between 3.0 and 4.2 times. A much more effective compression method, which introduced a new DCNN Fire module (1×1 squeeze convolutional layer, followed by 1×1 and 3×3 convolutions) and SqueezeNet architecture, also experimented with the model from Krizhevsky et al. (2012) reducing it to 4.8 MB without any loss in accuracy (Iandola et al., 2016). Furthermore, when they applied deep compression to their model, with 6 bit weight quantization, they managed a further reduction to 0.47 MB, also without any impact on the classification error.

Another extreme quantization approach, which presented binary weight (filters approximated with binary weights) and XNOR networks (binary weights and binary inputs to convolutional layers), resulted in networks with 32 times memory savings compared to baseline (Krizhevsky et al., 2012) and, in the case of the XNOR-network, 58 times speed gains during inference (Rastegari, Ordonez, Redmon, & Farhadi, 2016). Courbariaux, Hubara, Soudry, El-Yaniv, and Bengio (2016) introduced binarized networks that have binary weights and activations at run time and when computing the parameters' gradients during training. Similar to binary and XNOR networks, these networks also reduce memory footprint by a factor of up to 32 times, while also reducing memory access by the same amount. Courbariaux, Bengio, and David (2015), Cheng, Soudry, Mao, and Lan (2015) and Kim and Smaragdis (2016) carried out other recent quantization attempts along these lines that also introduced binarized weights and activations during training and inference. DCNNs with weight quantization are expected to substantially advance power efficiency since they perform bit-wise operations rather than the usual arithmetic ones and their fast computation and significantly reduced memory footprints make them well suited for mobile deployment. What remains to be investigated is whether they deteriorate or improve model performance on other open issues such as those introduced in sections 6.2 and 6.4. Furthermore, the expressive abilities of binarized networks are also questionable, and this has resulted in the development of other

quantization-based models such as ternary weight networks (Li, Zhang, & Liu, 2016).

Recently Lavin and Gray (2016) also proposed a new class of DCNN algorithms based on Winograd's minimal filtering algorithms (Winograd, 1980) and achieved speed gains of between 1.48 and 7.42 times when compared to their baseline model from Simonyan and Zisserman (2014). However, their model varied up to a maximum size of 16 MB, which is bigger than the 11.3 MB achieved by Han, Mao, et al. (2016) for the same VGG-net baseline (Simonyan & Zisserman, 2014). Another recent development that allowed for the performance evaluation of several DCNNs on smartphones used Bayesian matrix factorization, Tucker decomposition (see Tucker, 1966) to compress entire DCNN layers, and fine tuning to recover accumulated accuracy losses (Kim et al., 2015).

On the hardware side, Qiao et al. (2016) built a prototype FPGA system to accelerate DCNNs and achieved speed gains of 3.54 times and energy efficiency gains of 7.4 times over baseline CPU and GPU implementations. To perform inference on the compressed model proposed by Han, Mao, et al. (2016), Han, Liu, et al. (2016) proposed an energy-efficient engine that exploits the compression techniques proposed earlier (Han, Mao, et al., 2016) and thus achieved substantial computational and energy gains. Continuing with the model and data parallelism approach introduced by Krizhevsky (2014), Dettmers (2016) recently built a GPU cluster consisting of 96 units and showed that by compressing gradients and nonlinear activations into 8-bit representations, speed-ups of up to 50 times were achievable. Thus, although it is conceivable to conclude that new hardware developments and improved algorithms that specifically consider the hardware architecture will fuel future advances in DCNN computation and storage requirements, the reasons behind the initial training of redundant models also necessitate further investigation; if this can be solved first, the need to counter its consequences can be relaxed. In the interim, although the methods highlighted in this section show promise, further empirical investigations and theoretical motivations to firmly establish them, especially on the diverse real world data sets collected on mobile devices, is required.

6.4 Deep Flaws. Among the open challenges, perhaps the most intriguing of all is that the classification accuracy of DCNNs and classifiers in general is not robust when faced with adversarial examples. These are small yet intentional perturbations applied to images with the aim of misleading or fooling the recognition or classification system. When these perturbations are used to alter an image, humans are easily able to classify the image correctly (Goodfellow, Shlens, & Szegedy, 2015; Uličný, Lundström, & Byttner, 2016), while classifiers see the image as being from a different class. Since the initial discovery of this phenomenon (Szegedy et al., 2014), several papers have confirmed the vulnerability of DCNNs to these images and proposed some viable countermeasures to mitigate them (Szegedy et al.,

2014; Goodfellow et al., 2015; Gu & Rigazio, 2014; Maharaj, 2015; Zhao & Griffin, 2016; Uličný et al., 2016; Jin, Dundar, & Culurciello, 2016; Tabacof & Valle, 2016; Miyato, Maeda, Koyama, Nakae, & Ishii, 2016; Sabour, Cao, Faghri, & Fleet, 2016; Papernot, McDaniel, Xu, Jha, & Swami, 2016; Huang, Xu, Schuurmans, & Szepesvári, 2016).

So far the most promising attempts to solve this issue focus on different training techniques such as adversarial training (Goodfellow et al., 2015) and distillation (Papernot et al., 2016), generative preprocessing methods such as the use of denoising autoencoders (Uličný et al., 2016), and changing DCNN architecture to make it more nonlinear or to penalize unusual signals (Zhao & Griffin, 2016; Jin et al., 2016). Furthermore, a theoretical framework to formally investigate robustness has also been introduced, and the fundamental limits on the robustness of selected classifiers, concerning a distinguishability measure between classes, have been established (Fawzi, Fawzi, & Frossard, 2015a, 2015b). Moreover, Bastani et al. (2016) recently proposed two statistical measures, adversarial frequency and adversarial severity, for measuring robustness based on the formalized conception of point-wise robustness, encoded as a constrained optimization problem. Interestingly, RBF networks have been shown to be intrinsically immune to adversarial examples (Goodfellow et al., 2015), and thus combining their features with standard DCNN architecture, like the work proposed by Zhao and Griffin (2016), remains a suitable approach that requires further investigation. Recently, image discretization has yielded improved adversarial performance (Maharaj, 2015); however, the robustness effect of other dimensionality-reduction techniques on input data and feature vectors from different DCNN layers is yet to be investigated.

