ImageNet Classification with Deep Convolutional Neural Networks

A seminar report submitted to Rajagiri School of Engineering & Technology in partial fulfilment of degree of

B.Tech.

in

Electronics & Communication Engineering

by

Sidharth S

Roll No:RET17EC146

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY

KERALA

DECEMBER 2020

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY

KOCHI - 682039,

Certificate

This is to certify that this report entitled "ImageNet Classification with Deep Convolutional Neural Network" is a bonafide record of the seminar presented by ***Mr.Sidharth S, Roll No.RET17EC146*** *under our guidance towards the partial fulfilment of the requirements for the award of* ***Bachelor of Technology in Electronics & Communication Engineering*** *of the* ***APJ Abdul Kalam Technological University****.*

Guide name	**Abhishek Viswakumar**
Designation	Assistant Professor
Dept. of ECE	Dept. of ECE
RSET	RSET
Kochi	Kochi

Acknowledgement

I am extremely grateful to **Prof.(Dr.)Sreejith P S**, Principal, Rajagiri School of Engineering and Technology, and **Dr.Rithu James**, Head of the Department, Electronics and Communication Engineering, for providing all the required resources for the successful completion of my seminar. My heartfelt gratitude to my seminar guide **Mr. Abhishek Viswakumar, Asst. professor, RSET** and seminar coordinators, **Mr.Naveen N, Ms.Mariya Vincent and Dr.Simi Zerine Sleeba** for their valuable suggestions and guidance. I express my sincere thanks to all staff members and friends for their help and co-ordination in bringing out this seminar successfully in time. I also acknowledge thankfully, the authors of the references and other literatures referred to in this seminar. Thank you.

Abstract

Deep Convolutional Neural Networks*(CNNs)* are used widely in the area of computer vision and image processing today and thus it is becoming increasingly important to have a comprehensive understanding of how they work. AlexNet introduced in the year 2012 has been ground breaking in the world of computer vision. This network is capable of classifying images into any one of the pre defined 1000 classes. The network consists of about 60 million parameters and 650,000 neurons. The various layers such as Convolutional Layers, Maxpooling layers, Fully connected layers and Softmax layers have been used in the network and these are all a part of the subjects for study in this seminar. The importance of activation functions and in particular, the ReLU activation function is also discussed along with other important topics such as dropout regularization which is used in the network. Study of AlexNet gives a comprehensive insight into the world of convolutional neural netwroks thus cementing the basics for diving into the world of computer vision.

Contents

Chapter 1

Introduction

Deep neural nets are widely in use today for a variety of machine learning purposes. Deep CNNs are usually used in the field of computer vision for applications such as object detection, face recognition, neural style transfer etc. The seminar deals with the study of AlexNet, a pioneer in the field of deep convolutional neural networks. This will provide us a intuition as to why CNNs are useful for image processing applications.

1.1 Introduction to AlexNet

The model under study in the main reference paper of this seminar is termed 'AlexNet' after the main author of the paper, Alex Krizhevsky. He was working at the University of Toronto along with the co-authors Ilya Sutskever and Geoffrey E. Hinton.

AlexNet is considered one of the most influential papers published in computer vision, having spurred many more papers published employing

CNNs and GPUs to accelerate deep learning. As of 2020, the AlexNet paper has been cited over 70,000 times according to Google Scholar.

Convolutional Neural Networks (CNNs) had always been the go-to model for object recognition — they're strong models that are easy to control and even easier to train. They don't experience overfitting at any alarming scales when being used on millions of images. Their performance is almost identical to standard feedforward neural networks of the same size. AlexNet is the first Convolutional Neural Network (CNN) where multiple convolution operations were used.

1.2 Dataset: ImageNet

ImageNet is a dataset made of more than 15 million high-resolution images labeled with 22 thousand classes. The key: web-scraping images and crowd-sourcing human labelers. ImageNet even has its own competition: the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). This competition uses a subset of ImageNet's images and challenges researchers to achieve the lowest top-1 and top-5 error rates (top-5 error rate would be the percent of images where the correct label is not one of the model's five most likely labels). In this competition, data is not a problem; there are about 1.2 million training images, 50 thousand validation images, and 150 thousand testing images.

