LOCALLY CONNECTED NEURAL NETWORKS FOR IMAGE RECOGNITION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Shakti N. Wadekar

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF THESIS APPROVAL

ACKNOWLEDGMENTS

I am thankful to all my committee members, Prof. Kaushik Roy, Prof. Anand Raghunathan and Prof. Vijay Raghunathan for being supportive throughout the my Master's degree. I specially thank Prof Kaushik Roy for his crucial guidance during this master's thesis research work. I am grateful to all my colleagues in lab for extending there help whenever needed and my family members who were always there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

LCN	Locally Connected Network
P-LCN	Partially-Local Connected Network
CNN	Convolutional Neural Network
VC Dimension	Vapnik–Chervonenkis Dimension
BN	Batch Normalization
LN	Ladder Network
SSL	Semi-Supervised Learning

ABSTRACT

Wadekar, Shakti N. Master's, Purdue University, December 2019. Locally Connected Neural Networks for Image Recognition. Major Professor: Kaushik Roy.

Weight-sharing property in convolutional neural network (CNN) is useful in reducing number of parameters in the network and also introduces regularization effect which helps to gain high performance. Non-weight-shared convolutional neural networks also known as Locally connected networks (LCNs) has potential to learn more in each layer due to large number of parameters without increasing number of inference computations as compared to CNNs. This work explores the idea of where Locally connected layers can be used to gain performance benefits in terms of accuracy and computations, what are the challenges in training the locally connected networks and what are the techniques that should be introduced in order to train this network and achieve high performance. Partially-local connected network (P-LCN) VGG-16 which is hybrid of convolutional layers and Locally connected layers achieves on average 2.0% accuracy gain over VGG-16 full convolutional network on CIFAR100 and 0.32% on CIFAR10. Modified implementation of batch normalization for Full LCNs (all layers in network are locally connected layers) gives improvement of 50% in training accuracy as compared to using CNN batch normalization layer in full LCN. Since L1, L2 and Dropout regularization does not help improve accuracy of LCNs, regularization methods which focuses on kernels rather than individual weight for regularizing the network were explored. Ladder networks with semi-supervised learning achieves this goal. Training methodology of ladder networks was modified to achieve $\sim$2% accuracy improvement on Pavia-University hyper-spectral image dataset with 5 labels per class.

1. INTRODUCTION

Convolutional neural networks (CNNs) [1] have surpassed human level accuracy [2] on image recognition datasets like imagenet [3]. But these networks perform poorly when the same image is scaled, rotated or translated by a large amount, but a human vision can still identify an object even with these variations. So CNNs have not truly surpassed the ability of human vision system. The local receptive fields and weight sharing of kernels in CNNs embeds the prior knowledge or assumption that local features are translational invariant, these assumptions may be true in the initial layers but hard to justify in the deeper layers. Weight sharing indeed reduces number of parameters, but it also limits the learning capacity of a layer due to fewer parameters. A solution to less number of parameters is to increase number of output maps in each layer, but it increases number of computations during training and inference. Non-weight-sharing convolutional neural networks called as Locally Connected Neural Networks (LCNs) provides an alternative solution to this parameter and computations problem. *So if we keep the number of output maps of each layer same and only remove weight-sharing constraint in CNNs, number of parameters in a layer increases without increasing number of output maps, hence capacity of the network is now increased without increase in number of computations during inference as compared to weight-shared CNN.* So now question arises that what happens to learning in the network when we do not share weights? Does the high capacity network due to increased number of parameters because of not sharing weights help to obtain high accuracy? If not why? and if we do not obtain high accuracy, how do we change the training methodologies to gain high accuracy from this high capacity network?. These are the questions this work tries to address. This work explores Locally Connected Neural Network (LCN) a non-weight-sharing version of convolutional neural network.

Prior works which explored locally connected networks best to my knowledge are *Tiled Convolutional Neural Networks* [4] (T-CNN), Facebook's *DeepFace* network [5] and Google's speaker recognition network [6].

T-CNN also called as diluted convolutional network since it shares kernel weights after a certain number of strides. This paper argues that, CNNs have hard-coded translational invariance due to weight sharing and pooling over outputs belonging to same kernels, so in order to allow the network to learn more complex invariances, pooling over different kernels should be done and variety of different kernels can be learned by not sharing the weights after each stride. This paper evaluated CNN and T-CNN on NORB dataset [7] which contains images of 5 categories which are taken at 18 different azimuths, 9 different elevations and 6 different lightning conditions. T-CNN achieved ∼2% higher accuracy than CNN. Also the kernels learned by the network were scale and rotationally invariant. Therefore allowing the network to learn different weight at different location and pooling over these different kernel outputs allowed network to capture the invariances in the dataset which weight shared CNN could not capture.

Table 1.1.
T-CNN and CNN comparison on NORB dataset

#	**CNN**	**Deep T-LCN**
NORB	94.4%	**96.1%**

DeepFace network is a combination of convolutional and locally connected layers used for face recognition. The first two layers are convolutional, next three layers are locally connected layers and last two Fully connected layers forms a classifier. This paper states that, since local statistics of face is important for face recognition and locally connected layers helps to capture this local statistics by learning kernels which are independent of the other kernels learned in different locations, this DeepFace architecture achieves high performance in terms of accuracy on face recog-

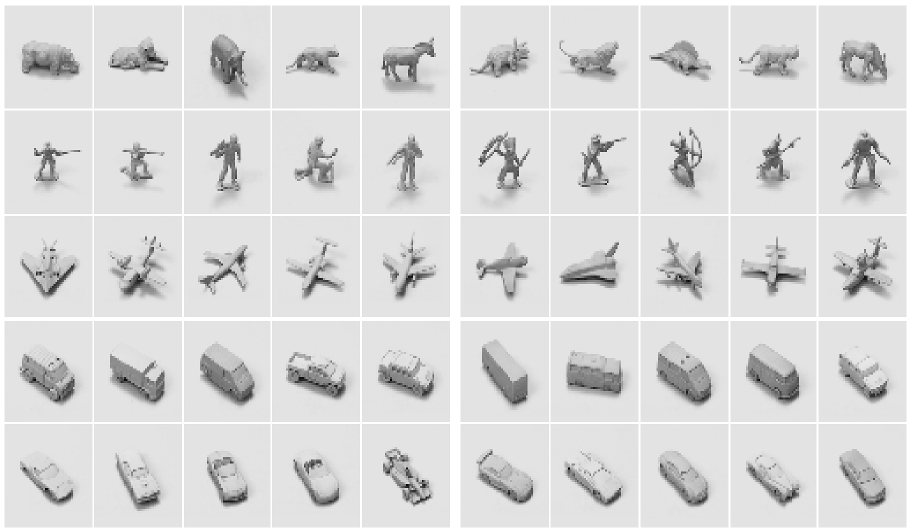


Fig. 1.1. NORB dataset

nition. DeepFace achieved state of the art accuracy of 97.35% on Labeled Faces in the Wild (LFW) [8] dataset. These locally connected layers when used instead of convolutional layers helped improve the accuracy on Labeled Faces in the Wild (LFW) dataset, therefore it indicates that a locally connected layer has potential to capture more than a convolutional layer.

Google's speaker recognition system (2015) used locally connected layer as its first layer. The system observed an decrease in error rate to 3.60% as compared to the fully connected network which had 3.88% on the Text-Dependent Speaker Verification (TD-SV) task. CNN which had comparable number of parameters as the fully connected network with first layer as LCN, achieved 3.52% accuracy, but the number of output maps in each layer were more, therefore number of computations (multiplications) were significantly higher. Hence performance gain was achieved without increasing number of computations in the network by using locally connected layer.

So we explore Locally Connected Neural Network (LCN) and individual locally connected layers in different architectures which will try to benefit from the non-weight sharing property. Initially effect of individual locally connected layers on the network is explored. A hybrid architecture of convolutional layers and locally connected layers together called as Partially-local connected network (P-LCN) is studied on CIFAR100 [9] and CIFAR10 [9] datasets. *A VGG-16 architecture of P-LCN which has last 6 feature layers as locally connected layers and rest feature layers as convolutional achieved an improvement of ∼2.0% and ∼0.32% on CIFAR100 and CIFAR10 respectively.* Next, VGG-9 architecture of a full LCN (all layers are locally connected layers) was trained to observe that the training accuracy saturated close to 40%. The reason was batch-normalization layer. Convolutional batch-normalization layer is not compatible for batch-normalizing a locally connected layer. To train a Full-LCN (all the layers in the network are locally connected layers), batch-normalization layer implementation has to be changed. *In 300 epochs the modified batch-normalization implementation reaches 90% accuracy as opposed to normal implementation which saturates at 40%, hence 50% training accuracy improvement.* The conventional regularization techniques such as L1, L2 regularization and Dropout regularization did not help to improve the testing accuracy of LCN. Testing accuracy for Full LCN is ∼10% less than its CNN counterpart on CIFAR100 even after we achieve the training performance improvement by modifying batch normalization layer. There is a need of a new or unconventional regularization technique for locally connected layers. Semi-supervised learning and Multitask learning serves as regularizers which focuses on kernels as whole as opposed to L1 and L2 which focuses on individual weights for regularizing the network. Semi-supervised learning on Ladder Network (LN) was explored in search of regularization for LCNs. *State of the art performance on Pavia-University hyper-spectral image dataset was achieved by improving training methodology of ladder network.* Semi-supervised learning and Multitask learning has potential to be effective methods to regularize networks with large parameters such as LCNs. This work can be extended further by evaluating the semi-supervised and multi-task learning more and developing methodologies to effectively apply it to LCNs.