Complementary to the finding of adversarial examples, Nguyen, Yosinski, & Clune (2015) found that it was possible to generate images that are entirely unrecognizable to humans but that state-of-the-art DCNNs classify as recognizable objects—alarmingly, with extremely high confidence. These images were generated using evolutionary algorithms and gradient-based optimization and are referred to as fooling images. Referring to these images as being from a rubbish class, Goodfellow et al. (2015) showed that they can be generated by ways that are more efficient and that they affect not only deep networks (this study included DCNNs) but shallow classifiers as well. Supplementary and up-to-date work by Zhao and Griffin (2016) referred to these images as nonsense images and began work showing that DCNN architecture changes are a viable solution to overcome this flaw; however, as with adversarial examples, they remain an ongoing challenge requiring further attention.

Since these adversarial (Szegedy et al., 2014) and fooling (Nguyen et al., 2015) images highlight a vast gap between the vision capabilities of humans and computer vision systems, finding these intriguing properties has brought about several questions regarding the generalization, function approximation, and security features of deep networks (Szegedy et al., 2014;

Goodfellow et al., 2015; Gu & Rigazio, 2014; Maharaj, 2015; Zhao & Griffin, 2016; Uličný et al., 2016; Jin et al., 2016; Tabacof & Valle, 2016; Papernot et al., 2016). This has opened up a completely new area of research focusing on the generation of these images and the design of systems that are robust against them. Naturally, since DCNNs have become the leading architecture for visual tasks and given the fact that they are not immune to both adversarial and fooling images, research into their robustness is a momentous problem.

6.5 Multilabeled Images and Deeper Image Understanding. Even though DCNNs have surpassed human-level performance on single-label image data sets such as the MNIST (LeCun et al., 1998; Ciresan et al., 2012; Wan et al., 2013) and ImageNet (Russakovsky et al., 2015; Ioffe & Szegedy, 2015; Szegedy, Vanhoucke et al., 2015; He et al., 2015a, 2015b; Szegedy et al., 2016) data sets, real-world images usually contain multiple labels, which relate to different objects, parts, scenes, actions, and their interactions or attributes (Wang, Yang, et al., 2016). Furthermore, the ability to correctly describe the semantic content of an image, in properly formed natural language sentences, is a challenging problem (Vinyals, Toshev, Bengio, & Erhan, 2015), which finds itself at the intersection between computer vision and natural language processing. To address these problems, a recent trend is to combine DCNNs with RNNs. To counter the multilabel classification problem, Wang, Yang, et al. (2016) presented a DCNN-RNN framework in which the DCNN extracts semantic representations from images, while the RNN models the image-label relationship and label dependency. Both Vinyals et al. (2015) and Karpathy and Fei-Fei (2016) used DCNNs for image classification and RNNs for sequence modeling and combined them into a unified network, which they used to generate English-language descriptions of images. Another promising direction is to train these combined architectures using reinforcement learning, and although systems combining deep and reinforcement learning are still in their early days (LeCun et al., 2015), they have already produced some exceptional image classification results (Ba, Mnih, & Kavukcuoglu, 2015).

6.6 Other Selected Trends and Challenges. As mentioned in section 4.2.1, despite the contribution of unsupervised pretraining to the deep learning renaissance (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2006), current DCNNs mostly utilize the supervised learning paradigm; thus, they are not able to exploit the massive amounts of unlabeled data available on the Internet, stored in cloud-based systems or even captured by mobile devices. Furthermore, human learning is naturally unsupervised (LeCun et al., 2015), and thus it is expected that future DCNN models will attempt to mimic nature more than our current models do. Recent attempts along these lines include work by Goodfellow et al.

(2014), Kingma and Welling (2014), Bengio, Thibodeau-Laufer, Alain, and Yosinski (2014), Kulkarni, Whitney, Kohli, and Tenenbaum (2015) and, more recently, Bachman (2016), all of which use promising generative-based modeling techniques.

DCNNs require several hyperparameters, such as the number of epochs to run the model and the learning rate; however, determining them requires careful tuning, which is often based on expert experience, rules of thumb, or computationally exorbitant search methods. By using an automated Bayesian optimization technique, Snoek et al. (2012) were able to find better hyperparameters faster than a human expert did and obtained excellent image classification results. However, this method is time-consuming and does not scale well to large models (Srinivas et al., 2016); thus, alternatives along these lines are required. A possible solution may be to use evolutionary algorithms, such as particle swarm optimization (Kennedy & Eberhart, 1995), which has become a popular optimization technique, to conduct the hyperparameter search and then integrate the results with DCNNs. If successful, this will see two popular biologically inspired techniques working in tandem with each other. Another interesting yet challenging direction is to leverage the discriminative and expressive classification abilities of DCNNs in online robotic systems, with some recent successes described by Pinto and Gupta (2015), Finn et al. (2015), and Levine, Finn, Darrell, and Abbeel (2016).

One more trend that is gaining momentum is the factorization of convolutions to improve computational efficiency. The technique was popularized by the revised Inception model (Szegedy, Vanhoucke et al., 2015), which found that $n \times n$ convolutions can be factorized into $1 \times n$ followed by $n \times 1$ convolutions. For example, they found that 3×1 convolutions followed by 1×3 convolutions resulted in a 33% reduction in computation in comparison to using a single 3×3 filter having the same effective receptive field size. Inspired by this, Chollet (2016) proposed the Xception architecture in which they replace the Inception building blocks with depth-wise separable convolutions (depth-wise convolution followed by a point-wise convolution). Depth-wise convolutions was also exploited by Xie, Girshick, Dollar, and He (2016), who repeat a residual building block that concatenates a series of transformations that have an identical topology. The efficiency gains of utilizing this type of factorization, especially as we proceed toward mobile DCNN deployment, is of significant importance and will probably be used extensively in future models.

The most significant challenge is to close the theoretical gap between biological neural networks and DCNNs, and although the fresh theoretical analysis by Bengio, Mesnard, Fischer, Zhang, and Wu (2017) did not specifically deal with DCNNs, their motivation of how the biological brain executes credit assignment in deep hierarchies, perhaps as proficiently as backpropagation does, can be regarded as an important step toward linking our deep computational models to mechanisms of the human brain.

