Object Classification using Deep Convolutional Neural Networks

Nicholas Boddy
Electrical Engineering and Computer Science
Milwaukee School of Engineering
Milwaukee, WI 53202
boddyn@msoe.edu

March 24, 2017

Abstract

The objective of this research project is to explore the impact on performance by varying architectures of deep neural networks. Deep neural networks have resurged in interest by researchers when, in 2012, Krizhevsky et al. submitted a deep convolutional neural network to the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) and achieved significantly-higher results than the entire competition, which consisted primarily of SVMs (support vector machines). Krizhevsky et al. were able to do this by using neural networks constructed with an architecture of five hidden layers (rather than just one) while training on a dataset that is vastly larger than when neural networks were previously researched. I wish to explore many of the techniques used in their architecture, such as max pooling, dropout, GPU computation, and non-saturating non-linear activation functions, in the context of classifying objects in images. I have begun by examining the public codebase of Krizhevsky et al.'s architecture and working towards reproducing their results; I will follow this by experimenting with these previously-mentioned techniques, specifically max pooling and dropout, with the overarching goal of constructing a better-performing architecture. I expect that I will find some deeper insight on the purposes and effects of these techniques and how to use them to optimize object classification. To develop a framework for optimizing a deep neural network architecture for object classification, a better understanding of the techniques used to accompish it must be made. My objective is to determine the best way of using techniques like max pooling when designing a deep neural network architecture for classifying objects.

1 Introduction

The problem of classifying objects using machine learning has grown in recent years due to industrial demand and interest. Just a handful of the potential applications of object detection and classification are face/pedestrian detection, video stabilization, and panoramic photos. Some other applications that could greatly utilize object classification in the future include popular media like Facebook and Snapchat, scientific tools like medical imaging and other automated identifications. Research in neural networks has resurged in correlation with these types of problems that are involving some combination of computer vision and machine learning and this increase in research is believed to be largely apportioned to the successful applications of convolutional neural networks[8]. A convolutional neural network is a multi-layered neural network whose perceptrons are involved in multiple, overlapping receptive fields that resembles biological visual mechanisms of the visual cortex in most organisms. Convolutional neural networks have proven to be very successful in detecting and classifying objects[7]. The ImageNet Large Scale Visual Recognition Challenge, or ILSVRC, is a yearly competition that measures model submissions on their ability to learn and recognize large sets of objects. Krizhevsky et al.'s deep convolutional neural network was the highest scoring model in both the ILSVRC-2010 and ILSVRC-2012 classification competitions, outperforming other neural networks and other types of machine learning models[7]. Before their CNN model led the scores, support vector machines (SVMs) were de facto the highest-performing object-learning model. And even still today, most of the high-scoring models submitted to the ILSVRC are some variant of a convolutional neural network. These modern approaches to classification using convolution neural networks need to employ various techniques to achieve effective results; if left without, they are very prone to overfitting as well as impractically long training times. Some of these techniques are simply applied as guesswork, fiddling around with parameters until best results are substantially found, but there are some patterns to determining what works best when a closer consideration is taken on what the inputs and other factors are. Some of these techniques are dropout, rectified linear units as activation functions, max pooling, convolutions, and network size. The goal of my research is to address some of these techniques with the goal of gaining some insight as to how to best approach the problem of classifying objects using neural networks and develop some framework for determining a best optimization. I first start with a base source code of Krizhevsky et al.'s CNN model, and follow by adapting the architecture while making analyses on the changes in performance with different configurations.

2 Literature Search

2.1 ImageNet Classification with Deep Convolutional Neural Networks[7]

In this article, Krizhevsky et al. discuss the details of their machine learning model that they submitted to the ILSVRC 2012. Their model is a deep convolutional neural network and achieved the top results in the competition. This accomplishment is more notable due to their model being so different than the others - largely, support vector machines. To some it seems as if Krizhevsky et al. "revolutionized" usage of neural networks by architecting a model consisting of many hidden layers, which allows the model to generalize the many intricate features of an image unlike its predecessors. Their architecture consisted of five convolutional layers and three fully-connected layers. In addition to the size of the model, Krizhevsky et al. employ some various techniques to reduce overfitting and computation time. These techniques are: dropout, max pooling, usage of ReLUs (rectified linear units), parallelization across two GPUs, data augmentation, and convolutions. These will be discussed later.