2. LOCALLY CONNECTED NEURAL NETWORKS

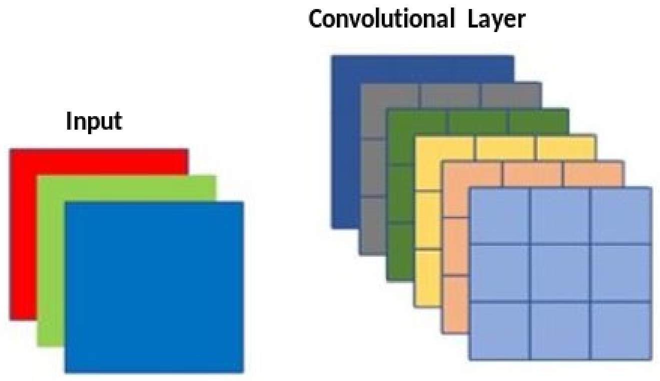
2.1 Locally connected neural networks

Nobel prize winning work done by Hubel and Weisel [10] showed that each neuron in primary visual cortex has a specific receptive field i.e, local receptive field and can detect certain specific orientations, also there exists hierarchy between simple and complex cells. This local receptive field motivated the convolutional neural networks, where each neuron in a layer is connected to only few neurons is the previous layer. The additional constraint was to share parameter i.e, weight sharing or kernel weight sharing in order to reduce the number of learnable parameters. This allowed the layer to use same kernel to be convolved over the entire image and multiple such kernels are used to learn the features from input images. In convolutional neural networks, due to weight sharing, huge reduction in learning parameters is possible, which is beneficial in terms of memory consumption.

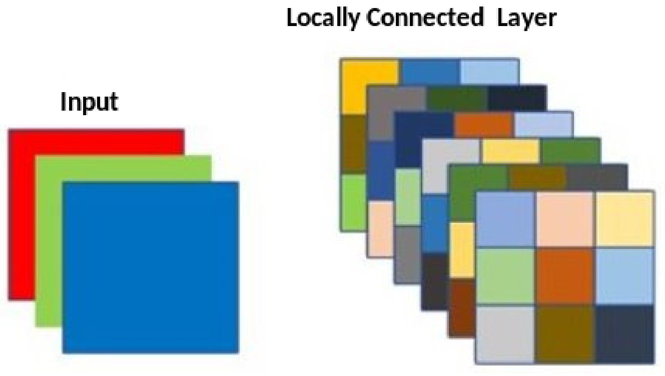
Weight sharing is not bio-plausible and this problem is know as weight transport problem [11]. Locally connected neural network (LCN) is convolutional neural network with weight sharing constraint removed. Hence, locally connected neural networks are more bio-plausible as compared to convolutional neural network. The work here explores how these networks can be trained, where and how locally connected layers can be useful to improve performance and what are the trade-offs.

Figure 2.1 shows the difference in the output activation maps between convolutional and locally connected layer in a and b respectively. Each output activation map generated in CNNs belongs to one kernel, while in LCN layer each output activation is generated by different kernel.

The input in figure 2.1 has 3 channels and number of output channels or maps are 6. In CNNs, each output channel is generated by a kernel of size *[(kernel-height)*



(a) Convolutional Neural Network



(b) Locally Connected Neural Network

Fig. 2.1. Input is an coloured image with 3 channels (Red, Green and Blue). Figure *a* shows the output activations map when we convolve kernel over the entire image. Convolutional layer shows 6 maps which are generated by 6 different kernels. Figure *b* shows output activations generated in locally connected layer. Each map has 9 activations and there are 6 such maps, therefore 54 activations generated from all different kernels.

** (kernel-width) * (input-channels)]* and since we have 6 output maps, we have in total 6 such kernels. Each kernel is convolved over the entire image with certain number of strides which can overlap or do not overlap with previous stride. In LCNs, each output activation in each map is generated by different kernel. The figure has 9 outputs in each map, therefore total number of output activations are 54. Number of kernels in LCNs are equal to number of output activations, hence number of kernels required are 54. So each kernel size is same as the kernel size of CNNs, but the number of kernels increases and in this case its 54 and number of parameters are *[(kernel-height) * (kernel-width) * (input-channels) * (Number-of-activations-in-each-map) * (Number-of-maps)].*

2.2 Partially-Local connected networks

When we use both convolutional layers and locally connected layers in a hybrid manner in the network, we call these networks as Partially-Local Connected Networks (P-LCN). Each convolutional neural network is made up of two parts, first is a feature extractor and second is classifier. Feature extractor is made up of stacked convolutional layers. In the case of P-LCN, its feature extractor consists of stack of convolutional and locally connected layers. The locally connected layers are connected after a stack of convolutional layers and the classifier is connected after the locally connected layers. Classifier layer consists of multiple fully connected layers and a softmax layer.

The inspirations for P-LCN network architecture comes from DeepFace network architecture. Facebook's DeepFace network for face recognition has last three convolutional layers replaced with locally connected layers, which help the network to gain improvement in accuracy on LFW dataset. The paper argues that different regions of face images has different local statistics [5]. Eye region has very different appearance than the eyebrow region. Hence in order to capture these different local statistics, locally connected layers are better because by not sharing weights we are allowing features to be combined locally and independent of the other location. In contrast, CNNs assume that features present in an location are also present through out the image. This assumption may not be true in the deeper layers as seen experimentally with this network. Therefore locally connected layers help to improve the accuracy of DeepFace network.

The same argument can taken forward to the other image recognition tasks. Datasets such has CIFAR-10, CIFAR-100 and MNIST has single objects or digits in it, therefore as we allow local features to be combined independently of the other location due to non-weight-sharing in locally connected layers connected in deeper layers of the network, then it might help us learn or capture the details which convolutional layer might not capture.

So now we explore P-LCN with VGG-16 architecture on CIFAR100 and CIFAR10 dataset.

2.2.1 Network

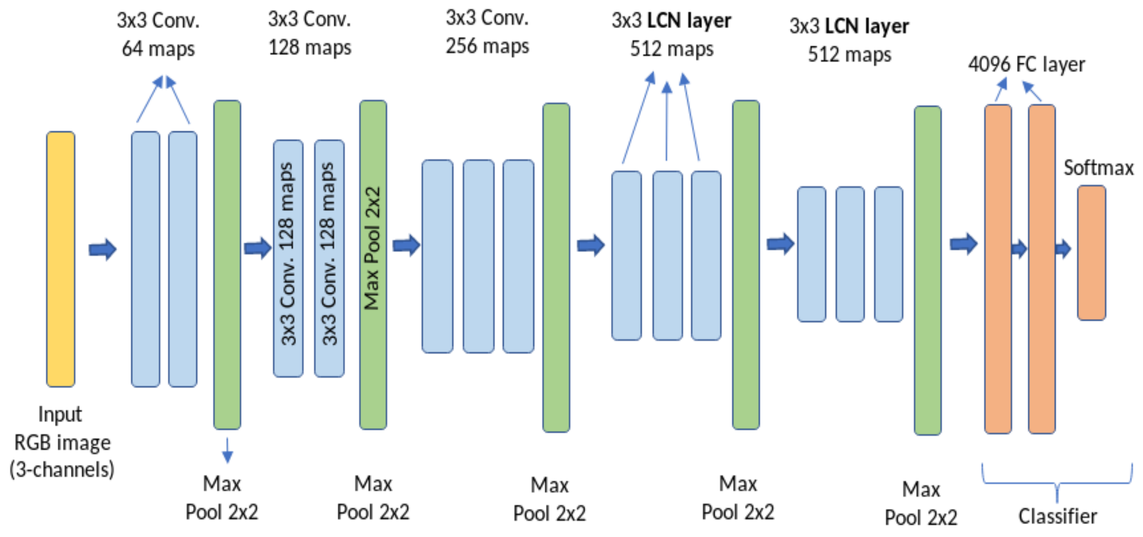


Fig. 2.2. Vgg-16 P-LCN architecture. Convolutional layers (Conv.) and LCN layers act as feature extractor part of architecture and at the end we have classifier to classify the input RGB image. First 7 layers are convolutional layers and next 6 are Locally Connected layers (LCN layers). Classifier consists of 2 fully connected layers with 4096 neurons in each and last layer is the output softmax layer.

Vgg-16 architecture is used with replacing last 6 convolutional layers with locally connected layers. Each locally connected layer has 512 maps and every activation in each map belongs to different kernel. A convolutional VGG-16 architecture has ∼15 million parameters and P-LCN VGG-16 architecture has ∼124 million parameters i.e, ∼8x more number of parameters. More details about the input layer, data normalization, non-linear activation function, initialization, training algorithm and data augmentation is detailed in 2.2.4 section.