7 Conclusion

This review presents a comprehensive review of CNNs for image classification tasks. It categorizes their progression into their early development, their contribution to the deep learning renaissance, and their rapid advancement over the past few years. In particular, it focuses on their advancement by deliberating and analyzing most of the notable advances in relation to their architectures, supervision components, regularization mechanisms, optimization techniques, and computation since 2012. Despite successes in other domains, DCNNs have seen remarkable progression in image classification tasks, setting the state of the art on several challenging classification benchmarks and dominating numerous image-classification-related challenges and competitions. In fact, on several single label image classification benchmarks, their performance has surpassed human-level performance. However, the contemporary rise of DCNNs has led researchers to scrutinize their classification performance, robustness, and computational characteristics, resulting in the discoveries of several challenges and trends to address them. Accordingly, this review also recapitulates these open issues and their associated trends and, most significant, provides several recommendations and research directions for future exploration.

Acknowledgments

This work was supported in part by South African National Research Foundation Incentive Grant 81705.

References

Abdulkader, A. (2006). A two-tier Arabic offline handwriting recognition based on conditional joining rules. In *Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition* (pp. 1–6). Los Alamitos, CA: IEEE Computer Society.

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). *Learning activation functions to improve deep neural networks*. arXiv 1412.6830.

Ahmed, A., Yu, K., Xu, W., Gong, Y., & Xing, E. (2008). Training hierarchical feedforward visual recognition models using transfer learning from pseudo-tasks. In *Proceedings of the European Conference on Computer Vision* (pp. 69–82). Berlin: Springer.

Arora, S., Bhaskara, A., Ge, R., & Ma, T. (2014). Provable bounds for learning some deep representations. In *Proceedings of the 31th International Conference Machine Learning* (pp. 584–592). N.p.: International Machine Learning Society.

Ba, J., & Frey, B. (2013). Adaptive dropout for training deep neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 26* (pp. 3084–3092). Red Hook, NY: Curran.

Ba, J., Mnih, V., & Kavukcuoglu, K. (2015). Multiple object recognition with visual attention. In *Proceedings of the 3rd International Conference on Learning Representations* (pp. 1–10). N.p.: Computational and Biological Learning Society.

Bachman, P. (2016). An architecture for deep, hierarchical generative models. In D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systemsm, 29* (pp. 4826–4834). Red Hook, NY: Curran.

Baldi, P., & Sadowski, P. J. (2013). Understanding dropout. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 26* (pp. 3084–3092). Red Hook, NY: Curran.

Baldi, P., & Sadowski, P. (2014). The dropout learning algorithm. *Artificial Intelligence*, *210*, 78–122.

Barkan, O., Weill, J., Wolf, L., & Aronowitz, H. (2013). Fast high dimensional vector multiplication face recognition. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1960–1967). Red Hook, NY: Curran.

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., & Criminisi, A. (2016). Measuring neural net robustness with constraints. In D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 2613–2621). Red Hook, NY: Curran.

Basu, S., Karki, M., DiBiano, R., Mukhopadhyay, S., Ganguly, S., Nemani, R., & Gayaka, S. (2016). *A theoretical analysis of deep neural networks for texture classification*. arXiv 1605.02699.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bell, S., & Bala, K. (2015). Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics*, *34*(4), 98–107.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bengio, Y. (2013). Deep learning of representations: Looking forward. In *Proceedings of International Conference on Statistical Language and Speech Processing* (pp. 1–37). Berlin: Springer.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems, 19* (pp. 2814–2822). Red Hook, NY: Curran.

Bengio, Y., Mesnard, T., Fischer, A., Zhang, S., & Wu, Y. (2017). STDP-compatible approximation of backpropagation in an energy-based model. *Neural Computation*, *29*(3), 555–577.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Bengio, Y., Thibodeau-Laufer, E., Alain, G., & Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. In *Proceedings of the 31st International Conference Machine Learning* (pp. 226–234). N.p.: International Machine Learning Society.

Bottou, L. (1998). Online learning and stochastic approximations. *On-Line Learning in Neural Networks*, *17*(9), 142–177.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of the International Conference on Computational Statistics* (pp. 177–186). Berlin: Physica-Verlag Heidelberg.

Boureau, Y., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 111–118). N.p.: International Machine Learning Society.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., . . . Shah, R. (1993). Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, *7*(4), 669–688.

Bulo, S., & Kontschieder, P. (2014). Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 81–88). Los Alamitos, CA: IEEE Computer Society.

Bruna, J., Szlam, A., & LeCun, Y. (2013). *Signal recovery from pooling representations*. arXiv 1311.4025.

Cao, X., Wipf, D., Wen, F., Duan, G., & Sun, J. (2013). A practical transfer learning algorithm for face verification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 3208–3215). Red Hook, NY: Curran.

Chang, J., & Chen, Y. (2015). *Batch-normalized maxout network in network*. arXiv 1511.02583.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). *Return of the devil in the details: Delving deep into convolutional nets*. arXiv 1405.3531.

Chellapilla, K., & Puri, S., & Simard, P. (2006). High performance convolutional neural networks for document processing. In *Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition.* Los Alamitos, CA: IEEE Computer Society.

Chellapilla, K., Shilman, M., & Simard, P. (2006). Optimally combining a cascade of classifiers. In *Proceedings of the 18th Annual Symposium on Electronic Imaging* (pp. 6067–6126). N.p: International Society for Optical Engineering.

Chellapilla, K., & Simard, P. (2006). Two-tier approach for Arabic offline handwriting recognition. In *Proceedings of the 10th Internstional Workshop on Frontiers in Handwriting Recognition* (pp. 1–6). Los Alamitos, CA: IEEE Computer Society.

Chen, D., Cao, X., Wen, F., & Sun, J. (2013). Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3025–3032). Red Hook, NY: Curran.

Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q., & Chen, Y. (2015). *Compressing neural networks with the hashing trick*. arXiv 1504.04788v1.

Cheng, Y., Felix, X. Y., Feris, R. S., Kumar, S., Choudhary, A., & Chang, S. (2015). *Fast neural networks with circulant projections*. arXiv 1502.03436.

Cheng, Y., Yu, F. X., Feris, R. S., Kumar, S., Choudhary, A., & Chang, S. (2015). An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2857–2865). Red Hook, NY: Curran.

Cheng, Z., Soudry, D., Mao, Z., & Lan, Z. (2015). *Training binary multilayer neural networks for image classification using expectation backpropagation*. arXiv 1503.03562.

