A Review on Deep Convolutional Neural Networks

Neena Aloysius and Geetha M

***Abstract*—The success of traditional methods for solving computer vision problems heavily depends on the feature extraction process. But Convolutional Neural Networks (CNN) have provided an alternative for automatically learning the domain specific features. Now every problem in the broader domain of computer vision is re-examined from the perspective of this new methodology. Therefore it is essential to figure-out the type of network specific to a problem. In this work, we have done a thorough literature survey of Convolutional Neural Networks which is the widely used framework of deep learning. With AlexNet as the base CNN model, we have reviewed all the variations emerged over time to suit various applications and a small discussion on the available frameworks for the implementation of the same. We hope this piece of article will really serve as a guide for any neophyte in the area.**

***Index Terms*—convolutional neural networks, computer vision, deep leaning, classification**

I. Introduction

Convolutional Neural Network is the widely used deep learning framework [1] which was inspired by the visual cortex of animals [2]. Initially it had been widely used for object recognition tasks but now it is being examined in other domains as well like object tracking [3], pose estimation [4], text detection and recognition [5], visual saliency detection [6], action recognition [7], scene labeling [8] and many more [9].

The neocognitron in 1980 [10] is considered as the predecessor of ConvNets. LeNet was the pioneering work in Convolutional Neural Networks by LeCun et al. in 1990 [11] and later improved on it [12]. It was specifically designed to classify handwritten digits and was successful in recognizing visual patterns directly from the input image without any pre-processing. But, due to lack of sufficient training data and computing power, this architecture failed to perform well in complex problems. Later in 2012, Krizhevsky et al. [13] had come up with a CNN model that succeeded in bringing down the error rate on ILSVRC competition [14]. Over the years later, their work has become one of the most influential one in the field of computer vision and used by many for trying out variations in CNN architecture. AlexNet was able to achieve remarkable results compared to the previous model of

ConvNets [15, 16, 17, 18, 19, 20], using purely supervised learning and without any unsupervised pre-training to keep the net simple. The architecture can be considered as a major variant of LeNet having five convolutional layers followed by three fully-connected layers. There have been various variations of AlexNet since its huge success in ILSVRC-2012 competitions. This article will serve as a guide for beginners in the area.

The set of paper is mentioned as below. Section II, describes the convnet layers. The activation functions of these work is described in Section III. Section IV says about the CNN training and testing. Section V describes the architectures of common CNN. Section VI describes the implementation. Section VII tells about the open issues and at last, Section VIII concludes the paper.

II. ConvNet Layers: Structure explained with variations introduced at each stage

Convnets are very similar to normal neural networks which can be visualized as a collection of neurons arranged as an acyclic graph. The main difference from a neural network is that a hidden layer neuron is only connected to a subset of neurons in the previous layer. Because of this sparse connectivity it is capable to learn features implicitly. The deep architecture of the network results in hierarchical feature extraction i.e. the trained filters of first layer can be visualized as set of edges or color blobs, of second layer as some shapes, the next layer filters might learn object parts and the filters of final layers can identify the objects.

A. Convolutional Layer

This layer forms the basic unit of a ConvNet where most of the computation is involved. It is a set of feature maps with neurons arranged in it. The parameters of the layer are a set of learnable filters or kernels. These filters are convolved with the feature maps to produce a separate 2-dimensional activation map which when stacked together along the depth dimension, produce the output volume. Neurons that lie in the same feature map shares the weight (parameter sharing) thereby reducing the complexity of network by keeping the number of parameters low[21]. The spatial extend of sparse connectivity between the neurons of two layers is a hyperparamter called receptive field. The hyperparameters that control the size of the output volume are the depth(number of filters at a layer), stride(for filter movement) and zero-padding(to control spatial size of output). The ConvNets are trained with Backpropagation and the backward pass as well involves convolution operation but with spatially flipped filters. Fig. 1. shows the basic convolution operation of a convnet.

Neena Aloysius is with the Dept. of Computer Science and Engineering, Amrita School of Engineering, Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India. (e-mail: neenaloysius@ymail.com).

Geetha M is with the Dept. of Computer Science and Engineering, Amrita School of Engineering, Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India. (e-mail: geetham@am.amrita.edu).