2.2.2 P-LCN results on CIFAR-100 and CIFAR-10

Max accuracy improvement by 2.32% was achieved on CIFAR-100 by the P-LCN Vgg-16 architecture as compared to the Vgg-16 CNN baseline. On average ∼2% accuray improvement is observed with different initialization.

Table 2.1.
Accuracy table of CNN and P-LCN with VGG-16 architecture on CIFAR100 and CIFAR10

#	**Vgg-16 CNN**	**Vgg-16 P-LCN**
cifar100	65.21%	**67.53%**
cifar10	91.03%	**91.32%**

Here P-LCN has ∼8x more number of parameters than the CNN, but number of computations during inference remains same. The reason why computations does not increases is when a convolutional layer is changed to Locally connected layer, due to non-weight-sharing, only parameters in each layer are increased and not the number of output channels, therefore number of computations remains same. So we achieve accuracy improvement by ∼2% without increasing number of computations during inference with the trade-off of increases number of parameters.

It can be argued that accuracy can be improved on the CNN by increasing number of kernels in each layer or few layers. But, for that accuracy improvement we are now increasing number of parameters and number of output maps which increases number of computations during inference. Hence, for the accuracy gain we trade-off number of parameters and number of computations in CNNs, but P-LCN provides accuracy

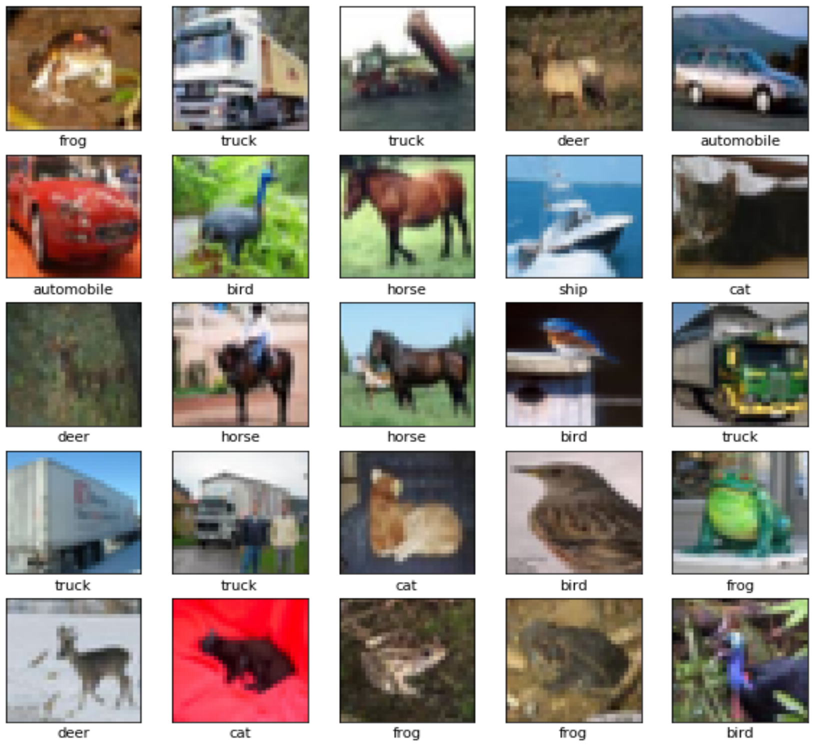


Fig. 2.3. CIFAR10 Images.

gain by keeping the number of computations during inference same and only trade-off with increases in number of parameters.

There are variations in floating point computations with different gpus and different versions of drivers of the same gpu. The results here vary when the this architecture is run on different gpu versions. So reproducibility is a challenge as is the case in overall deep learning networks. Also other possible reason would be related to where the optimization functions has found it optimum parameter values. If the minima is too narrow, then the results even due to small changes can be significantly different and specially it more likely here due the high number of parameters in the network due to locally connected layers.

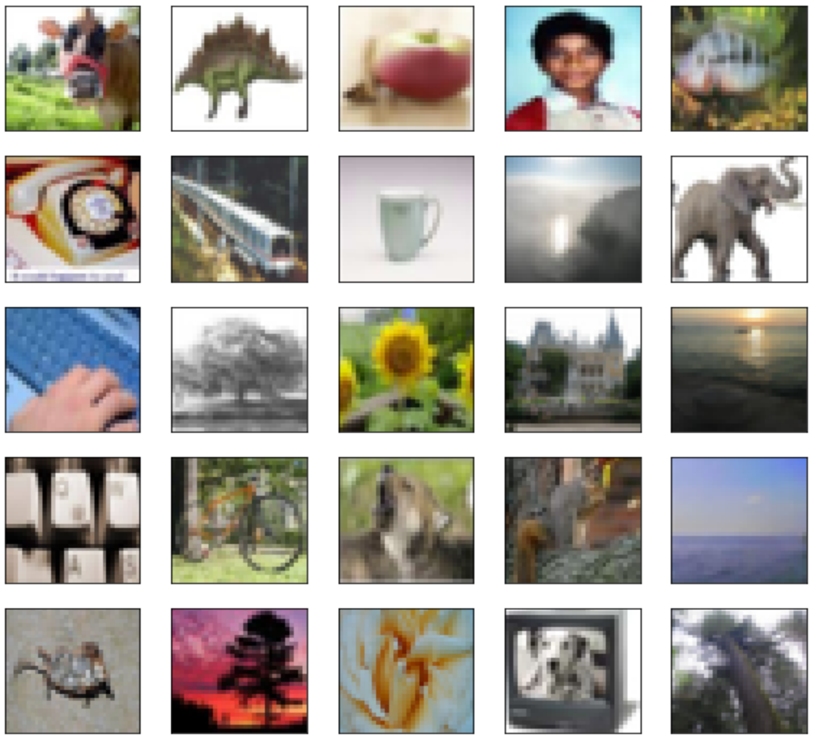


Fig. 2.4. CIFAR100 Images.

2.2.3 Discussion on improved result

The central question is why do we get the accuracy gain?. The answer lies in the increase of capacity of the network. As the number of parameters in the network has increased, it allows the network layer to learn more complex function or decision boundaries i.e, hypothesis space of that layer has increased. Hence this new increased hypothesis space provides ability to that layer to explore or learn a function that is able to classify data more accurately than its counterpart convolutional layer which has less parameters given the constraint that number of computations during inference does not increase.

VC dimension is a measure of capacity of a network or classifier i.e, how large is the hypothesis space of the network. The number of parameters can be related to VC dimension. Hypothesis space is number of different functions the network can learn.

So larger the hypothesis space, larger is the capacity of network. So, as the number of parameters increases, the network can learn more complex functions, therefore hypothesis space of the network has increased i.e, capacity of network has increased, hence VC dimension of network increases.

2.2.4 Network architecture details:

Input layer: 3-channel RGB images of 32x32x3 pixels each. CIFAR100 and CIFAR10 has 50000 training images and 10000 test images of 32x32x3 pixels each. CIFAR100 has images with 100 classes and CIFAR10 has images with 10 classes in it. All the input images are normalized using the following equation,

$$x_{normalized} = \frac{x_{data} - \mu}{\sigma + 0.0000001} \tag{2.1}$$

μ here is the mean of all input pixel values of the input images and σ is the standard deviation. A constant is added in the denominator to avoid division by zero.

Convolutional layer: This is the layer where the convolution operation with weight sharing occurs. The kernel with specific size is convolved over the entire image with a fixed stride to produce the convolution output. A non-linear activation function is applied on the convolution output. Here ReLU activation [2] function is used as non-linear activation.

Locally connected layer: local convolution without weight sharing happens in this layer. Each activation in this layer is generated by different kernel. Hence, number of activations is equal to number of kernels learnt in this layer. This layer also uses a non-linear activation function. In the experiments ReLU is used as non-linear activation function for this layer.