Chollet, F. (2016). *Xception: Deep learning with depthwise separable convolutions*. arXiv 1610.02357.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 539–546). Los Alamitos, CA: IEEE Computer Society.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Proceedings 18th International Conference on Artificial Intelligence and Statistics* (pp. 192–204). www.jmlr.org/proceedings/papers/v38/choromanska15.pdf

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, *22*(12), 3207–3220.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the International Joint Conference on Artificial Intelligence* (vol. 1, pp. 1237–1242). Menlo Park, CA: AAAI Press.

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3642–3649). Red Hook, NY: Curran.

Clevert, D., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the 4th International Conference on Learning Representations* (pp. 1–14). N.p.: Computational and Biological Learning Society.

Coates, A., Lee, H., & Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (pp. 215–223). www.jmlr.org/proceedings/papers/v15/coates11a/coates11a.pdf

Collobert, R., & Bengio, S. (2004). A gentle Hessian for efficient gradient descent. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 517–520). N.p.: IEEE Signal Processing Society.

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Large scale transductive SVMs. *Journal of Machine Learning Research*, *7*, 1687–1712.

Courbariaux, M., Bengio, Y., & David, J. P. (2015). BinaryConnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 3123–3131). Red Hook, NY: Curran.

Courbariaux, M., Hubara, I., Soudry, D., & Yaniv, R. E. (2016). Binarized neural networks. In D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 1–9). N.p.: Preproceedings.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., . . . Le, Q. V. (2012). Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 25* (pp. 1223–1231). Red Hook, NY: Curran.

Decoste, D., & Schölkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, *46*(1–3), 161–190.

Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, *3*(2), 1–29.

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, *7*(3–4), 197–387.

Denil, M., Shakibi, B., Dinh, L., & de Freitas, N. (2013). Predicting parameters in deep learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *26* (pp. 2148–2156). Red Hook, NY: Curran.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27* (pp. 1269–1277). Red Hook, NY: Curran.

Dettmers, T. (2016). 8-bit approximations for parallelism in deep learning. In *Proceedings of the 4th International Conference on Learning Representations* (pp. 1–14). N.p.: Computational and Biological Learning Society.

Dreyfus, S. (1962). The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications*, *5*(1), 30–45.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

Dumoulin, V., & Visin, F. (2016). *A guide to convolution arithmetic for deep learning*. arXiv 1603.07285.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, *11*, 625–660.

Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2012). *Scene parsing with multiscale feature learning, purity trees, and optimal covers*. arXiv 1202.2160.

Fawzi, A., Fawzi, O., & Frossard, P. (2015a). *Analysis of classifiers' robustness to adversarial perturbations*. arXiv 1502.02590.

Fawzi, A., Fawzi, O., & Frossard, P. (2015b). Fundamental limits on adversarial robustness. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1–7). N.p.: International Machine Learning Society.

Fei-Fei, L. (2006). Knowledge transfer in learning to recognize visual objects classes. In *Proceedings of the 4th International Conference on Development and Learning* (pp. 11–17). N.p.: IEEE Computational Intelligence Society.

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(4), 594–611.

Filho, D. P., Luiz, P., Oliveira, L. S., & Britto, A. S. Jr. (2009). A database for forest species recognition. In *Proceedings of the XXII Brazilian Symposium on Computer Graphics and Image Processing* (pp. 1–2). Red Hook, NY: Curran.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2015). Deep spatial autoencoders for visuomotor learning. arXiv 1509.06113.

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. Cambridge, MA: MIT Press.

Fukushima, K. (1979). Self-organization of a neural network which gives position-invariant response. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence* (vol. 1, pp. 291–293). San Francisco: Morgan Kaufmann.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, *15*(6), 455–469.

Garcia, C., & Delakis, M. (2002). A neural architecture for fast and robust face detection. In *Proceedings of the 16th International Conference on Pattern Recognition* (pp. 44–47). Los Alamitos, CA: IEEE Computer Society.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580–587). Red Hook, NY: Curran.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics* (pp. 249–256). jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (pp. 315–323). www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

Gong, Y., Liu, L., Yang, M., & Bourdev, L. (2014). *Compressing deep convolutional networks using vector quantization*. arXiv 1412.6115.

Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (2014, September). Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of the European Conference on Computer Vision* (pp. 392–407). Berlin: Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 27* (pp. 2672–2680). Red Hook, NY: Curran.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations* (pp. 1–11). N.p.: Computational and Biological Learning Society.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., & Bengio, Y. (2013). Maxout networks. In *Proceedings of the 30th International Conference Machine Learning* (pp. 1319–1327). N.p.: International Machine Learning Society.

Graham, B. (2014). *Fractional max-pooling*. arXiv 1412.6071.

Grauman, K., & Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1458–1465). Red Hook, NY: Curran.

Griffin, G., Holub, A., & Perona, P. (2007). *Caltech-256 object category dataset*. Pasadena: California University of Technology.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., . . . Wang, G. (2015). *Recent advances in convolutional neural networks*. arXiv 1512.07108.

Gu, S., & Rigazio, L. (2014). *Towards deep neural network architectures robust to adversarial examples*. arXiv 1412.5068.

Gulcehre, C., Cho, K., Pascanu, R., & Bengio, Y. (2014). Learned-norm pooling for deep feedforward and recurrent neural networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (pp. 530–546). New York: Springer-Verlag.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, 27–48.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1735–1742). Los Alamitos, CA: IEEE Computer Society.

Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., . . . LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, *26*(2), 120–144.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). *EIE: Efficient inference engine on compressed deep neural network*. arXiv 1602.01528.

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proceedings of the 3rd International Conference on Learning Representations* (pp. 1–14). N.p.: Computational and Biological Learning Society.

Hardt, M., & Ma, T. (2016). *Identity matters in deep learning*. arXiv 1611.04231.

Harris, K. D., & Shepherd, G. M. (2015). The neocortical circuit: Themes and variations. *Nature Neuroscience*, *18*(2), 170–181.

He, K., & Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5353–5360). Red Hook, NY: Curran.

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the European Conference on Computer Vision* (pp. 346–361). Berlin: Springer.

He, K., Zhang, X., Ren, S., & Sun, J. (2015a). Deep residual learning for image recognition. arXiv 1512.03385.

He, K., Zhang, X., Ren, S., & Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026–1034). Red Hook, NY: Curran.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Identity mappings in deep residual networks*. arXiv 1603.05027.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, *40*(1), 185–234.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.

Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. arXiv 1207.0580.

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the knowledge in a neural network*. arXiv 1503.02531.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Huang, F. J., & LeCun, Y. (2006). Large-scale learning with SVM and convolutional nets for generic object categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 284–291). Red Hook, NY: Curran.

Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007). *Labeled faces in the wild: A database for studying face recognition in unconstrained environments* (vol. 1, p. 3) (Technical Report 07-49). Amherst: University of Massachusetts.

Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2016). *Densely connected convolutional networks*. arXiv 1608.06993.

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *Proceedings of the European Conference on Computer Vision* (pp. 646–661). Heidelberg, Berlin: Springer.

Huang, R., Xu, B., Schuurmans, D., & Szepesvári, C. (2016). *Learning with a strong adversary*. arXiv 1511.03034v6.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, *148*(1), 574–591.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, *160*(1), 106–154.

Hyvärinen, A., & Köster, U. (2007). Complex cell pooling and the statistics of natural images. *Network: Computation in Neural Systems*, *18*(2), 81–100.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., & Keutzer, K. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size*. arXiv 1602.07360.

Ioffe, S. (2017). *Batch renormalization: Towards reducing minibatch dependence in batch-normalized models*. arXiv 1702.03275.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference Machine Learning* (pp. 448–456). N.p.: International Machine Learning Society.

Ivakhnenko, A. G., & Lapa, V. G. (1966). *Cybernetic predicting devices.* New York: CCM Information Corp.

Jaderberg, M., Simonyan, K., & Zisserman, A. (2015). Spatial transformer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 2017–2025). Red Hook, NY: Curran.

Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference* (pp. 1–12). Durham, UK: BMVA Press.

Jarrett, K., Kavukcuoglu, K., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2146–2153). Red Hook, NY: Curran.

Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., & Schmid, C. (2012). Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(9), 1704–1716.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., . . . Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM International Conference on Multimedia* (pp. 675–678). New York: ACM.

Jin, J., Dundar, A., & Culurciello, E. (2016). Robust convolutional neural networks under adversarial noise. In *Proceedings of the 4th International Conference on Learning Representations* (pp. 1–8). N.p.: Computational and Biological Learning Society.

Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2015). *Deep learning with S-shaped rectified linear activation units*. arXiv 1512.07030.

Kalchbrenner, N., Danihelka, I., & Graves, A. (2015). *Grid long short-term memory*. arXiv 1507.01526.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. V. D., Graves, A., & Kavukcuoglu, K. (2016). *Neural machine translation in linear time*. arXiv 1610.10099.

Karpathy, A. (2016). *CS231n: Convolutional neural networks for visual recognition*. http://cs231n.github.io/classification/

Karpathy, A., & Fei-Fei, L. (2016). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3128–3137). Red Hook, NY: Curran.

Kavukcuoglu, K., Ranzato, M., Fergus, R., & LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1605–1612). Red Hook, NY: Curran.

Kavukcuoglu, K., Ranzato, M., & LeCun, Y. (2010). *Fast inference in sparse coding algorithms with applications to object recognition*. arXiv 1010.3467.

Kennedy, J., & Eberhart, R. C. (1995, October). A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science* (pp. 39–43). Piscataway, NJ: IEEE.

Kim, M., & Smaragdis, P. (2016). *Bitwise neural networks*. arXiv 1601.06071.

Kim, Y., Park, E., Yoo, S., Choi, T., Yang, L., & Shin, D. (2015). *Compression of deep convolutional neural networks for fast and low power mobile applications*. arXiv 1511.06530.

Kingma, D., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv 1412.6980.

Kingma, D. P., & Welling, M. (2014). *Auto-encoding variational Bayes*. arXiv 1312.6114 v10.

Koushik, J. (2016). *Understanding convolutional neural networks*. arXiv 1605.09081.

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images.* Master's thesis, University of Toronto, Canada.

Krizhevsky, A. (2014). *One weird trick for parallelizing convolutional neural networks*. arXiv 1404.5997.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 25* (pp. 1097–1105). Red Hook, NY: Curran.

Kulkarni, T. D., Whitney, W. F., Kohli, P., & Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systemsm, 28* (pp. 2539–2547). Red Hook, NY: Curran.

Kumar, N., Berg, A. C., Belhumeur, P. N., & Nayar, S. K. (2009). Attribute and simile classifiers for face verification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 365–372). Red Hook, NY: Curran.

Laptev, D., Savinov, N., Buhmann, J. M., & Pollefeys, M. (2016). *TI-POOLING: Transformation-invariant pooling for feature learning in convolutional neural networks*. arXiv 1604.06318.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 473–480). N.p.: International Machine Learning Society.

Lavin, A., & Gray, S. (2016). Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4013–4021). Red Hook, NY: Curran.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, *8*(1), 98–113.

Lazebnik, S., Schmid, C., & Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(8), 1265–1278.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2169–2178). Red Hook, NY: Curran.

Learned-Miller, E., Huang, G. B., RoyChowdhury, A., Li, H., & Hua, G. (2016). Labeled faces in the wild: A survey. In M. Kawulok, M. E. Celebi, & B. Smolka (Eds.), *Advances in face detection and facial image analysis* (pp. 189–248). Cham, Switzerland: Springer.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., & Lempitsky, V. (2014). *Speeding-up convolutional neural networks using fine-tuned CP-decomposition*. arXiv 1412.6553.

LeCun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.), *Connections in perspective* (pp. 143–155). Zurich, Switzerland: Elsevier.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989a). Handwritten digit recognition with a back-propagation network. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, 2* (pp. 396–404). Cambridge, MA: MIT Press.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989b). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Huang, F. J., & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 97–104). Red Hook, NY: Curran.

LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (pp. 253–256). Red Hook, NY: Curran.

Lee, C., Gallagher, P. W., & Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (pp. 464–472). www.jmlr.org/proceedings/papers/v51/lee16a.pdf

Lee, C., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics* (pp. 562–570). jmlr.org/proceedings/papers/v38/lee15a.pdf

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference Machine Learning* (pp. 609–616). N.p.: International Machine Learning Society.

Leibo, J. Z., Mutch, J., & Poggio, T. (2011). Why the brain separates face recognition from object recognition. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 24* (pp. 711–719). Red Hook, NY: Curran.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, *17*(39), 1–40.