1.2.1 The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and image classification at large scale. One high level motivation is to allow researchers to compare progress in detection across a wider variety of objects – taking advantage of the quite expensive labeling effort. Another motivation is to measure the progress of computer vision for large scale image indexing for retrieval and annotation.

AlexNet was originally developed by the authors to take part in the ILSVRC challenge. The performance of the network is also benchmarked with this competition and compared with the other competitors. The authors of the paper has discussed in detail about this in the original paper.

Chapter 2

A Comprehensive Study of Convolutional Neural Networks

Before going into more details about the network discussed in the paper, it is imperative to know a few basic details about CNNs. This secction serves the purpose of providing the necessary insight into the working of CNNs so as to help the study AlexNet in detail.

2.1 Need for CNNs in Computer Vision

Fully connected neural networks (*FCNNs*) are a type of neural network where the architecture is such that all the nodes, or neurones, in one layer are connected to the neurons in the next layer. Fully connected neural networks are used for a wide variety of applications in the world of deep

learning. But studies have shown that FCNNs cannot be used in computer vision applications due to a number of reasons. FCNNs involve a lot of computations, especially for datasets involving images. *For eg:* A simple 128 x 128 x 3 image given as a input would require *151, 875* units just at the first layer of the network.

There exists other problems such as overfitting, slower time for gradient descent in networks having a large number of parameters. So it can be concluded that FCNNs are not the ideal solution in the case of computer vision applications and thus arises the need for CNNs.

2.2 Convolutional Neural Networks

2.2.1 Convolution Operation

In mathematics, convolution is a mathematical operation on two functions f and g that produces a third function $f * g$ that expresses how the shape of one is modified by the other.

Mathematically,

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

2.2.2 Convolution Operation in CNNs

In the case of a CNN, the convolution is performed on the input data, which is an image, with the use of a **filter** or **kernel** (these terms are used interchangeably) to then produce a feature map. We can use an input image and a filter to produce an output image by convolving the filter with the

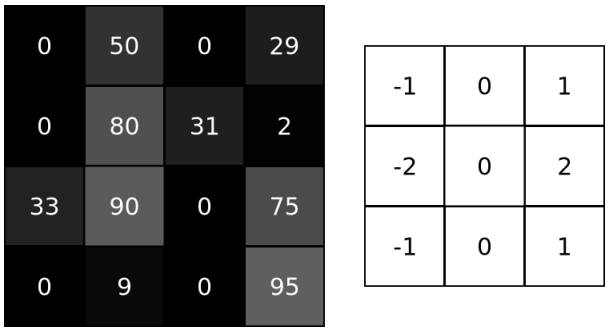

Figure 2.1: An Example - A 4x4 image (left) and a 3x3 Sobel filter (right)

input image. The steps for convolutions are:

1. Overlaying the filter on top of the image at some location.
2. Performing **element-wise multiplication** between the values in the filter and their corresponding values in the image.
3. Summing up all the element-wise products. This sum is the output value for the destination pixel in the output image.
4. Repeating for all locations.

Figure 2.2: Output of the convolution operation

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically

to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

2.2.3 Kernals or Filters

The systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as **translation invariance.**

In figure 2.1, we saw an example of convolution with the *Sobel filter*. Sobel filter is an example of an edge detector. It can be used to detect either horizontal or vertical edges present in an image. It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction. The result shows how abruptly or smoothly the image changes at each pixel, and therefore how likely it is that that pixel represents an edge.

Figure 2.3: An image convolved with the vertical Sobel filter.

Thus kernals can detect features on a global scale anywhere in the image. While Sobel filters are used to find edges, filters to find other required features, say, occurrence of a particular object in the image (object detection) can be learned using machine learning. For this, we can give random weight matrices instead of fixed values for the filters and we can make the network learn from a certain dataset.

2.2.4 Padding and Stride

Padding

One issue when applying convolutional neural networks is that we tend to lose pixels on the perimeter of our image.