Batch-normalization layer: The output activations of each convolutional and locally connected layer is batch-normalized i.e, values are scaled down mostly between -1 to 1. This is mainly done to reduce the internal co-variance shift [12] of activa-

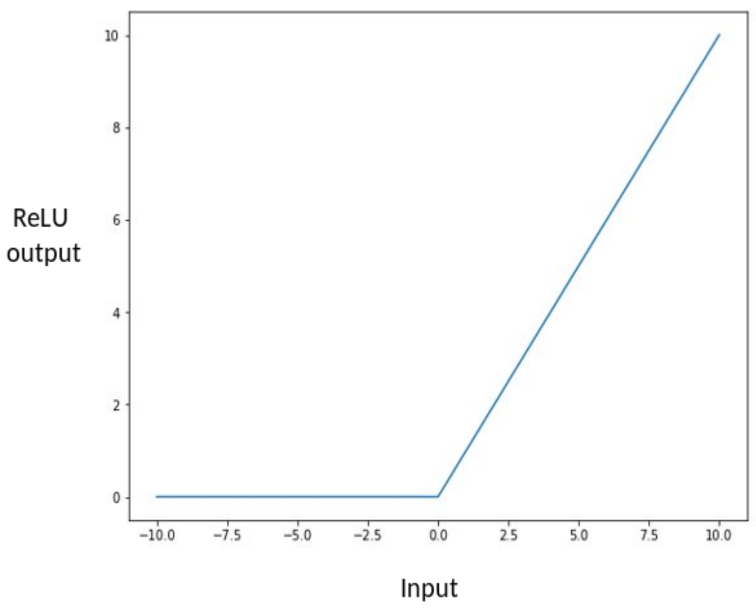


Fig. 2.5. ReLU activation function.

tions of each layer before these activations are given as input to the next layer. This technique accelerates training as it allows higher learning rate in the initial epochs of training. The batch-normalization layer used in this architecture is same for convolutional and locally connected layers, but we will see in the coming sections that the batch-normalization implementation has to be changed for the locally connected layers. Also, we will see that convolutional batch-normalization layer is in some cases is useful when used for locally connected layer.

Pooling layer: Max-pooling layer is used in this architecture which has size of 2x2 and does stride of (2,2). Therefore there is no overlap with previous units which were used for pooling.

L2 regularizer: Convolutional layers here uses L2 regularization [13] but the locally connected layers do not. Testing accuracy of this system decreases when L2 regularization is used in locally connected layers.

$$C_{total} = C_{CrossEntropy} + \lambda * C_{regularizer} \tag{2.2}$$

$$C_{total} = C_{CrossEntropy} + \lambda * \frac{1}{2} * \|w\|^2 \tag{2.3}$$

L2 regularizer is a constraint on the cost or loss function. This regularization pushes most of the weights of in the network towards zero, but not completely make them zero as it is in the case when L1 regularization is applied. λ is the scaling factor which determines the amount of contribution of L2 regularization cost ($C_{regularizer}$) to the major or main cost function.

Dropout layer: Dropout layer [14] is used to regularize the network along with L2 regularization. The output activations in each layer are dropped randomly during each forward pass during the entire training. Every time a dropout mask is applied to the network, different structure of the network is trained. Hence, at the end of training we have an ensemble of different network structure embedded into one neural network structure. This creates the regularization effect which helps network to generalize better and have high testing accuracy.

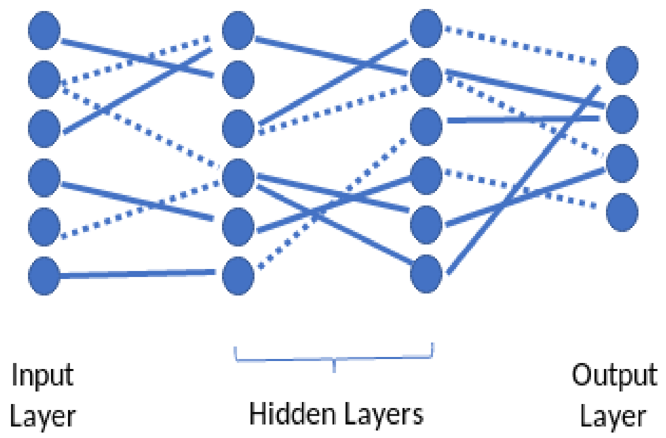


Fig. 2.6. Fully Connected Neural Network with Dropout. The dashed lines or connections between the neurons are connections which are masked when dropout is used. Different connections are dropped at random during the complete training procedure. This figure is an example of how the network looks when few connections are masked. The connections or weights which are masked are not updated during back-propagation since they did not contribute to the activation of neuron in the forward pass.

Softmax layer:

$$R(y_i) = \frac{e^{y_i}}{\sum_i e^{y_i}} \tag{2.4}$$

Cross-entropy Loss: Multi-class cross-entropy loss is given by equation 2.5

$$C_{CrossEntropy} = -\sum_{k=1}^{C} y_{(l,k)} log(R_{(l,k)}) \tag{2.5}$$

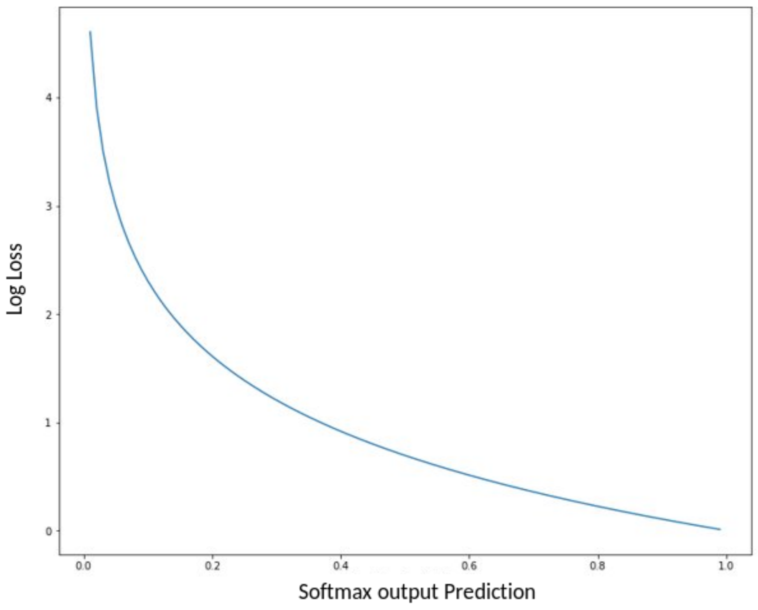


Fig. 2.7. Log loss function

C is total number of classes and also equal to number of output units of last layer in classifier. y is the label of l^{th} output for the k^{th} class of the input image. $y_{(l,k)}$ is generally 1 or 0 if the output is one-hot coded. R is the prediction generated by l^{th} neuron in the last layer i.e, softmax layer.

Figure 2.7 shows how the log-loss changes with the prediction (output of softmax unit). If the prediction is accurate i.e, close to 1 then the loss is minimum. Loss grows exponentially higher as the prediction diverges from the actual label.

Training algorithm: Stochastic gradient descent algorithm with nestrov momentum [15] is used with Back-propagation algorithm to train this deep neural network.

$$v_{new} = \alpha v_{old} - \eta * \nabla_\theta L(\theta + \alpha v_{old}) \tag{2.6}$$

$$\theta_{new} = \theta_{old} + v_{new} \tag{2.7}$$

η and α are hyperparameters. v_{new} an v_{old} are new and old velocities respectively and θ is network parameter or weight. The P-LCN network is trained for 700 epochs with data augmentation.

2.3 Full-Locally connected networks

Further we explore full-locally connected network i.e, all layers in the network are locally connected layers. They are referred to as LCNs. LCN of VGG-9 architecture is used throughout the following experiments for training and testing the performance of LCNs on CIFAR100 and CIFAR10. During training the LCNs, it is observed that batch-normalization layer of CNN is not suitable as batch-normalization layer of LCNs. The CNN batch-normalization layer introduces error in the network, which then does not allow the training accuracy to increases above 40%. Following section discusses how the CNN batch-normalization layer introduces error and what is alternate implementation of batch-normalization layer for LCNs.

2.3.1 Batch Normalization for Locally connected networks

Now as all the network layers are changed from convolutional layers to locally connected layers, high training accuracy is not achieved while training LCNs, hence test accuracy suffers. The reason for this is the Batch Normalization layer. We will now see what problem this layer poses and how can we resolve it to improve training accuracy.

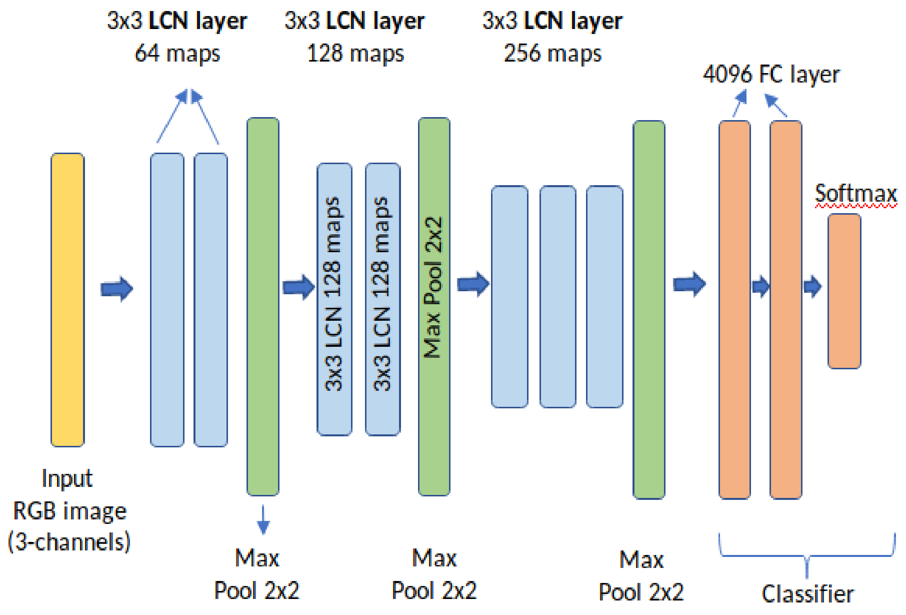


Fig. 2.8. Vgg-9 LCN architecture.