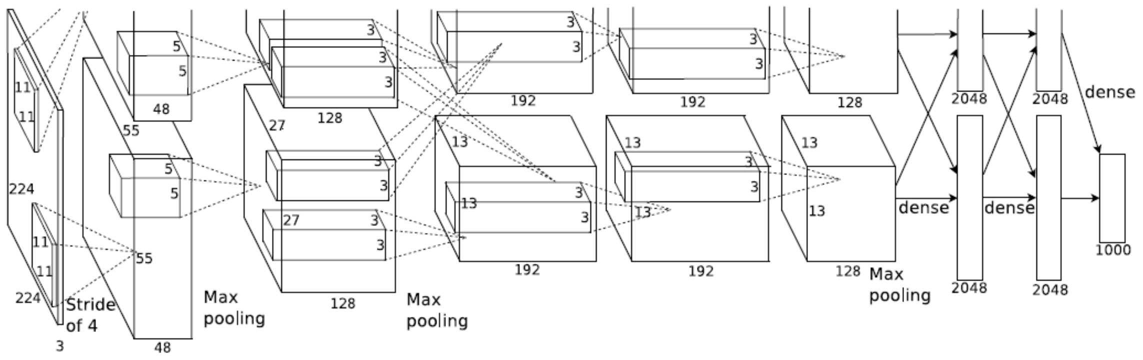
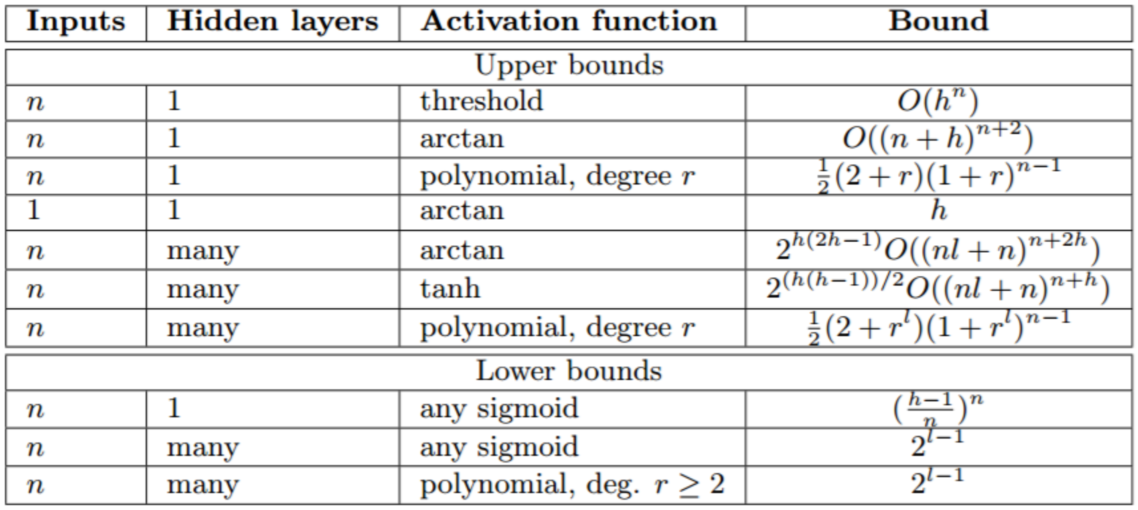


Figure 1: Illustration of the architecture of Krizhevsky et al.'s CNN submitted to the ILSVRC. This architecture is suited for two GPUs. Reproduced from [7].

2.2 On the complexity of shallow and deep neural network classifiers[1]

Bianchini and Scarselli discuss the theoretical results of deep and shallow neural networks for comparison purposes. To be more specific, they look at the upper and lower bounds of complexity for varying sizes of neural networks. They look at and compare the bound of shallow neural networks (one hidden layer) and deep neural networks (many hidden layers). Another variable that is compared is the activation function used by the neurons in the network. What this article offers my research is that it gives some insight into the effects size of neural networks has on the computational complexity of its training.