Li, F., Zhang, B., & Liu, B. (2016). *Ternary weight networks*. arXiv 1605.04711.

Li, S., Jiao, J., Han, Y., & Weissman, T. (2016). *Demystifying ResNet*. arXiv 1611.01186.

Li, Z., Gong, B., & Yang, T. (2016). Improved dropout for shallow and deep learning. In D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systemsm* (pp. 1–9). N.p.: Preproceedings.

Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3367–3375). Red Hook, NY: Curran.

Liao, Z., & Carneiro, G. (2016). On the importance of normalisation layers in deep learning with piecewise linear activation units. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision* (pp. 1–8). Red Hook, NY: Curran.

Lin, J., Morere, O., Chandrasekhar, V., Veillard, A., & Goh, H. (2015). *Deephash: Getting regularization, depth and fine-tuning right*. arXiv 1501.04711.

Lin, M., Chen, Q., & Yan, S. (2013). *Network in network*. arXiv 1312.4400.

Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., . . . Huang, T. (2011). Large-scale image classification: Fast feature extraction and SVM training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1689–1696). Red Hook, NY: Curran.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Master's thesis, University of Helsinki, Finland.

Littwin, E., & Wolf, L. (2016). *The loss surface of residual networks: Ensembles and the role of batch normalization*. arXiv 1611.02525.

Liu, H., Tian, Y., Yang, Y., Pang, L., & Huang, T. (2016). Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2167–2175). Red Hook, NY: Curran.

Liu, W., Wen, Y., Scut, M., Yu, Z., & Yang, M. (2016). Large-margin softmax loss for convolutional neural networks. In *Proceedings of the 33rd International Conference Machine Learning* (pp. 507–516). N.p.: International Machine Learning Society.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference Machine Learning* (pp. 1–8). N.p.: International Machine Learning Society.

Maharaj, A. V. (2015). *Improving the adversarial robustness of ConvNets by reduction of input dimensionality*. Stanford, CA: Department of Physics, Stanford University.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2008). Discriminative learned dictionaries for local image analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). Red Hook, NY: Curran.

Malinowski, M., & Fritz, M. (2013). *Learnable pooling regions for image classification*. arXiv 1301.3516.

Mallat, S. (2012). Group invariant scattering. *Communications on Pure and Applied Mathematics, 65*(10), 1331–1398. doi:10.1002/cpa.21413

Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J. (2011). Stacked convolutional autoencoders for hierarchical feature extraction. In *Proceedings of the 21th International Conference on Artificial Neural Networks* (pp. 52–59). Berlin: Springer.

Mathieu, M., Henaff, M., & LeCun, Y. (2013). *Fast training of convolutional networks through FFTs*. arXiv 1312.5851.

Mishkin, D., & Matas, J. (2016). All you need is a good init. In *Proceedings of the 4th International Conference on Learning Representations* (pp. 1–13). N.p.: Computational and Biological Learning Society.

Miyato, T., Maeda, S., Koyama, M., Nakae, K., & Ishii, S. (2016). Distributional smoothing with virtual adversarial training. In *Proceedings of the 5th International Conference on Learning Representations* (pp. 1–12). N.p.: Computational and Biological Learning Society.

Montavon, G., Orr, G. B., & Müller, K. (Eds.). (2012). *Neural networks: Tricks of the trade* (2nd ed.). Berlin: Springer.

Muller, U., Ben, J., Cosatto, E., Flepp, B., & LeCun, Y. (2005). Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, P. B. Schölkopf, & J. C. Platt (Eds.), *Advances in neural information processing systems, 18* (pp. 739–746). Cambridge, MA: MIT Press.

Nagi, J., Di Caro, G. A., Giusti, A., Nagi, F., & Gambardella, L. M. (2012). Convolutional neural support vector machines: Hybrid visual pattern classifiers for multi-robot systems. In *Proceedings of the 11th International Conference on Machine Learning and Applications* (pp. 27–32). Los Alamitos, CA: IEEE Computer Society.

Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., . . . Gambardella, L. M. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Proceedings of the IEEE International Conference on Signal and Image Processing Applications* (pp. 342–347). Red Hook, NY: Curran.

Nair, V., & Hinton, G. E. (2009). 3D object recognition with deep belief nets. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems*, *22* (pp. 1339–1347). Red Hook, NY: Curran.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 807–814). N.p.: International Machine Learning Society.

Nasse, F., Thurau, C., & Fink, G. A. (2009). Face detection using GPU-based convolutional neural networks. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns* (pp. 83–90). Berlin: Springer.

National Data Science Bowl | Kaggle. (2016). *Kaggle.com*. https://www.kaggle.com/c/datasciencebowl

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Advances in neural information processing systems, 24 (NIPS) Workshop on Deep Learning and Unsupervised Feature Learning* (pp. 1–9). Red Hook, NY: Curran.

Ngiam, J., Chen, Z., Chia, D., Koh, P. W., Le, Q. V., & Ng, A. Y. (2010). Tiled convolutional neural networks. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems, 23* (pp. 1279–1287). Red Hook, NY: Curran.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 427–436). Los Alamitos, CA: IEEE Computer Society.

Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., & Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, *14*(9), 1360–1371.

Novikov, A., Podoprikhin, D., Osokin, A., & Vetrov, D. P. (2015). Tensorizing neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 442–450). Red Hook, NY: Curran.

Nowlan, S. J., & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, *4*(4), 473–493.

Oh, K., & Jung, K. (2004). GPU implementation of neural networks. *Pattern Recognition*, *37*(6), 1311–1314.

Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). *Wavenet: A generative model for raw audio*. CoRR abs/1609.03499

Orhan, A. E. (2017). Skip connections as effective symmetry-breaking. arXiv 1701.09175.

Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, *33*(5), 2295–2317.

Oxholm, G., Bariya, P., & Nishino, K. (2012). The scale of geometric texture. In *Proceedings of the European Conference on Computer Vision* (pp. 58–71). Berlin: Springer.

Paine, T., Jin, H., Yang, J., Lin, Z., & Huang, T. (2013). *GPU asynchronous stochastic gradient descent to speed up neural network training*. arXiv 1312.6186.

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the 37th IEEE Symposium on Security and Privacy* (pp. 1–16). Los Alamitos, CA: IEEE Computer Society.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., & Hinton, G. (2017). *Regularizing neural networks by penalizing confident output distributions*. arXiv 1701.06548v1.