Problem:

Normalizing the activations of each layer [12],

$$\hat{x}^i = \frac{x^i - E[x^{i,k}]}{\sqrt{Var[x^{i,k}] + \epsilon}} \tag{2.8}$$

$E[x^{i,k}]$ is mean of all the activations generated by the specific kernel k in a layer l i.e, it is mean of output channel belonging to kernel k. $Var[x^{i,k}]$ is the variance in activations of that output channel and ϵ is a small positive value added to variance to avoid the denominator to become zero. Each output activation value x^i is normalized to obtain $\hat{x}^i$.

In Batch normalization of convolutional neural networks, normalization is done by grouping activation values which belong to same kernel i.e, each channel is normalized independently. Therefore the correlation between the output activation values in each channel is preserved even after scaling down values due to normalization.

In Locally connected network each and every output is obtained from different kernel. Therefore, when we normalize over these different kernel outputs within on

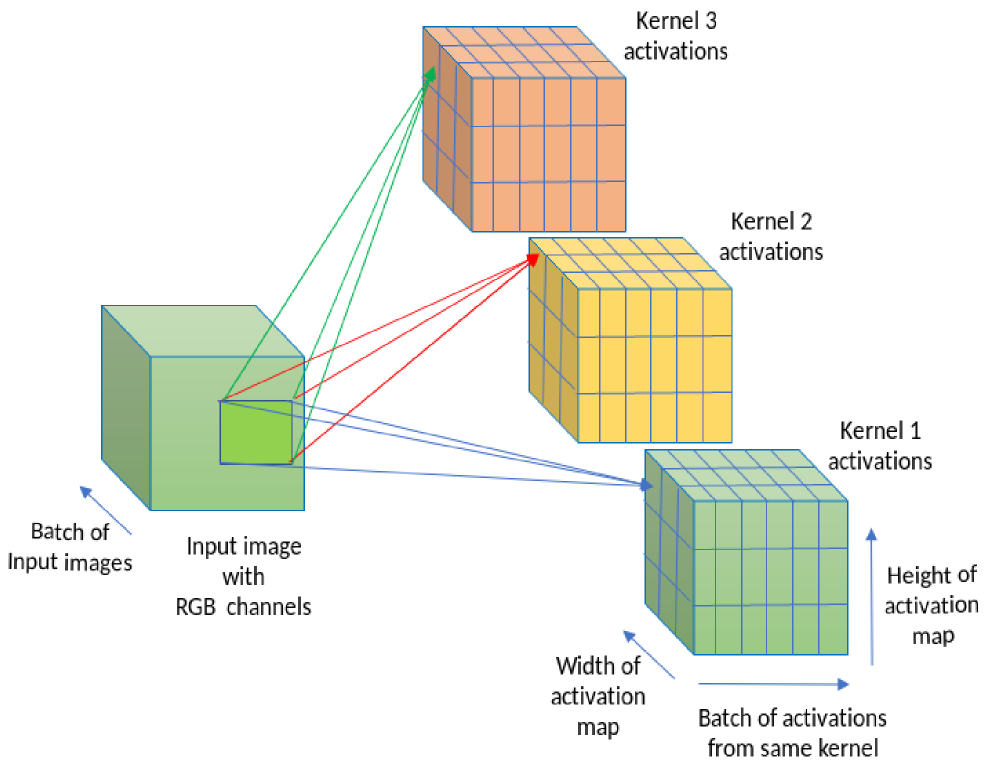


Fig. 2.9. Batch of input images is given to a convolutional layer. Each kernel represented by kernel 1: *blue*, 2: *red* and 3: *green* generates activations for that entire batch. All the activations generated in layer 1 by kernel 1 are represented in green, kernel 2 by yellow and kernel 3 by orange block. The batch normalization in CNN is done on each of these blocks *independently*, hence the output activation correlation is maintained.

channel, we are introducing noise or error by not preserving the correlation between outputs while scaling it down. So important outputs or activations may get scaled down to zero, hence crucial classification information is lost. Hence network, due to this normalization error, suffers from low training accuracy.

Solution:

The problem can be fixed by together normalizing the output activation values which belong to the same kernel within each channel.

$$\hat{x}^i = \frac{x^i - E[x^{i,k}]}{\sqrt{Var[x^{i,k}] + \epsilon}}$$

Now, kernels are specific to a location, so k is kernel of that specific location and x^i is the output activation generated by that kernel k. So we *normalize across the batch in a channel for a particular location.*

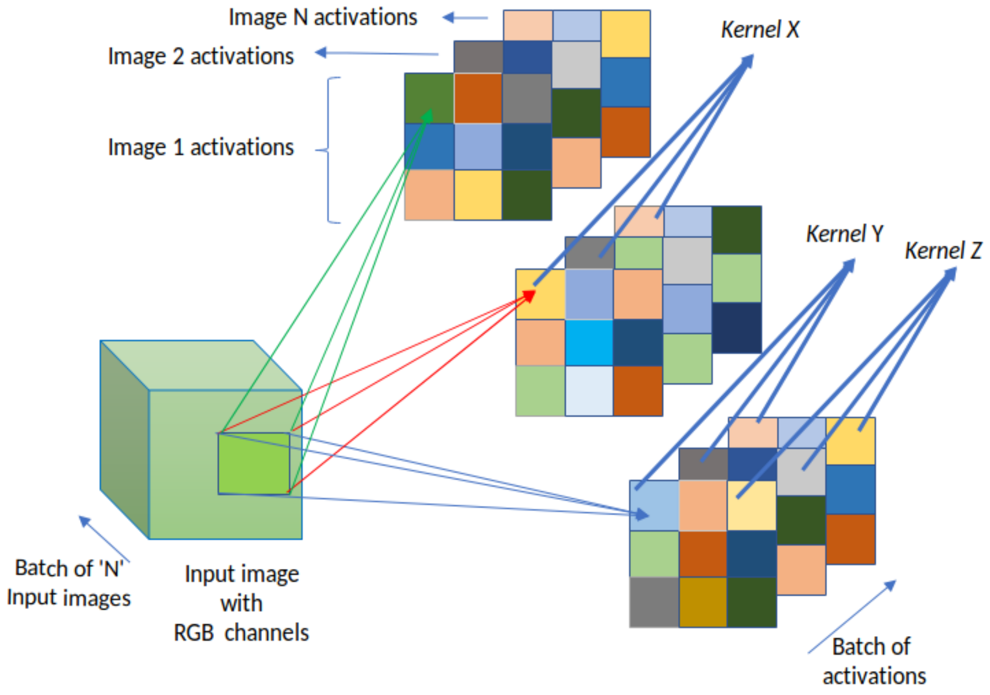


Fig. 2.10. Batch normalization for LCN layer. Output activations generated by $Kernel\ X, Y\ and\ Z$ belongs to a specific location in the image over batch of images. All the activation generated by kernel Z are taken together and normalized. So the normalization of output activations of $Kernel\ X$, $Kernel\ Y$ and $Kernel\ Z$ are independent from each other.

Specific location on image dimensions has same kernel over the batch of images and output activations of this kernel are related to each other, hence we normalize these values together. So output activations of each location over the batch of images are normalized together and activations from different kernels are normalized independently, hence the correlation between activations is preserved even after normalization. This solution works and we get the training accuracy improvement as shown in figure 2.7.

Improved results with this solution:

Figure 2.7 shows the training accuracy improvement by over 50% in 300 epochs of training on CIFAR-100 with the modified batch-normalization implementation. Now a full LCN is fully trained and we can test network on test images.

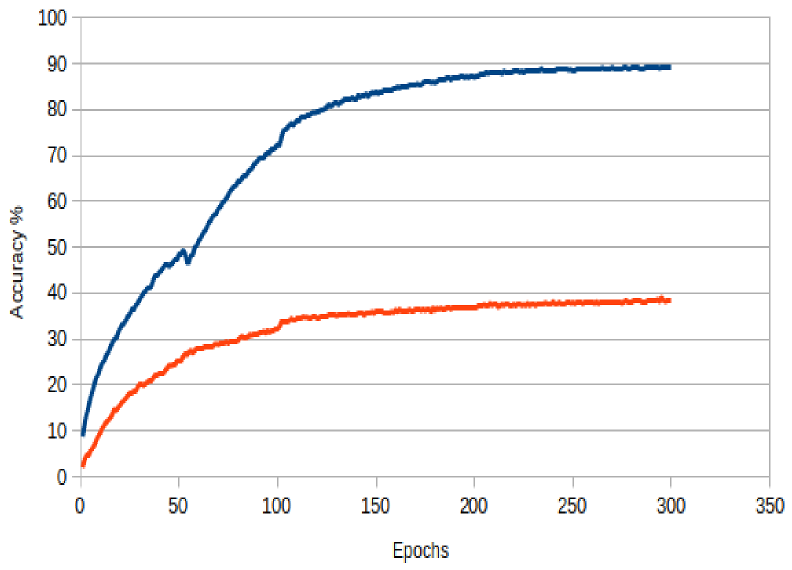


Fig. 2.11. Comparison of LCN training accuracy with *(blue)* and without *(orange)* modified batch normalization.