978-1-5090-3800-8/17/$31.00 ©2017 IEEE

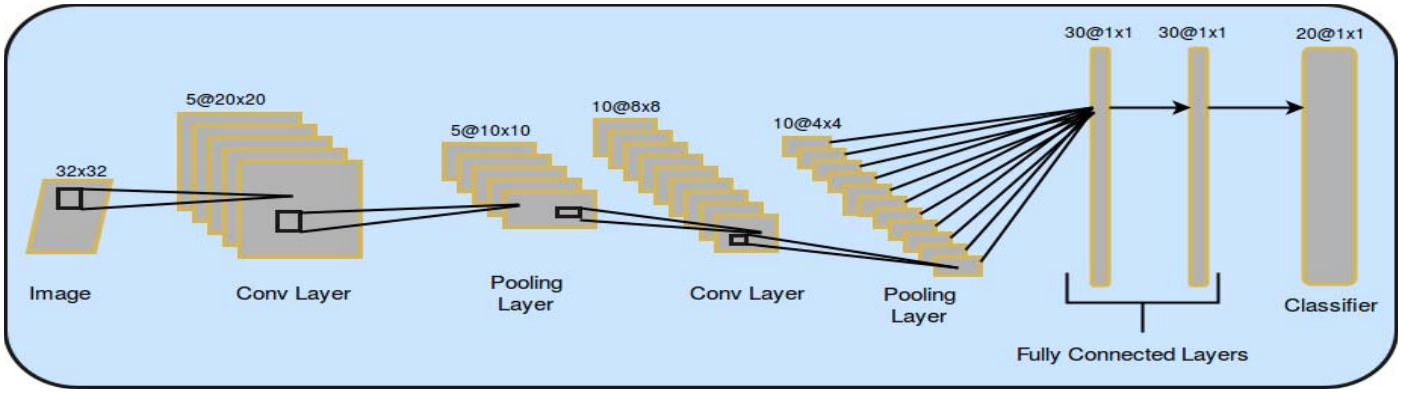

Fig 1. Basic ConvNet architecture showing alternating convolution and pooling layers. The highlighted small boxes are the receptive regions. The connections show implicit hierarchical learning of features.

One of the variants of the traditional CNN is "Network In Network"(NIN) proposed by Lin et al. [22], where the 1*1 convolution filter used is a Multi-Layer Perceptron(mlp) instead of the conventional linear filters and the fully connected layers are replaced by a global average pooling layer. The resultant structure is called mlpconv layer since the micro network consists of stack of mlpconv layers. Unlike CNN, NIN is capable of enhancing the abstraction ability of the latent concepts. They were successful in proving that the last mlpconv layer of NIN were confidence maps of the categories which leads to the possibility of performing object

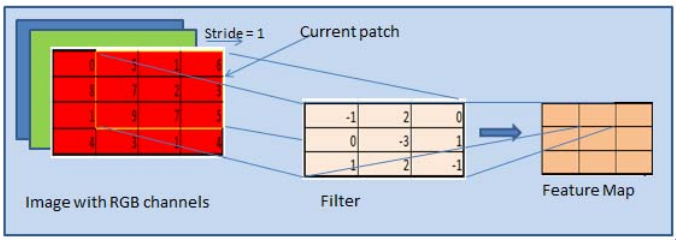

Fig. 2. Convolution Operation. The input image with Red (R), Green (G) and Blue (B) color channels, whose current receptive region is highlighted as a yellow box. The convolution operation involves computing dot products with corresponding elements of receptive region (of R,G,B channels) and filter. The receptive field window slides through the image, spatially computing inner products and resulting in a feature map.

recognition via NIN. In the recent work by Fisher Yu and Vladlen Koltun [23], CNN, which have been originally devised for image classification have been adapted for dense prediction problems such as semantic segmentation that are structurally different from image classification.

Dilated convolutions are used which support exponential expansion of the receptive field without loss of resolution. They have dilated the convolution operator and do not involve construction of dilated filters. The architecture designed specifically for dense prediction are arranged as rectangular prism of convolution layers with no pooling or sub-sampling in contrast to the traditional pyramidal structure used in image classification. And this has resulted in state of the art results for dense predictions.