Inputs	**Hidden layers**	**Activation function**	**Bound**
Upper bounds			
n	1	threshold	$O(h^n)$
n	1	arctan	$O((n+h)^{n+2})$
n	1	polynomial, degree r	$\frac{1}{2}(2+r)(1+r)^{n-1}$
1	1	arctan	h
n	many	arctan	$2^{h(2h-1)}O((nl+n)^{n+2h})$
n	many	tanh	$2^{(h(h-1))/2}O((nl+n)^{n+h})$
n	many	polynomial, degree r	$\frac{1}{2}(2+r^l)(1+r^l)^{n-1}$
Lower bounds			
n	1	any sigmoid	$(\frac{h-1}{n})^n$
n	many	any sigmoid	2^{l-1}
n	many	polynomial, deg. $r \geq 2$	2^{l-1}

Table 1: Complexities of various configurations of neural networks. Upper and lower bounds for networks with h hidden neurons, n inputs, and l hidden layers. Reproduced from [1].

2.3 Object Detection with Discriminatively Trained Part Based Models[4]

Felzenszwalb, Girshick, McAllester, and Ramanan describe their object detection system that they call a latent SVM. This article offers another perspective separate but comparable to convolutional neural networks in the context of detecting objects. The main feature of their model is using deformable parts as features of objects to aid in detection.

2.4 Other Sources

Other resources I've gathered revolve around information pertaining to both convolutional neural networks and other methods used in object detection, such as histograms of oriented gradients and deformable parts. [6]

is an excellent web resource that discusses strategies and concepts of convolutional neural networks; Karpathy goes over different types of layers and patterns for object recognition, providing me with an additional framework of knowledge to work with when architecting a neural network.

3 Theoretical Analysis

One of the key challenges of a designing a neural network model is developing an intuition about how changes in the network affect changes in the performance. Intuition gained from small, 1-layer or 2-layer neural networks does not necessarily apply to large, multi-layer networks[3]. But Krizhevsky et al. managed to model a neural network with eight hidden layers to outperform all other models at the ILSVRC in 2012. They were able to accomplish this by identifying many intuitions about behavior and applications of neural networks that were previously disregarded.

The first intuition is that a greater number of hidden layers leads to more expressive networks than adding additional hidden nodes to a single-layer network. In other words, deep neural networks can solve and implement functions of higher complexity than shallow ones under the same constraints[1].

A second intuition is that having a higher expressiveness increases the risk of overfitting. Overfitting is the event of a model's predictions being too closely tied on the training data used as input. A model that suffers from overfitting will generally have a low error rate on the training data but a high error rate on the test data. Some techniques exist that are used to reduce risk of overfitting - dropout, data augmentation, categorizing data into training, validation, and test sets, and early stopping. Dropout is when you apply, at random during each iteration of the network, a probability of each perceptron in the network of being zeroed. These zeroed perceptrons do not contribute to the forward predictions and also are not affected by the back-propagation. This effectively reduces the inter-perceptron reliances in the network, reducing overfitting. The drawback of dropout is an increase in computation time - in usual cases, this is roughly a factor of two[7]. Data augmentation is simply taking a set of input data and generating new inputs from those. For example, an image can be mirrored, rotated, windowed, etc., to create a larger dataset where features of the image are located in all sections of the input so that the model can be more general and not overfit to the likely-centrally-located features of "good" pictures.

Another intuition is that with these larger datasets and a more complex problem (that being classifying images), a point has been reached in which many hidden layers are needed to fully describe the generalizations in the dataset[7]. And with larger datasets with inputs of greater spatial sizes, methods of efficiently reducing the input size while retaining as much useful data as possible are required for practice. Some techniques for this include max pooling, parallelization, and ReLUs. Max pooling is used to downsample the input space while maintaining as much as useful data as possible. To reduce the spatial size of the representation, a max pooling layer can be placed between convolution layers in the architecture. This will decrease computation time and also help to control overfitting[6]. Parallelization, which in the case of Krizhevsky et al. is using two GPUs in the training process[7], is a common solution to reducing computation time in many areas of research. ReLUs, or rectified linear units, which are perceptrons whose activation functions are non-saturating nonlinearities, such as the max function $f(x) = max(0,x)$, as opposed to a sigmoidal or logistic function, are found to significantly reduce the training time of larger networks[7].

With this knowledge in mind, I wish to look into specific methods of designing and optimizing a neural network model.