Since the batch-normalization layer was used in each layer, error is propagated further during forward propagation and amplified. The error introduced by batch-normalization layer is removed with the new implementation of batch-normalization for locally connected layers. We will see in the next section that how this error is small in last layers and this error can potentially act regularizer in the P-LCN.

2.4 Convolution BN layer as a regularizer in P-LCN:

The convolution batch normalization layer introduces error in the locally connected layer output. This error acts as noise for this layer. Since noise acts as a good regularizer in neural networks, the convolutional batch normalization layer is acting as a regularizer for P-LCN.

Now, as we replace all convolution layers with locally connected layers and use convolutional batch normalization layer with it, error is introduced in each layer and gets amplified as the activations are forward propagated down the network. This amplified error hinders the training of network and causes the low training accuracy. In the case of P-LCN the output dimensions of of LCN layers are small compared to the initial layers, hence the error due to batch-normalization layer is not significant as compared to the error that is generated in the first layers due to large dimensions. Therefore, this small error is acting as noise and noise regularizes the P-LCN network.

2.5 Regularization of Locally connected neural networks

Following table shows the performance of LCN (all layers are locally connected layers) on CIFAR10 and CIFAR100. The network performance degrades by 8% and 13%.

Table 2.2.
Accuracy table of CNN and LCN with VGG-9 architecture on CIFAR10 and CIFAR100

#	**Vgg-9 CNN**	**Vgg-9 LCN**
cifar10	78.51%	70.29%
cifar100	60.33%	47.17%

What locally connected neural networks lack is a good regularization technique. Due to large parameter space, the effective regularization techniques for deep neural *networks such as L1, L2 regularization and dropout doesn't provide performance benefit here. New regularization techniques suitable to LCN needs to be explored to achieve performance gain.*

In P-LCN, due to noise by batch normalization layer, a regularization effect was introduced and a good accuracy performance is achieved. But noise is not a good regularizer when we use all layers as locally connected layer since error is amplified in the network and training accuracy degrades.

In the search of new or non-conventional ways of regularizing neural networks, we look towards semi-supervised learning and Multi-task learning. Semi-supervised learning with ladder network is explored in next chapter.

3. LADDER NETWORKS

3.1 Why to choose this network and learning methods for regularization?

The widely used regularization techniques such as L2 and L1 regularization focuses on magnitude of weight values. Experimentally, these regularization techniques are not very useful in improving performance of locally connected neural networks. *So focusing on regularizing features as a whole i.e, focusing on what kernels are learned, rather than only magnitude of individual weights, might help to improve performance.* Semi-supervised learning and Multi-task learning has a potential to achieve this kind of regularization of features. Ladder networks uses semi-supervised learning for its classification task, therefore ladder networks were explored here.

3.2 Network

Ladder network consists of two branches , one is the encoder path and second is decoder path. Encoder path is made up of stacked denoising Auto-encoder whose purpose is to learn features in a supervised way to predict label of the input image. Decoder path takes input from last layer of encoder path and tries to reconstruct the image in an unsupervised way as it propagates information in reverse manner i.e, last layer to first layer. It has same number of neurons in each layer as its encoder counterpart. The parameters of encoder and decoder paths are shared, hence features learnt in the network are contributed by both supervised learning and unsupervised learning. This allows learning in a semi-supervised way since we can now have few labelled images which will generate the supervised cost and unlabelled images will generate the reconstruction cost.

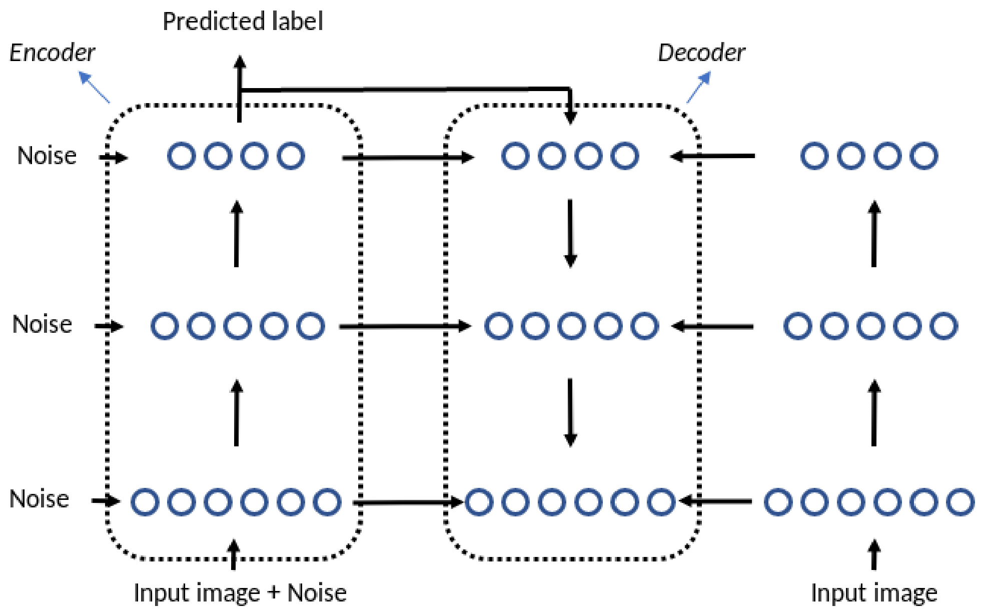


Fig. 3.1. Ladder network concept [16]. Noisy image is fed into encoder. Each layer adds a Gaussian noise $[\mathcal{N}(0, \sigma^2)]$ to the output activations and image label $\tilde{y}$ is predicted at the last layer. Supervised cost $C_{supervised}$ is calculated if the image has a label. The decoders input layer receives input from last layer of encoder and now decoder tries to reconstruct the image by passing information from last layer to first. Noise to the outputs of decoder are added by doing a dot product $g(.)$ between the encoder layer's output and decoder layer's output. Reconstruction cost $C_{reconstruction}$ at each layer of decoder is calculated by subtracting the noiseless encoder output with decoder's output of that layer.

CNN-ladder network is used here. In the experiments our encoder path is convolutional neural network and decoder path is same convolutional neural network but now the information flows from last layer of to first layer instead of first to last as in the encoder path. The same parameters or kernels used for encoder path are transposed and used for in the decoder path.

The parameter sharing between encoder and decoder path is the key of learning general features to achieve high performance due to regularization effect, but even if different parameters are used, the network can still perform better in terms of classification with few labelled images because we are making optimizer's job easy by

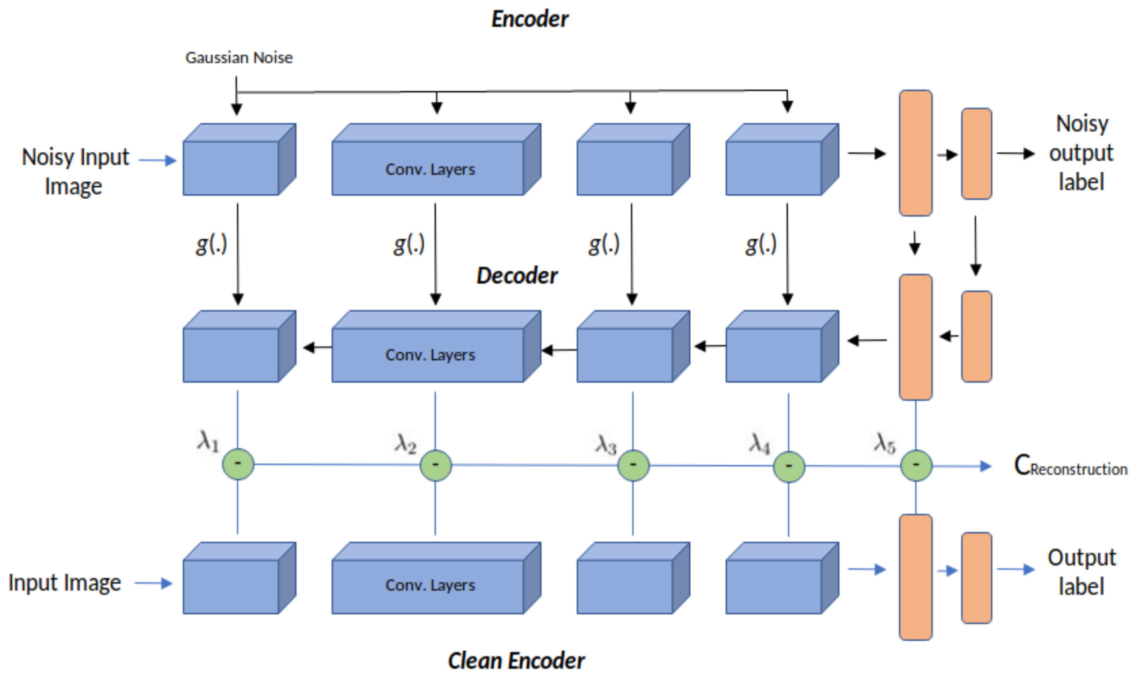


Fig. 3.2. Convolutional Ladder network [17].