B. Pooling Layer

Basic ConvNet architecture have alternating conv layers and pooling layers and the latter functions to reduce the spatial dimension of the activation maps (without loss of information) and the number of parameters in the net and thus reducing the overall computational complexity. This controls the problem of overfitting. Some of the common pooling operations are max pooling, average pooling, stochastic pooling [24], spectral pooling [25], spatial pyramid pooling [26] and multi-scale orderless pooling [27]. Fig. 2 shows the operation of max pooling.

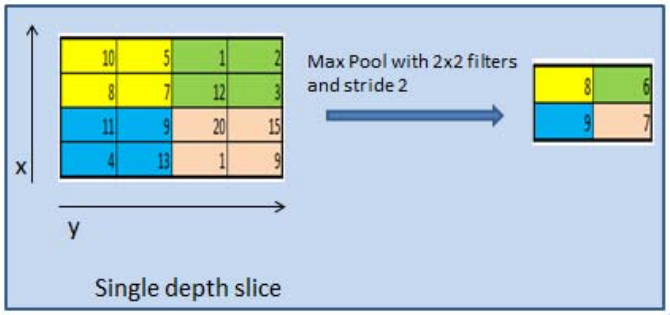

Fig. 3. Max-Pooling Operation

The work by Alexey Dosovitskiy et al. [28] questions the need for different components of a ConvNet and has found that max pooling layers can be replaced with conv layers with stride of two. And this is mainly applicable for simple nets which have proved to outperform many existing complex architecture.

C. Fully Connected Layer

Neurons in this layer are fully connected to all neurons in the previous layer, as in a regular Neural Network. High level reasoning is done here. The neurons are not spatially arranged (one dimensional) so there cannot be a conv layer after a fully connected layer. Recently, some architecture have their FC layer replaced, as in "Network In Network"(NIN) by Lin et al. [29], where fully connected layer is replaced by a global average pooling layer.

$$f(x)=max(0,x) \tag{1}$$

D. Loss Layer

The last fully connected layer serves as the loss layer that computes the loss or error which is a penalty for discrepancy between desired and actual output. For predicting a single class out of K mutually exclusive classes Softmax loss is used. It is the commonly used loss function. Basically it is multinomial logistic regression. It maps the predictions to non-

negative values and also normalized to get probability distribution over classes. Large margin classifier, Support Vector Machine, is trained by calculating Hinge loss. For regressing to real-valued labels Euclidean loss can be used.

III. ACTIVATION FUNCTIONS

Activation functions are non-linarites that take on a single number and do some mathematical operations on it. Many such functions exist and the widely used ones are discussed here.

Sigmoid: This non-linearity takes as input a real-valued function and outputs value (as shown in Equation (1)) in the range of 0 and 1. It is widely used in the area of Neural Networks but with CNN it has two major drawbacks - *(i)* saturate and vanishes gradient *(ii)* sigmoid outputs are not zero-centered causing gradients to oscillate between positive and negative values.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

Tanh: From Equation (2) it is clear that Tanh can be considered as a scaled up version of sigmoid, outputting values in the range of -1 and 1. The problem of saturating gradients exists with this function as well but since the outputs are zero-centered, the second problem stated above is eliminated. Thus practically tanh is preferred to sigmoid.

$$\tanh(x) = 2f(2x) - 1 \tag{3}$$

ReLU: It is a linear activation function which has thresholding at zero as shown in Equation (3). The convergence of gradient descent can be accelerated on applying ReLU.

$$rect(x) = max(0, x) \tag{4}$$

Leaky ReLU: Slight modification to ReLU resulted in Leaky ReLU given by Equation (4).

$$f(x) = \begin{cases} x, & x \leq 0 \\ 0.01x, & otherwise \end{cases} \tag{5}$$

REGULARIZATION

One of the main concerns of deep convnets is the problem of overfitting. Regularization is the technique employed to overcome this problem. Dropout [30] and DropConnect [31] are the two main regularization techniques.

In the above Fig. (i) shows a standard neural network. Applying Dropout as shown in (ii) is equivalent to sampling a neural network. The activations of some of the neurons are set to zero during forward and backward passes of training. While DropConnect randomly set the link weights to zero this is marked red in (iii) in the fig.