4 Computational Analysis

What I plan on doing with this knowledge and these resources is to learn more about these techniques and how to use them optimally, if at all, with the overarching goal of finding improvements in performance on training and classifying objects. To begin this process, I shall recreate and train a convolutional neural network model similar or even identical to that illustrated in Figure 1. Then, I can make modifications to the model's architecture and observe the changes in error rate, training time, and other affected results. A step-by-step outline for this is below:

1. Drop the max pooling layer between the convolutional layer and the fully-connected layer, and retrain the network.
2. Observe the results. I expect to find the performance to be lower, since this was addressed by Krizhevsky et. al.
3. Make alterations to the max pooling layer's configuration by perhaps changing its stride, and retrain the network.
4. Observe the results. Perhaps it's found that the error rate has decreased.
5. Modify the network to use dropout with a higher probability, and retrain.
6. Observe the results. I expect the error rate to decrease due to a lowered risk of overfitting but a longer training time, but by what factor?
7. Continue to make changes like these to gain insight into what works well and what does not when optimizing a network.

Finally, a framework can be developed for optimizing a neural network model to be used for any generalized machine learning problem. Of course, there are some prerequisites to beginning this process. Both hardware and software need to be obtained that can mimic and obtain similar results found in [7]. Large datasets would also need to be gathered. I have acquired the entire ILSVRC 2012 dataset for this research.

5 Experimental Work

In order to closely recreate the architecture described in [7], I've acquired a GeForce GTX 580 3GB GPU. Although Krizhevsky et al. use two GPUs in parallel, they've stated that it only marginally decreases the training time; I decided to cut costs and only purchase one. During the time of ordering this GPU and its duration of delivery, I decided to invest some time into researching deep neural networks in MATLAB.

In my MATLAB-guided research, I looked at matconvet, a toolbox for MATLAB that implements convolutional neural networks for computer vision. As initial research, I decided to try out the pre-trained CNNs it provides. There were several issues that came up when setting up the environment for trying out the toolbox - trying to get a C/C++ compiler support for MATLAB required some workarounds on my part on account of MATLAB's built-in support for MinGW not working, as well as lacking the proper graphics drivers to support usage of my GPU in the network's computations. After eventually overcoming these roadblocks in setting up the toolbox and configuring MATLAB to be able to compile it, I was able to successfully run the pre-trained network on images of a few select object classes for testing purposes. It seems to be run relatively well, especially when I had it configured to run with the GPU making calculations instead of the CPU. The toolbox itself seems to have a very well-made format for understanding and configuring network architectures - layers are very easily added and changed. I analyzed the pre-trained networks' layer details for insight on how these models accomplish their relatively good performance.

I've concluded from this research that, in the case that the later developments with Krizhevsky et al.'s source code fails to bring results, I can also get significant results from matconvnet as a backup plan.

Once all of the hardware came in, and once the many initial problems with booting the dedicated computer and integrating the GTX were solved, I downloaded Krizhevsky et al.'s repository in addition to the entire ILSVRC 2012 dataset, which was roughly 150GB altogether. I also installed proper versions of CUDA, which the source code relies on for integration with the NVIDIA graphics card, and other dependencies. Following that, I configured the code to comply with the local system and successfully compiled the net. After running a script that converted all of the ILSVRC dataset into batches for training, testing, and validation, I have begun to follow the initial steps outlined in section four.

6 Conclusions and Project Status

Looking at the original scope of my project, it is clear to me that I will not be able to fulfill all of my goals - those being gathering and analyzing enough results to gauge methods of determining exact incorporations of common techniques employed in deep convolutional neural networks. Part of being behind track is attributed to the several hardware and software issues in preparing and setting up the codebase of Krizhevsky et al. I split my attention during those times to also researching usage of matconvnet, the MATLAB deep neural network module.

The current state of my research is simply gathering results by training differently-configured networks on the large ILSVRC dataset and observing both the accuracies and training times. To outline a simple plan going forward, I will be following the process given in section 4; however, I will be focusing on just one or two common techniques - max pooling and dropout. Since I am continuing with Krizhevsky et al.'s codebase, I will not be using much of matconvnet; this does not mean that the time spent looking into it was wasteful, because I have still observed and may apply knowledge from that module's methods of architecting a network.