Assuming that the input shape is $k_h \times k_w$, then the output shape will be $(n_h - k_h + 1) \times (n_w - k_w + 1)$ We can see that the ouput of the concolutional layer is shrinked. Especially in deep CNNs, the successive convolution operations may lead to the output layer being very small and hence affecting the learning of the network. One straightforward solution to this problem is to add extra pixels of filler around the boundary of our input image, thus increasing the effective size of the image. Typically, we set the values of the extra pixels to zero.

In general, if we add a total of p_h rows of padding (roughly half on top and half on bottom) and a total of p_w columns of padding (roughly half on the left and half on the right), the output shape will be:

$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

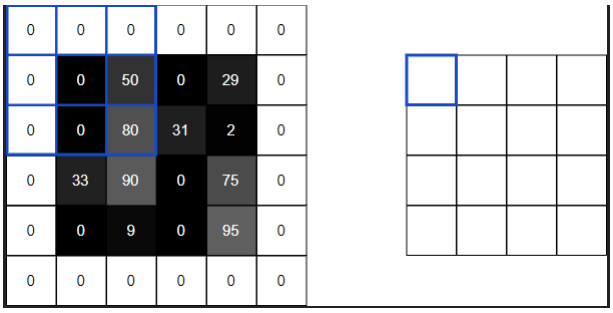

Figure 2.4: Example of padding.

Stride

When computing the convolution, we start with the convolution window at the top-left corner of the input tensor, and then slide it over all locations both down and to the right. In previous examples, we default to sliding one element at a time. However, sometimes, either for computational efficiency or because we wish to downsample, we move our window more than one element at a time, skipping the intermediate locations.

We refer to the number of rows and columns traversed per slide as the *stride*. In general, when the stride for the height is s_h and the stride for the width is s_w , the output shape is:

$$\lfloor (n_h - k_h + p_h + s_h) \ / \ s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w) \ / \ s_w \rfloor$$

2.3 Maxpool layers

A problem with the output feature maps is that they are sensitive to the location of the features in the input. One approach to address this sensitivity is to down sample the feature maps. This has the effect of making the

resulting down sampled feature maps more robust to changes in the position of the feature in the image, referred to by the technical phrase *"local translation invariance."*

Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively. Maxpool layers does not have any parameters to learn.

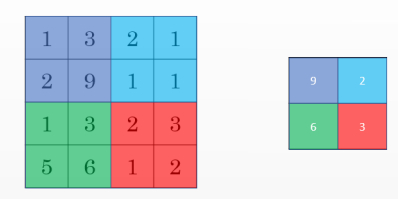

Figure 2.5: Example of a maxpool layer

The maxpool layer is basically saying, if the feature is detected anywhere in this filter then keep a high number. Thus the objective of the maxpool layer is to take care of the redundant information as the neighbouring pixels in images tend to have similar values. It also helps in reducing the spatial dimension of the input volume for next layers. AlexNet makes use of maxpool layers.

2.4 Activation Functions

The activation function is the non linear transformation that we do over the input signal. This transformed output is then sent to the next layer of neurons as input. Activation fundtion enables the network to learn complex patterns in the dataset given to it.

Need for activation function

Activations are an essential part in neural networks for a number of reasons. The primary reason as discussed earlier is that without activation functions, the network can learn only linear relationship between data. So no mattter how deep the network is, it still amounts to a linear regression model as all layers will behave same way because the composition of two linear function is a linear function itself. As most of the features we need the network to train on amounts to non-linear distribution, non linear activations functions are indispensable.

Activation functions also help in keeping the value of the output from the neuron restricted to a certain limit as per our requirement. This is important because input into the activation function is $W * x + b$ where W is the weights of the cell and the x is the inputs and then there is the bias b added to that. This value if not restricted to a certain limit can go very high in magnitude especially in case of very deep neural networks that have millions of parameters. This will lead to computational issues.