removing parameter sharing constraint across tasks. So if we want more regularized kernels, parameters should be shared across the encoder and decoder but it is not necessary to do so if expected high performance is achieved without adding constraints on optimizer. If only convolutional network were to be used, it learns certain set of kernels, but when this network is used along with the decoder path, the network has to learn those general features which are useful for both paths to minimize overall cost of the network. In the experiments here the parameters are not shared across the encoder and decoder since we allow the network to learn features independently. Here the goal is to achieve high performance using as few labels as possible by learning in unsupervised way. But, if the goal is to regularize the kernels, the parameters should be shared across the encoder and decoder path.

x is the input image fed to encoder with noise,

$$\tilde{h}^{(0)} = x + noise \tag{3.1}$$

This input is now multiplied by the weights W^1 of first layer, output activations are batch-normalized and a small noise is added to the outputs. These computations are done for each layer l,

$$\tilde{z}^{(l)} = \textbf{batchnorm}(W^l \tilde{h}^{(l-1)}) + noise \tag{3.2}$$

$$\tilde{h}^{(l)} = \textbf{Relu}(\gamma * (\tilde{z}^{(l)} + \beta^{(l)})) \tag{3.3}$$

$\tilde{h}^{(l)}$ is output of the each layer l. We denote $\tilde{h}^{(L)}$ as the output of last layer of the encoder.

Now for the decoder path computations are as follows, for last layer,

$$u^{(L)} = \textbf{batchnorm}(\tilde{h}^{(L)}) \tag{3.4}$$

for rest of the layers,

$$u^{(l)} = \textbf{batchnorm}(V^{(l+1)} \hat{z}^{(l+1)}) \tag{3.5}$$

$$\hat{z}^{(l+1)} = g(\tilde{z}^{(l+1)}, u^{(l+1)}) \tag{3.6}$$

g(.) is point-wise multiplication function.

Supervised cost for the encoder is,

$$C_{supervised} = -\frac{1}{N} \sum_i log(P(\tilde{y}_i = t_i | x_i)) \tag{3.7}$$

Unsupervised cost i.e, reconstruction cost of decoder layer is given by,

$$C_{reconstruction} = \sum_l \lambda_l \left\| z^{(l)} - \hat{z}^{(l)} \right\|^2 \tag{3.8}$$

Total cost is given by,

$$C_{total} = C_{supervised} + C_{reconstruction}$$

$$C_{total} = -\frac{1}{N} \sum_i log(P(\tilde{y}_i = t_i | x_i)) + \sum_l \lambda_l \left\| z^{(l)} - \hat{z}^{(l)} \right\|^2 \tag{3.9}$$

Reconstruction cost of each layer is considered and scaled with λ_l, where l is layer number. Error in the top layer where the activations of output units of encoder layer as passed to decoder layer should be weighed more since that error is propagated through all the layers. Therefore λ_l for the top layer is larger than the λ_l of other layers. Also, the magnitude of the λ_l values does play an important role in terms accuracy performance, therefore the next section talks about a technique to modify λ_l values for achieving high performance.

3.3 Normalizing Lagrange Multipliers

λ_l are called lagrange multipliers which are hyperparameters here and contribute by scaling reconstruction cost of a layer in decoder path. Since we account cost of each layer from the decoder path in the total, the cost magnitude can reach high values during training even when majority number of layers has lower cost magnitude but one layer has high cost. So control over Lagrange multipliers λ_l is essential in order to control the magnitude of total cost. Therefore Lagrange multipliers are normalized in the following way to obtain higher performance.

$$\lambda_l^{new} = \frac{\lambda_l}{\sum_{i=1}^{L} \lambda_i} \tag{3.10}$$

Therefore,

$$\sum_{l=1}^{L} \lambda_l^{new} = 1 \tag{3.11}$$

λ_l is initially randomly chosen value to weight each layer such that λ_l for top layer is higher than the later layers. Now by normalizing these λ_l values using equation 3.10 and 3.11 we obtain new scaling factors λ_l^{new} by preserving the relation between different λ_l values. Hence, we scale down the magnitude of cost and still preserve the relation between λ_l values of each layer. Next section shows the improved results with this technique.

3.4 Results on Pavia-University dataset

Table 3.1.
Accuracy comparison of different networks on Pavia-University dataset

#	**5 labels per class**	**10 labels per class**
CNN-Ladder (with normalization)	**90.38 ±3.18 %**	**95.71 ±1.73%**
CNN-Ladder (w/o normalization)	**88.92 ±2.97%**	93.13 ±2.03%
CDL-MD-L	72.85%	82.61 ±2.95%
Co-DC-CNN	83.47 ±3.01%	**94.99 ±1.49%**
Co-DC-Res	86.69 ±2.94%	**96.16** ±1.05%
PNGrow	**88.11 ±2.87%**	93.85 ±2.23%
TT-AL-MSH-MKE	79.04 ±3.95%	86.00 ±3.04%
S^2CoTraC	50.76 ±1.68%	80.75 ±0.35%
FC-Ladder	71.2 ±1.5%	77.4 ±1.0%

Table 3.1 compares our CNN-ladder network (with and without normalization) accuracy with other competing network accuracies on Pavia-University hyper-spectral dataset which are CDL-MD-L [18], Co-DC-CNN [19], Co-DC-Res [20], PNGrow [21], TT-AL-MSH-MKE [22], S^2CoTraC [23], CNN [24].

CNN-ladder network achieves 90.38% accuracy in 5 labels per class category. The second highest accuracy other than CNN-ladder network is 88.11% achieved by PN-Grow network. Therefore accuracy improvement of 2% is achieved with CNN-ladder network with lagrange multiplier normalization on Pavia-University dataset.

The normalization of lagrange multipliers may also apply to the multi-task learning because the optimization function looks similar to the ladder network cost function with the reconstruction cost replaced with cost of different tasks.

4. SUMMARY

This work explored Locally Connected Neural Networks which are a non-weight-sharing version of Convolutional Neural Networks. A hybrid network of convolutional layers and locally connected layers called as P-LCN achieved average accuracy gain of 2% on CIFAR100 and 0.32% on CIFAR10. While training a full-LCN (all layers are locally connected layers) it was observed that in order to train the network and achieve more than 90% training accuracy, batch-normalization implementation has to be modified. Modified batch-normalization improved training accuracy from 40% to 90% i.e, 50% improvement. The testing accuracy of LCN is still lower than its CNN baseline by 13% and 8% on CIFAR100 and CIFAR10 respectively because LCNs lack suitable regularization method. Experimentally it is observed that L1, L2 and Dropout regularization does not improve performance of LCNs, hence new or non-conventional regularization methods were explored in this work. Multi-tasking and Semi-supervised learning are the learning techniques which can regularize the features on kernel level rather than individual weights, which can be beneficial since large number of kernels are learnt in LCNs as compared to CNNs. Ladder network using semi-supervised learning which achieves the kernel level regularization was explored here. Training methodology of ladder network was modified to achieve 2% accuracy improvement with 5 labels per class on Pavia-University dataset.

5. RECOMMENDATIONS

Multi-Tasking and Semi-Supervised Learning can be explored further for regularizing LCNs. Also, the LCNs can help improve performance of various multi-tasking networks. A multi-task network [25] has various architectures. The architecture where the LCNs can be relevant is shown in figure 5.1.