IV. CNN TRAINING AND TESTING

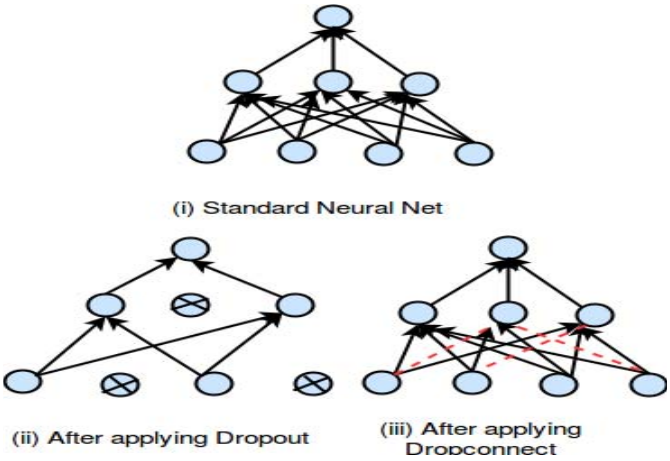

Fig. 4. CNN with many layers

Training a CNN with many layers is a tedious process due to non-convex nature of loss function [32] and requires high computing GPUs if the datasets are huge. Rarely are CNN trained from scratch with random initialization. Convergence can be accelerated with proper network initialization [33] [34]. The technique of Transfer Learning can be employed to have a meaningful weight initialization or if the training dataset is small. In that case, only the classifier or the last few layers are required to be retrained, thereby reducing the overall training time. With careful selection of parameters like learning rate, number of iterations, number of training and testing samples, batch size etc., one could come up with a good model. An ensemble model can be designed for more accurate results by training the same network on different parameter settings. The network is trained with Backpropagation and gradient descent for weight updation. Training through Backpropagation is affected by a key issue- the diminishing gradient flow (also called the long time lag or vanishing gradient problem) [35], which makes it harder to train the lower layers of a multi-layer neural network since it either decays the error rate or explodes exponentially to the lower layers. This issue can be handled by appropriately by selecting a sparse activation function like sigmoid or tanh. But recently rectified linear units (ReLU) have been proved more effective for ConvNets due to its sparsity property and they suffer less from diminishing gradient flow[36].

The final layer of a convnet is the classifier. Commonly used one is the softmax classifier which output probabilities for each class, adding upto 1. But according to the need, various other classifiers like SVM, Probabilistic classifiers etc. are also used. SVM is used for structured prediction. Ensembles of classifiers are also widely used.

V. COMMON CNN ARCHITECTURES

A. LeNet

This was one of the pioneering work in Convolutional Neural Networks by LeCun et al. In 1990 [37] and later improved it in 1998 [38]. In their work, the task of

Handwritten Digit Recognition was performed using ConvNets. It finds application in reading zip codes, digits, etc. The lack of high computing machines at that time led to a break in the use of CNN.

B. AlexNet

This architecture developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton [39] is credited as the first work in Convolutional Networks to popularize it in the field of computer vision. The network was similar to LeNet but instead of alternating conv layers and pooling layers, AlexNet had all the conv layers stacked together. And compared to LeNet, this network is much bigger and deeper. AlexNet was able to win the ILSVRC-2012 competitions achieving top-1 and top-5 error rates on test dataset.

C. AlexNet

Matthew D. Zeiler and Rob Fergus have introduced a novel way to visualize the activity within a CNN using a multi-layered Deconvolutional Network (DeConvNet) [40]. The evolution of features during training can be tracked with this and also troubleshoot the network in case of any issues. They have used these tools to analyze the components of AlexNet and did a tweaking of the network by reducing the filter size and stride in first layer and expanded the size of the middle convolutional layers. Thus it became an improvement over AlexNet and was the ILSVRC 2013 winner.