Chapter 3

AlexNet

3.1 Architecture

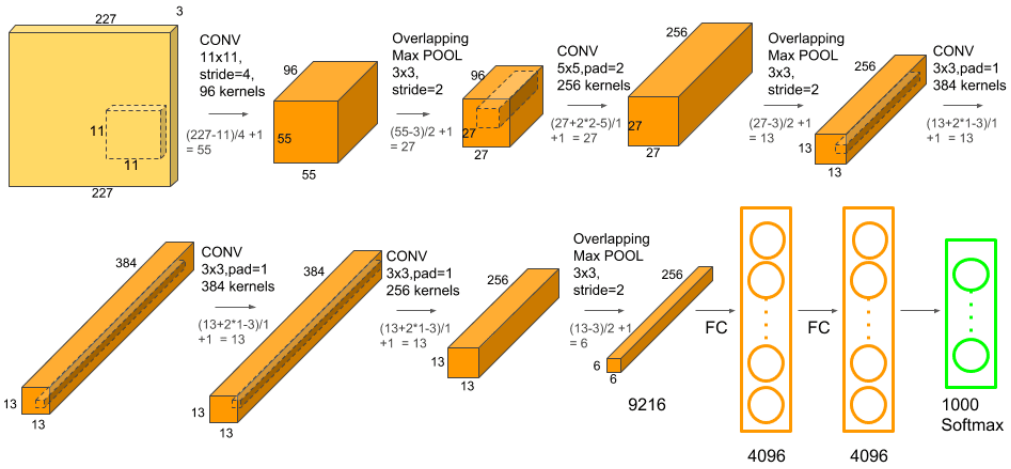

Figure 3.1: Architecture of AlexNet

Now we are ready to describe the overall architecture of our CNN. As de-

picted in Figure 3.1, the net contains eight layers with weights; the first five are convolutional and the remaining three are fullyconnected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels. The neurons in the fullyconnected layers are connected to all neurons in the previous layer. Max-pooling layers of the kind described in section 2.3 is also used.

The first convolutional layer filters the 227×227×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels. The second convolutional layer takes as input the output of the first convolutional layer and filters it with 256 kernels of size 5×5×48. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size 3×3×256 connected to the outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3×3×192 , and the fifth convolutional layer has 256 kernels of size 3×3×192. The fully-connected layers have 4096 neurons each.

In the following sections, the architecture and different features of the network are dicussed in detail.

3.2 Dataset

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowdsourcing tool. ILSVRC (see section 1.2.1) uses a subset of ImageNet with

roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

3.3 ReLU Nonlinearity

ReLU nonlinearity is an activation function which is widely in use today. But when this paper was published other activation functions like tanh activation where more commonly in use. AlexNet was one of the factors which popularised the use of ReLU nonlinearity in neural networks. The authors of the paper have discussed about this particular activation function at length.

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units. This is demonstrated in Figure 3.2, which shows the number of iterations required to reach 25% training error on the CIFAR-10 dataset for a particular four-layer convolutional network. This plot shows that a neural network as large as AlexNet would not be possible with traditional activation functions such as tanh.

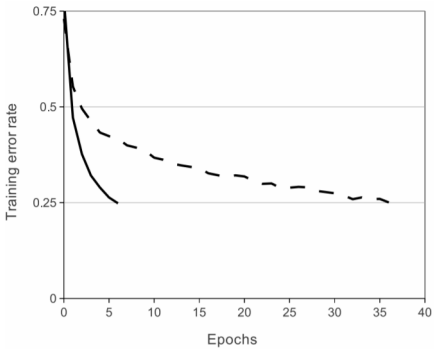

Figure 3.2: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**.

3.4 Reducing Overfitting

Since the network is large with over 60 million parameters, even the 1000 classes of the image data set was found to be insufficient to learn so many parameters without considerable overfitting. Two methods are in use in AlexNet to combat overfitting.

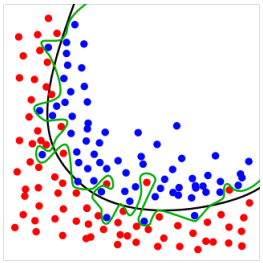

Figure 3.3: Example of overfitting - Black line fits the data well. Green is overfitting.