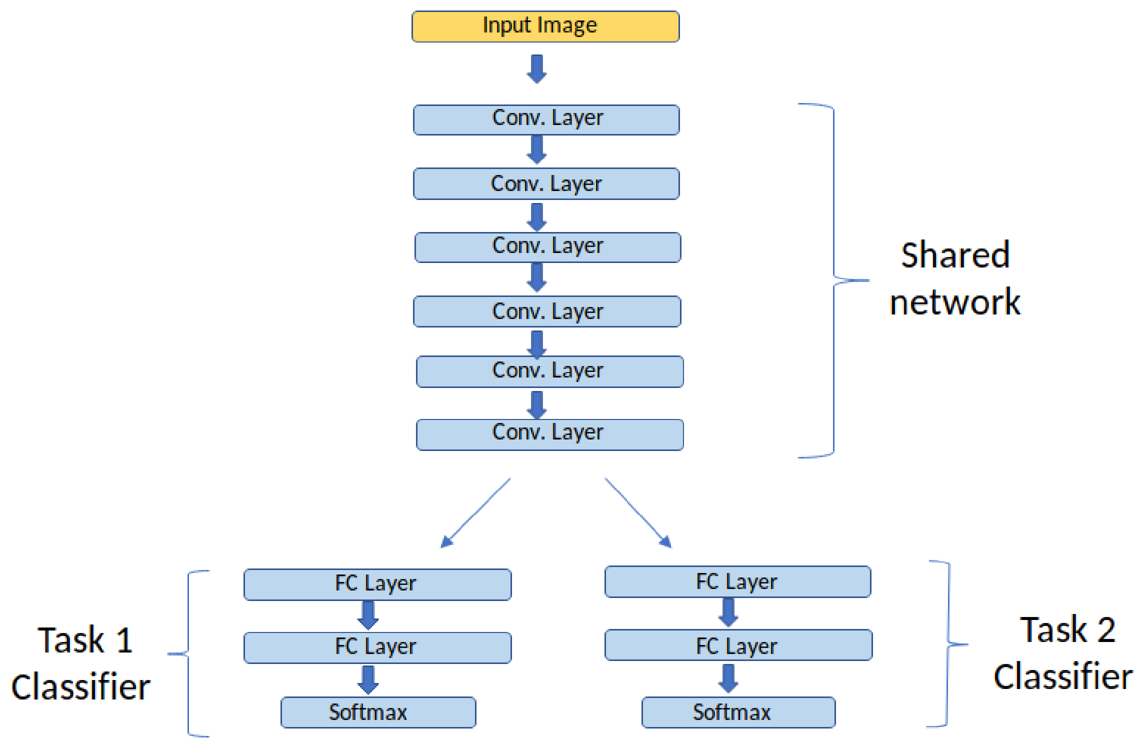


Fig. 5.1. Multi-tasking network with convolutional layers.

Figure 5.1 shows a Multi-task network which uses shared CNN layers. Both the tasks when optimized will push the shared network to learn features which are relevant to both tasks and allows the network to learn more general features as compared to the features when only one task is learned. Now, in the CNN layers, the parameters are shared across the images and across the tasks therefore can be under-parameterized. One way to over come the problem is increases number of parameters or kernels which will increase number of maps in the network. This increases number of computations during inference and also memory, therefore we trade-off memory and computations to increase the capacity of CNN layers. Instead, when an Locally Connected layer is used in place of CNN layers with same number of output maps, we increases number of parameters in the network without increasing number of computations during inference. Hence we only trade-off memory to achieve increase in capacity of shared layers. Multi-task networks needs high capacity layers in it and locally connected layers provides a suitable alternative to convolutional layers.

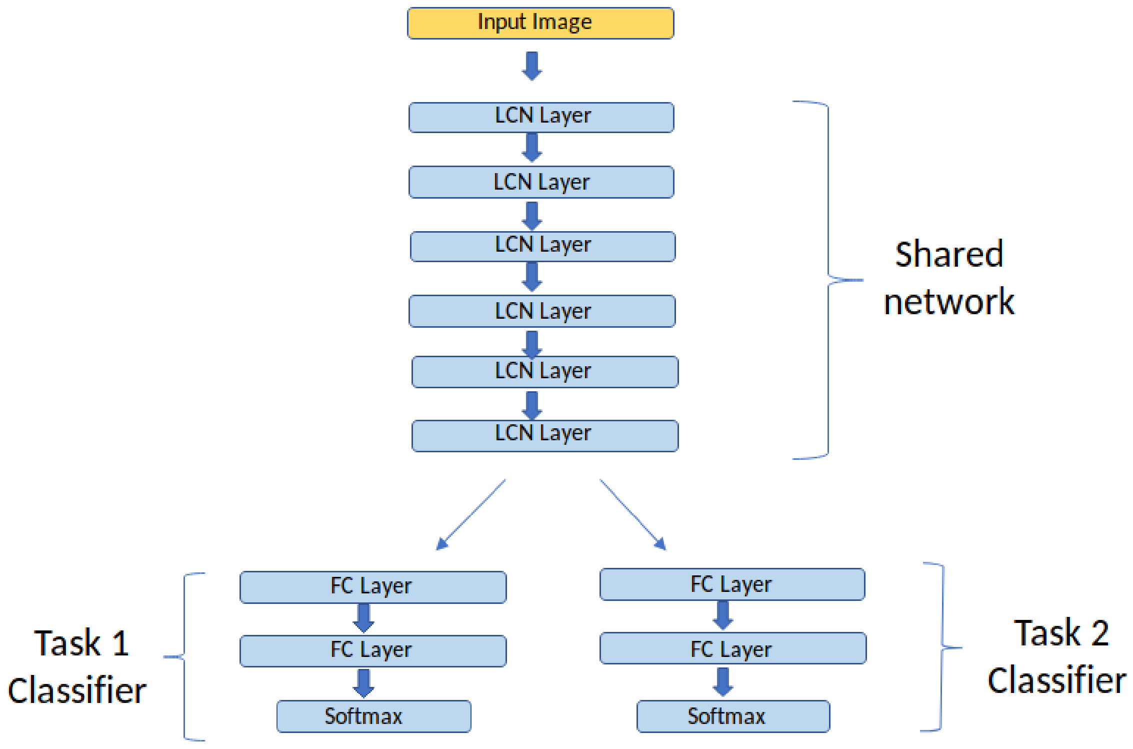


Fig. 5.2. Multi-tasking network with locally connected layers.

Figure 5.2 shows multi-task network with locally connected layers in the shared network. The objective function is as follows,

$$C_{total} = C_{task1} + \lambda * C_{task2} \tag{5.1}$$

C_{task1} and C_{task2} are the cost functions of task 1 and task 2 respectively. The λ is a hyper-parameter used to determine the effect of cost on total cost (C_{total}). If the number of task are more than two, then the cost function can be modified to equation 5.2.

$$C_{total} = \sum_{i=1}^{T} \lambda_i * C_i \tag{5.2}$$

C_i is the cost of each i^{th} task, λ_i is the scaling factor and T is the number of tasks being learned.

Multi-tasking networks allow general features to be learned and locally connected layers has potential improve the performance of these network by increasing the capacity of network without increasing number of inference computations.

REFERENCES

REFERENCES

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

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[4] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng, “Tiled convolutional neural networks,” in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1279–1287. [Online]. Available: http://papers.nips.cc/paper/4136-tiled-convolutional-neural-networks.pdf

[5] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1701–1708.

[6] Y. hsin Chen, I. L. Moreno, T. Sainath, M. Visontai, R. Alvarez, and C. Parada, “Locally-connected and convolutional neural networks for small footprint speaker recognition,” in *Interspeech*, 2015.

[7] Y. LeCun, F. Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004.

[8] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[9] A. Krizhevsky, “Learning multiple layers of features from tiny images,” University of Toronto, Technical Report, 2009.

[10] W. T. HUBEL DH, “Receptive fields of single neurones in the cat’s striate cortex,” *The Journal of physiology*, vol. 148, no. 3, p. 574–591, 1959.

[11] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, no. 1, pp. 23–63, 1987. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1551-6708.1987.tb00862.x

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167

[13] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. Morgan-Kaufmann, 1992, pp. 950–957. [Online]. Available: http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[15] Y. E. NESTEROV, "A method for solving the convex programming problem with convergence rate," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983. [Online]. Available: https://ci.nii.ac.jp/naid/10029946121/en/

[16] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder network," *CoRR*, vol. abs/1507.02672, 2015. [Online]. Available: http://arxiv.org/abs/1507.02672

[17] J. Büchel and O. K. Ersoy, "Ladder networks for semi-supervised hyperspectral image classification," *CoRR*, vol. abs/1812.01222, 2018. [Online]. Available: http://arxiv.org/abs/1812.01222

[18] X. Ma, H. Wang, and J. Wang, "Semisupervised classification for hyperspectral image based on multi-decision labeling and deep feature learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 120, pp. 99 – 107, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271616303124

[19] H. Zhang, Y. Li, Y. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sensing Letters*, vol. 8, no. 5, pp. 438–447, 2017. [Online]. Available: https://doi.org/10.1080/2150704X.2017.1280200

[20] B. Fang, Y. Li, H. Zhang, and J. C.-W. Chan, "Semi-supervised deep learning classification for hyperspectral image based on dual-strategy sample selection," *Remote Sensing*, vol. 10, no. 4, 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/4/574

[21] M. Romaszewski, P. Głomb, and M. Cholewa, "Semi-supervised hyperspectral classification from a small number of training samples using a co-training approach," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 121, pp. 60 – 76, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271616303446

[22] K. Tan, J. Zhu, Q. Du, L. Wu, and P. Du, “A novel tri-training technique for semi-supervised classification of hyperspectral images based on diversity measurement,” *Remote Sensing*, vol. 8, no. 9, 2016. [Online]. Available: https://www.mdpi.com/2072-4292/8/9/749

[23] A. Appice, P. Guccione, and D. Malerba, “A novel spectral-spatial co-training algorithm for the transductive classification of hyperspectral imagery data,” *Pattern Recognition*, vol. 63, pp. 229 – 245, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320316303259

[24] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, “Deep feature extraction and classification of hyperspectral images based on convolutional neural networks,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, Oct 2016.

[25] S. Ruder, “An overview of multi-task learning in deep neural networks,” *CoRR*, vol. abs/1706.05098, 2017. [Online]. Available: http://arxiv.org/abs/1706.05098