D. GoogleNet

This ConvNet architecture from Szegedy et al.[41] from Google won the ILSVRC 2014 competition. They have proposed a new architecture called Inception (v1) that gives more utilization of the computing resources in the network. GoogLeNet is a particular incarnation that has twenty-two layers of Inception module but with less parameter compared to AlexNet. Later, many improvements had been done on Inception-v1, the major being introduction of batch normalization which led to Inception-v2 by Ioffe et al. [42]. More refinements were added to this version and the architecture was referred Inception-v3 [43].

E. VGGNet

Karen Simonyan and Andrew Zisserman have done a thorough analysis of the depth factor in a ConvNet, keeping all other parameters fixed. This try could have led to huge number of parameters in the network but it was efficiently controlled by using very small 3x3 convolution filters in all layers. This study has led to the development of a more accurate ConvNet architecture called, VGGNet. It was the runner-up in ILSVRC 2014 contest.

F. Inception V4

Szegedy et al., later in 2015 proposed an architecture [44], which is an improvement over GoogLeNet where the training of Inception modules (by Szegedy et al.[41]) are accelerated when trained with residual connections (introduced by He et al. [21]). The network has yielded state-of-the-art performance in the 2015 ILSVRC challenge and has won the contest.

G. ResNet

Kaiming et. al. [45] have presented a residual learning framework where the layers learn residual functions with respect to the inputs received instead of learning unreferenced functions. They were able to prove that this work is particularly useful for training deeper networks since residual networks are easier to optimize and gain much accuracy. The main drawback of this network is that it is much expensive to evaluate due to the huge number of parameters. But the number of parameters can be reduced to an extend by removing the first Fully-Connected layer (since most of the parameters are due to this layer), without any effect on the performance.

VI. Implementation

Many frameworks are available for deep learning, of which Google's TensorFlow is the latest and fast growing. It is an open source software library for doing high computational jobs using data flow graphs where edges denote tensors. With this, a single API can be used to distribute load between multiple nodes (CPUs or GPUs). This library is publicly available since November, 2015. Keras is considered to be second fast growing deep learning framework. This open source library written in Python is capable of running on top of TensorFlow or Theano. Theano is an open source Python library for numerical computations and simplifies the process of writing deep learning models. Another framework is Caffe, developed by the Berkeley Vision and Learning Center (BVLC). It has many worked examples of deep learning, written in Python. Giving more importance to GPUs is the framework called Torch, having an underlying C/CUDA implementation. Matlab's matconvnet and Torch's torch are also widely used frameworks for deep learning.

VII. Open Issues

Some of the open issues in the area of Convolutional Neural Networks are discussed here but most problems are dealt in many ongoing works. One of the major issues is that training CNNs require tuning of a large number of parameters leading to trial and error of the model architecture. Snoek et al. [46] has suggested that some automated tuning process is crucial for practitioners. Often multiple models need to be trained to finally get a sub-optimal one and this process can be both time consuming and impractical for large networks.

Work by Nguyen et al. in 2015, shows that most of the successfully trained ConvNets give high confidence result for objects that are unrecognizable to human-vision. Szegedy et al. [48] showed in a related work that when a gradual change is made in the image, the CNN results in an entirely different classification label. Goodfellow et al. [49] tried to reduce such variations, but to completely eliminate the issue is still open.

All the CNNs trained till-date are on one-shot. Some amount of research is still open for online training of CNN. A detailed analysis is done by Canziani et al. [50] paves way for a wider research.

VIII. Conclusion

Although CNNs are successful in its application areas, there is no theoretical proof to claim why it performs so well. In this work, we have reviewed the use of convolutional neural networks for computer vision. The entire spectrum of CNN may not captured in this paper despite our best efforts towards it. We hope that our article will be useful to vision researchers beginning to work on convolutional neural network.

References

[1] Ramachandran R, Rajeev DC, Krishnan SG, P Subathra, Deep learning an overview, IJAER, Volume 10, Issue 10, 2015, Pages 25433-25448.

[2] D. H. Hubel and T. N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, The Journal of physiology, 1968.

[3] J. Fan, W. Xu, Y. Wu, and Y. Gong, Human tracking using convolutional neural networks, Neural Networks, IEEE Transactions, 2010.

[4] A. Toshev and C. Szegedy, Deep -pose: Human pose estimation via deepneural networks, in CVPR, 2014.