Perronnin, F., Sánchez, J., & Mensink, T. (2010). Improving the Fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision* (pp. 143–156). Berlin: Springer.

Pinto, L., & Gupta, A. (2015). *Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours*. arXiv 1509.06825.

Qiao, Y., Shen, J., Xiao, T., Yang, Q., Wen, M., & Zhang, C. (2016). FPGA-accelerated deep convolutional neural networks for high throughput and energy efficiency. *Concurrency and Computation: Practice and Experience*. doi:10.1002/cpe.3850

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, *12*, 145–150.

Ranzato, M. A., Huang, F. J., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). Los Alamitos, CA: IEEE Computer Society.

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In P. B. Schölkopf, J. C. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems, 19* (pp. 1137–1144). Cambridge, MA: MIT Press.

Ranzato, M. A., & Szummer, M. (2008). Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 792–799). N.p.: International Machine Learning Society.

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision* (pp. 525–542). Berlin: Springer.

Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 806–813). Los Alamitos, CA: IEEE Computer Society.

Recht, B., Re, C., Wright, S., & Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *24* (pp. 693–701). Red Hook, NY: Curran.

Rippel, O., Gelbart, M. A., & Adams, R. P. (2014). Learning ordered representations with nested dropout. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 1746–1754). N.p.: International Machine Learning Society.

Rippel, O., Snoek, J., & Adams, R. P. (2015). Spectral representations for convolutional neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 2449–2457). Red Hook, NY: Curran.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations* (pp. 1–13). N.p.: Computational and Biological Learning Society.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., . . . Bernstein, M. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Sabour, S., Cao, Y., Faghri, F., & Fleet, D. J. (2016). Adversarial manipulation of deep representations. In *Proceedings of the 4th International Conference on Learning Representations* (pp. 1–18). N.p.: Computational and Biological Learning Society.

Sainath, T. N., Kingsbury, B., Mohamed, A., Dahl, G. E., Saon, G., Soltau, H., . . . Ramabhadran, B. (2013). Improvements to deep convolutional neural networks for LVCSR. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding* (pp. 315–320). Red Hook, NY: Curran.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., & Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 6655–6659. N.p.: IEEE Signal Processing Society).

Salakhutdinov, R., & Hinton, G. E. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics* (pp. 412–419). http://www.jmlr.org/proceedings/papers/v2/salakhutdinov07a/salakhutdinov07a.pdf

Sánchez, J., & Perronnin, F. (2011). High-dimensional signature compression for large-scale image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1665–1672). Los Alamitos, CA: IEEE Computer Society.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*. arXiv 1312.6120.

Sercu, T., & Goel, V. (2016). *Dense prediction on sequences with time-dilated convolutions for speech recognition*. arXiv 1611.09288.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks* (pp. 92–101). Berlin: Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 815–823). Los Alamitos, CA: IEEE Computer Society.

Sermanet, P., Chintala, S., & LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition* (pp. 3288–3291). Red Hook, NY: Curran.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003, August). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the 7th International Conference on Document Analysis and Recognition* (vol. 3, pp. 958–963). Washington, DC: IEEE Computer Society.

Simoncelli, E. P., & Heeger, D. J. (1998). A model of neuronal responses in visual area MT. *Vision Research*, *38*(5), 743–761.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv 1409.1556.

Smirnov, E. A., Timoshenko, D. M., & Andrianov, S. N. (2014). Comparison of regularization methods for ImageNet classification with deep convolutional neural networks. *AASRI Procedia*, *6*, 89–94.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 25* (pp. 2951–2959). Red Hook, NY: Curran.

Sontag, E. D. (1998). VC dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, *168*, 69–96.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). *Striving for simplicity: The all convolutional net*. arXiv 1412.6806.

Springenberg, J. T., & Riedmiller, M. (2013). Improving deep neural networks with probabilistic maxout units. arXiv 1312.6116.

Srinivas, S., Sarvadevabhatla, R. K., Mopuri, K. R., Prabhu, N., Kruthiventi, S. S., & Babu, R. V. (2016). *A taxonomy of deep convolutional neural nets for computer vision*. arXiv 1601.06615.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Srivastava, N., & Salakhutdinov, R. R. (2013). Discriminative transfer learning with tree-based priors. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 26* (pp. 2094–2102). Red Hook, NY: Curran.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015a). Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 2377–2385). Red Hook, NY: Curran.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015b). *Highway networks*. arXiv 1505.00387.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011, July). The German traffic sign recognition benchmark: A multi-class classification competition. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (pp. 1453–1460). Red Hook, NY: Curran.

Steinkrau, D., Simard, P. Y., & Buck, I. (2005). Using GPUs for machine learning algorithms. In *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 1115–1119).Washington, DC: IEEE Computer Society.

Stollenga, M. F., Masci, J., Gomez, F., & Schmidhuber, J. (2014). Deep networks with internal selective attention through feedback connections. In Z. Ghahramani,

M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 27* (pp. 3545–3553). Red Hook, NY: Curran.

Sun, Y., Chen, Y., Wang, X., & Tang, X. (2014). Deep learning face representation by joint identification-verification. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 27* (pp. 1988–1996). Red Hook, NY: Curran.

Sun, Y., Liang, D., Wang, X., & Tang, X. (2015). *Deepid3: Face recognition with very deep neural networks*. arXiv 1502.00873.

Sun, Y., Wang, X., & Tang, X. (2014). Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1891–1898). Los Alamitos, CA: IEEE Computer Society.

Sun, Y., Wang, X., & Tang, X. (2015). Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2892–2900). Los Alamitos, CA: IEEE Computer Society.

Sussillo, D., & Abbott, L. (2014). *Random walk initialization for training very deep feedforward networks*. arXiv 1412.6558.

Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference Machine Learning* (pp. 1139–1147). N.p.: International Machine Learning Society.

Szegedy, C., Ioffe, S., & Vanhoucke, V. (2016). *Inception-v4, Inception-Resnet and the impact of residual connections on learning*. arXiv 1602.07261.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9). Los Alamitos, CA: IEEE Computer Society.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). *Rethinking the Inception architecture for computer vision*. arXiv 1512.00567.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *Proceedings of the 1st International Conference on Learning Representations* (pp. 1–10). N.p.: Computational and Biological Learning Society.

Tabacof, P., & Valle, E. (2016). Exploring the space of adversarial images. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1–8). Red Hook, NY: Curran.