Overall, apart from the many roadblocks encountered, decent progress has been made in reaching a point where I can uninterruptedly gather and analyse data from neural networks with varying architectures. I am expecting to find conclusions on max pooling and dropout similar to my hypotheses described in section 4.

Appendix A - Terms and Definitions

1. activation function - a function that is applied to the prior output of a perceptron to transform the output. Typical activation functions are sigmoidal or activation functions, the arctangent function, or even the max function.machine learning - an area of artificial intelligence focusing on allowing computers to learn and create a model implicitly from some form of input data.

2. convolutional neural network (CNN) - a neural network that more or less takes ideas from the mechanisms of the visual cortex by having its layers take a series of different convolutional slices of the inputs for feature mapping.

3. deep neural network (DNN) - a neural network with multiple hidden layers.

4. dropout - technique used to reduce overfitting by "dropping out" hidden neurons with a certain probability by setting its output to zero.

5. ILSVRC - ImageNet Large Scale Visual Recognition Challenge, a yearly competition that measures model submissions on their ability to learn and recognize large sets of objects.

6. max pooling - technique used to downsample an input by maxing regions, overlapping or non-overlapping, into a smaller input. I.e. [4, 2, 2, 1] -> [4].neural network (NN) - a machine learning model that more or less models a human brain by mapping a series of inputs through a number of layers and into an output.

7. overfitting - the event of a model's predictions being too closely tied on the training data used as input. A model that suffers from overfitting will generally have a low error rate on the training data but a high error rate on the test data.

8. perceptron/neuron - a single node that is a part of one layer of a neural network. A perceptron takes a number of inputs and performs some basic mathematical operation with them (usually a sum) and pushes that result through an activation function, such as a sigmoid or logistic function, into a singular output. A neural network can be thought of as a network of these perceptrons strung together by layers.

9. rectified linear units (ReLU) - neurons that use a non-saturating nonlinearity as its activation function, such as $f(x) = max(0,x)$.

10. shallow neural network - a neural network with exactly one hidden layer.

References

[1] Franco Bianchini, Monica;Scarselli. On the complexity of shallow and deep neural network classifiers. *European Symposium on Artifical Neural Networks*, 2014. This article provides some insight on the complexity of different kinds of neural networks depending on factors such as number of inputs, number of layers, and the activation functions used. This is relevant to my research because it will give me more information on what can impact execution time of training and testing with complex neural networks. With this reference, I can keep in mind the trade-offs of execution time and accuracy.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society. This provides useful and explanatory information about using histograms of oriented gradients (HOGs) for detecting pedestrians. It provides insight on identifying patterns using methods separate from neural networks.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. This somewhat-dated book provides an older perspective on what was the state-of-the-art in neural network research. It looks primarily at neural networks with small numbers of hidden layers since deep neural networks (a coin termed later) were deemed useless due to the lack of intuition in modeling them. Though this book only touches on neural networks, it is useful to consider perspectives from other times and contexts.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. Is not so directly relevant, but provides another high-performance solution to object class detection. Their solution uses training on parts using a latent SVM, and is described in detail their rationale. This is useful to give insight on how other solutions are going about classifying objects.

[5] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/. Describes a method of training on objects by looking at deformable parts, such as wheels on a bicycle. This paper forms a part of the basis of the "Object Detection with Discriminatively Trained Part Based Models".

[6] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. online, 2016. Covers key aspects of convolutional neural networks in the application of visual recognition. Karpathy looks at different types of layers, layer patterns, network architectures, and techniques, as well as what each accomplishes and how to best utilize them. This source gives insight into what kinds of configurations work well when using a CNN for object recognition.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. Provides a base architecture of a deep convolutional neural network to analyze. They also provide source code on their model, which is very configurable and useful for researching effects of different techniques on performance. Krizhevsky et al. employ relevant techniques such as dropout, max pooling, ReLUs, data augmentation, parallelization, and convolutions.

[8] Gonçalo Oliveira, Xavier Frazão, André Pimentel, and Bernardete Ribeiro. Automatic graphic logo detection via fast region-based convolutional networks. *CoRR*, abs/1604.06083, 2016. This article provides details on a more recent convolutional neural network that is used with the large ILSVRC datasets. This discusses the details of their FRCN (Fast Region-based Convolutional Neural Network), which is adapted from ideas proposed by Girshick.