[5] M. Jaderberg, A. Vedaldi, and A. Zisserman, Deep features for text spotting, in ECCV, 2014.

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

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

[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, Learning hierarchical features for scene labeling, PAMI, 2013.

[9] Nithin, D Kanishka and Sivakumar, P Bagavathi, Generic Feature Learning in Computer Vision, Elsevier, Vol.58, Pages202-209, 2015.

[10] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological cybernetics, 1980.

[11] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hub-bard,and L. D. Jackel, Handwritten digit recognition with a back-propagation network, in NIPS. Citeseer, 1990.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learningapplied to document recognition, Proceedings of the IEEE, 1998.

[13] Alex Krizhevsky, Sutskever I, and Hinton G.E, Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[14] A. Berg, J. Deng, and L. Fei-Fei, Large scale visual recognition challenge 2010, www.image-net.org/challenges. 2010.

[15] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In International Conference on Computer Vision, pages 21462153. IEEE, 2009.

[16] A. Krizhevsky. Convolutional deep belief networks on cifar-10. Unpublished manuscript, 2010.

[17] B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hub- bard, and L. D. Jackel, Handwritten digit recognition with a backpropagation network, in NIPS. Citeseer, 1990.

[18] Zeiler M., Taylor G., and Fergus R. Adaptive deconvolutional networks for mid and high level feature learning, In ICCV, 2011.

[19] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,2001.

[20] A. Krizhevsky. Convolutional deep belief networks on cifar-10. Unpublished manuscript, 2010.

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

[22] Matthew D. Zeiler, Rob Fergus, Visualizing and Understanding Convolutional Networks, 13th European Conference, Zurich, Switzerland, 2014, Proceedings, Part I, Volume 8689, pages:818-833.

[23] Fisher Yu and Vladlen Koltun, Multi-Scale Context Aggregation by Dilated Convolutions, ICLR,2016

[24] M. D. Zeiler and R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, CoRR, 2013.

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

[26] Nguyen A, Yosinski J, Clune J, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

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

[28] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, Striving For Simplicity : The All Convolutional Net, ICLR, 2015.

[29] Matthew D. Zeiler, Rob Fergus, Visualizing and Understanding Convolutional Networks, 13th European Conference, Zurich, Switzerland,2014, Proceedings, Part I, Volume 8689, pages:818-833.

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

[31] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, Regularization of neural networks using dropconnect, in ICML, 2013.

[32] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, The loss surfaces of multilayer networks, arXiv, 2014.

[33] D. Mishkin and J. Matas, All you need is a good init, arXiv preprint rXiv:1511.06422, 2015.

[34] I Sutskever, J. Martens, G. Dahl, and G. Hinton, On the importance Of initialization and momentum in deep learning, ICML, 2013.

[35] S. Hochreiter, Y. Bengio, P. Frasconi, Gradient flow inrecurrent nets: the difficulty of learning long-term dependencies. 2001.

[36] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks, In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W, 2011.

[37] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hub- bard, and L. D. Jackel, Handwritten digit recognition with a back- propagation network, in NIPS, 1990.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learningapplied to document recognition, Proceedings of the IEEE, 1998.

[39] Alex Krizhevsky, Sutskever I, and Hinton G.E. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[40] Zeiler M., Taylor G., and Fergus R. Adaptive deconvolutional networks or mid and high level feature learning, In ICCV, 2011.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, Going deeper with convolutions, CoRR, 2014.

[42] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of The 32nd International Conference on Machine Learning, 2015.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Re-thinking the inception architecture for computer vision. arXiv:1512, 2015.

[44] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv:1602.07261, 2016.

[45] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, arXiv preprint arXiv:1512.03385, 2015.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition,arXiv preprint arXiv:1512.00567, 2015.

[47] Snoek J, Larochelle H and Adams R P,Practical bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems 25, pages 29512959, Curran Associates, Inc.

[48] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I and Fergus R,Intriguing properties of neural networks, arXiv, 2013.

[49] Goodfellow I J, Shlens J and Szegedy C, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572, 2014.

[50] Canziani, Alfredo and Paszke, Adam and Culurciello, An Analysis of Deep Neural Network Models for Practical Applications, 2016.