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1701–1708). Los Alamitos, CA: IEEE Computer Society.

Tang, Y. (2013). *Deep learning using linear support vector machines*. arXiv 1306.0239.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 648–656). Los Alamitos, CA: IEEE Computer Society.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311.

Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., ... Seung, H. S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, *22*(2), 511–538.

Uličný, M., Lundström, J., & Byttner, S. (2016). Robustness of deep convolutional neural networks for image recognition. In *Proceedings of the 1st International Symposium on Intelligent Computing Systems* (pp. 16–30). Switzerland: Springer International Publishing.

Van Dyk, D. A., & Meng, X. (2012). The art of data augmentation. *Journal of Computational and Graphical Statistics*, *10*(1), 1–50.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer Science & Business Media.

Vapnik, V. N., & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications. 16*(2), 11–30.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3156–3164). Red Hook, NY: Curran.

Wager, S., Wang, S., & Liang, P. S. (2013). Dropout training as adaptive regularization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 26* (pp. 351–359). Red Hook, NY: Curran.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., & Fergus, R. (2013). Regularization of neural networks using Dropconnect. In *Proceedings of the 30th International Conference Machine Learning* (pp. 1058–1066). N.p.: International Machine Learning Society.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). *CNN-RNN: A unified framework for multi-label image classification*. arXiv 1604.04573.

Wang, S. I., & Manning, C. D. (2013). Fast dropout training. In *Proceedings of the 30th International Conference Machine Learning* (pp. 118–126). N.p.: International Machine Learning Society.

Wang, Y., Xu, C., You, S., Tao, D., & Xu, C. (2016). CNNpack: Packing convolutional neural networks in the frequency domain. In D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 1–9). N.p.: Preproceedings.

Wang, Z., & Oates, T. (2015). Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 40–46).

Warde-Farley, D., Goodfellow, I. J., Courville, A., & Bengio, Y. (2013). *An empirical analysis of dropout in piecewise linear networks*. arXiv 1312.6197.

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, P. B. Schölkopf, & J. C. Platt (Eds.), *Advances in neural information processing systems, 18* (pp. 1473–1480). Cambridge, MA: MIT Press.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Doctoral dissertation, Harvard University.

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In R. F. Drenick & F. Kozin (Eds.), *System modeling and optimization* (pp. 762–770). Berlin: Springer.

Weston, J., Ratle, F., Mobahi, H., & Collobert, R. (2008). Deep learning via semisupervised embedding. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1168–1175). N.p.: International Machine Learning Society.

Wiatowski, T., & Bölcskei, H. (2015). *A mathematical theory of deep convolutional neural networks for feature extraction*. arXiv 1512.06293.

Winograd, S. (1980). *Arithmetic complexity of computations.* Philadelphia: SIAM.

Wolf, L., Hassner, T., & Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 529–534). Red Hook, NY: Curran.

Wu, H., & Gu, X. (2015). Max-pooling dropout for regularization of convolutional neural networks. In *Proceedings of the 22nd International Conference Neural Information Processing* (pp. 46–53). Berlin: Springer.

Wu, J., Yu, Y., Huang, C., & Yu, K. (2015). Deep multiple instance learning for image classification and auto-annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3460–3469). Red Hook, NY: Curran.

Xie, L., Wang, J., Wei, Z., Wang, M., & Tian, Q. (2016). DisturbLabel: Regularizing CNN on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4753–4762). Red Hook, NY: Curran.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2016). *Aggregated residual transformations for deep neural networks*. arXiv 1611.05431.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical evaluation of rectified activations in convolutional network*. arXiv 1505.00853v2.

Yadan, O., Adams, K., Taigman, Y., & Ranzato, M. (2014). *Multi-GPU training of convnets*. arXiv 1312.5853v4.

Yang, J., Yu, K., Gong, Y., & Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1794–1801). Red Hook, NY: Curran.

Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *Proceedings of the 9th International Conference on Rough Sets and Knowledge Technology* (pp. 364–375). Berlin: Springer.

Yu, F., & Koltun, V. (2015). *Multi-scale context aggregation by dilated convolutions*. arXiv 1511.07122.

Yu, W., Yang, K., Bai, Y., Yao, H., & Rui, Y. (2014a). DNN flow: DNN feature pyramid based image matching. In *Proceedings of the British Machine Vision Conference* (pp. 1–10). Durham, UK: BMVA Press.

Yu, W., Yang, K., Bai, Y., Yao, H., & Rui, Y. (2014b). *Visualizing and comparing convolutional neural networks*. arXiv 1412.6631.

Zagoruyko, S., & Komodakis, N. (2017). *Wide residual networks*. arXiv 1605.07146v3.

Zeiler, M. D. (2012). *ADADELTA: An adaptive learning rate method*. arXiv 1212.5701.

Zeiler, M. D., & Fergus, R. (2013). *Stochastic pooling for regularization of deep convolutional neural networks*. arXiv 1301.3557.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision* (pp. 818–833). Berlin: Springer.

Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2018–2025). Red Hook, NY: Curran.

Zhai, S., Cheng, Y., & Zhang, Z. M. (2016). Doubly convolutional neural networks. In D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 1–9). N.p.: Preproceedings.

Zhang, H., Berg, A. C., Maire, M., & Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2126–2136). Red Hook, NY: Curran.

Zhao, Q., & Griffin, L. D. (2016). *Suppressing the unusual: Towards robust CNNs using symmetric activation functions*. arXiv 1603.05145v1.

Zhou, E., Cao, Z., & Yin, Q. (2015). *Naive-deep face recognition: Touching the limit of LFW benchmark or not?* arXiv 1501.04690.

Zhu, Z., Luo, P., Wang, X., & Tang, X. (2014). *Recover canonical-view faces in the wild with deep neural networks*. arXiv 1404.3543.

Zhuang, Y., Chin, W., Juan, Y., & Lin, C. (2013). A fast parallel SGD for matrix factorization in shared memory systems. *Proceedings of the 7th ACM Conference on Recommender Systems* (pp. 249–256). New York: ACM.

Zinkevich, M., Weimer, M., Li, L., & Smola, A. J. (2010). Parallelized stochastic gradient descent. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systemsm, 23* (pp. 2595–2603). Red Hook, NY: Curran.

Received January 15, 2017; accepted March 25, 2017.