3.4.1 Data Augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations. Data augmentation is a technique to increase the diversity of your training set by applying random (but realistic) transformations such as image rotation. AlexNet employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. In our implementation, the transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images. So these data augmentation schemes are, in effect, computationally free.

3.4.2 Dropout Regularization

Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.

During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer.

This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many

different random subsets of the other neurons. At test time, the network use all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks.

AlexNet use dropout in the first two fully-connected layers of Figure 3.1. Without dropout, our network exhibits substantial overfitting. Dropout roughly doubles the number of iterations required to converge.

Results

Our results on ILSVRC-2010 are summarized in Table 3.1. AlexNet achieves top-1 and top-5 test set error rates of 37.5% and 17.05% . The best performance achieved during the ILSVRC2010 competition was 47.1% and 28.2%

Model	**Top-1**	**Top-5**
Sparse Coding	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 3.1: Comparison of results on ILSVRC-2010 test set. In italics are best results achieved by others.

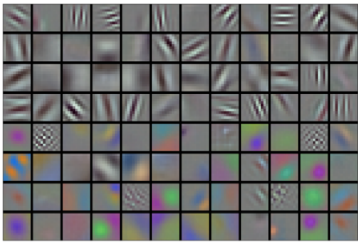

Figure 3.4: 96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images.

Figure 3.4 shows the convolutional kernels learned by the network's two data-connected layers. The network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs.

Observations

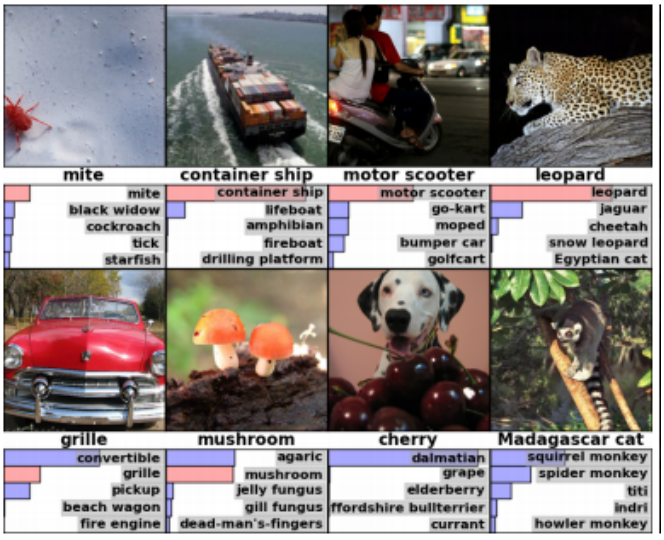

Figure 3.5: Eight ILSVRC-2010 test images and the five labels considered most probable by the model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5).

The model was found to have very good performance on the dataset provided. Use of ReLU activation and multiple GPUs enabled the network to train faster. Regularisation with dropout was found to be essential to prevent overfitting.

Conclusion

AlexNet was a pioneer in the field of computer vision and opened up a whole new research era. Dropout, ReLU, and deep layers are key steps in achieving excellent performance in computer vision task and removing any of the convolutional layers will drastically degrade the performance of the network. This indicated that deep learning is useful for machine learning tasks for computer vision as well as other applications.

Further studies regarding deep convolutional networks were done following the inception of AlexNet. This includes GoogleNet, which was the winner of ILSVRC 2014 developed by engineers at Google. ResNet, the winner of ILSVRC 2015 with a top-5 error rate of just 3.5% which beat human level performance on the dataset was the result of the research whose foundations were based on AlexNet.

Thus even today, AlexNet still remains relevant as it is the basis for many of the new developments in the field of computer vision.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[2] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu. Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1303–1314, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] G. Wang, G. B. Giannakis, and J. Chen. Learning relu networks on linearly separable data: Algorithm, optimality, and generalization. *IEEE Transactions on Signal Processing*, 67(9):2357–2370, 2019.

[1] [2] [3] [4]