2017

Toward a Scientific Investigation of Convolutional Neural Networks

Anh Tran
University of Richmond

Toward a Scientific Investigation of Convolutional Neural Networks

Anh Tran

Honors Thesis[*]

Department of Mathematics & Computer Science

University of Richmond

[*]Under the direction of Dr. Arthur Charlesworth

The signatures below, by the thesis advisor, the departmental reader, and the honors coordinator for computer science, certify that this thesis, prepared by Anh Tran, has been approved, as to style and content.

Arthur Charlesworth April 21, 2017

(Dr. Arthur Charlesworth, thesis advisor)

[illegible] April 21, 2017

(Dr. Lewis Barnett, departmental reader)

[illegible] April 21, 2017

(Dr. Lewis Barnett, honors coordinator)

The signatures below, by the thesis advisor, the departmental reader, and the honors coordinator for computer science, certify that this thesis, prepared by Anh Tran, has been approved, as to style and content.

(Dr. Arthur Charlesworth, thesis advisor)

(Dr. Lewis Barnett, departmental reader)

(Dr. Lewis Barnett, honors coordinator)

Contents

Preface: About the Distinction between Engineering and Science

Arthur Charlesworth

This preface discusses why one would carry out a scientific investigation of AI.

First, it's important to distinguish between engineering and science. Roughly speaking, engineering is about building things, such as an airplane, computer system, or AI system, whereas science is about understanding properties of things in the real-world. Although not their central purpose, scientific advances sometimes lead to better engineering. For example, their discovery of the properties of the air flow across a wing, using a hand-made wind tunnel, enabled the Wright brothers to engineer a better wing design.

Related to understanding properties of things in the real-world, Donald Knuth has suggested the following, first published in *American Scientist* (journal of the Scientific Research Society):

> ... Actually a person does not *really* understand something until after teaching it to a *computer*, i.e. expressing it as an algorithm ... An attempt to formalize things as algorithms leads to a much deeper understanding than if we simply try to comprehend things in the traditional way. [1], Emphasis in original.

The criterion Knuth gives wouldn't be satisfied by a person who programs a Machine Learning system to use data to *teach itself*. The reason many Machine Learning systems are developed is, in fact, because we humans *don't* fully understand something, such as how we are able to look at photos and put them into classifications like "airplane" and "bird". Also, it's possible for a Machine Learning program to give good classification accuracy for some reasons that are not as fully understood as they should be.

In recent years, my research students have used methodologies demonstrated by others in international competitions to be highly successful for the classification of visual images. We first used Restricted Boltzman Machine (RBM) pre-training and then Convolutional Neural Nets (CNNs); the latter (but not the former) concept is explained in Anh Tran's thesis. We found ourselves engaged in hundreds of trial-and-error experiments. Increasingly our engineering attempts caused us to realize first-hand the serious lack of scientific understanding of such systems, including by the designers of such systems. For instance, the leader in successfully engineering systems that use RBM pre-training, Professor Geoffrey Hinton of the University of Toronto, had published a how-to guide. Tran and I, and a former research student of mine,[1] found its advice to be non-algorithmic recipes of very limited usefulness.

In 2014, a letter in *American Scientist* entitled "Neural Networking" raised a related question:

> ... without considerable practical experience like that obtained by Hinton's team, to construct such a successful network, an 'immense space of network architectures' must be explored. The essential question: How much of the success of such a breakthrough artificial neural network should be credited to the learning within the human minds of the construction team, trained via feedback from many experiments with such networks, and how much to the machine learning of the resulting network, eventually trained on the examples within the actual data? [2]

As a simple analogy, notice that it's one thing for each of us who speak English to use extensive trial-and-error during our early years of life, learning how to successfully pronounce vowels

[1]Many thanks to that earlier research student, Joshua Fagan. We conducted ANN experiments for two years, at the end of which Josh helped Tran start on the research project.

and consonants. It's quite different to understand just how the lips and mouth are used fundamentally differently in producing such sounds. Most who learn the former lack understanding of the latter. Likewise, it's one thing for humans to learn to recognize individual people's faces or handwritten single digits, and quite another thing for humans to learn how humans recognize faces and digits.

To further emphasize the distinction between engineering and science, consider the following. Often scientific insight about a system can be obtained by investigating the effect of a specific *degrading* of that system, exactly the opposite of an engineering approach. Revolutionary discoveries in Cognitive Science and Neuroscience about the healthy human brain have resulted from studying people who became impaired, through injury or strokes or lesions, or who have had the connection between their two brain hemispheres surgically separated (for an ethically justified reason). Non-permanent interventions are also used, such as selectively applying a strong magnetic field (TMS, Transient Magnetic Stimulation) near the surface of the brain of healthy participants, to diminish activities in certain brain regions to see whether those regions are essential for a specific cognitive ability. Another non-permanent intervention is to selectively degrade a participant's performance by increasing the demands on a participant's language processing via asking the participant to do a verbal task, while simultaneously investigating the participant's ability to perform another task (that might require the same or a different brain region) such as estimating a magnitude.

Likewise, at one point in his scientific investigation, Tran deliberately degraded the performance of a CNN by using a too-small training set, thus increasing the number of misclassified test set images, to more clearly reveal the effect of an intervention he was making on the CNN.

Scientists sometimes question plausible explanations. An elementary example is questioning the explanation that day and night are the effect of the Sun going around the Earth. The engineering of CNNs provides many plausible (even compelling) explanations for why CNNs have some of their behavioral properties. It is appropriate to question and scientifically investigate such explanations even though it seems likely that most, possibly all, such explanations are accurate.

In framing a research question, a scientist often achieves greatest success by keeping the question as simple and focused as possible. Scientists also often achieve progress on a fundamental research question by studying the simplest system available for investigating the question. For instance, a scientist interested in obtaining insight about evolution would often choose to do research on the evolution of simple organisms such as bacteria and fruit flies, rather than on humans. Also, often a scientist chooses the simplest manipulations to a system, or to a system's inputs, that have the potential for resulting effects that can help answer the scientist's questions; in Cognitive Science, many now-classic examples are in [3]; for a more recent example, see [4].

It was similarly most appropriate for Tran to use a single convolution layer in the CNNs he investigated, and to explore single choices in the size of "local receptive fields", in view of the main research question investigated in his thesis.

One important aspect of scientific investigations is the necessity to separately carry out "exploratory" experiments that seek to identify interesting potential phenomena, and "actual" experiments that seek to provide evidence of such phenomena. Even if an exploratory experiment reveals an extraordinarily highly statistically significant result, it is a serious scientific error to report the result as being statistically significant. Remarkable coincidental patterns can often be found within a sizable data set, regardless of whether such patterns represent a repeatable phenomenon.

Tran's thesis is an opportunity for him to learn from his actual experiences the importance of such things as:

- distinguishing between engineering and scientific investigations,

- distinguishing between exploratory experiments and actual experiments,
- balancing the input data used in experiments, and
- analyzing data from an experiment to see how a possibly better experiment could be designed.

He is learning first-hand that one never knows what experimental data will show until one sees the data, how unpredictable (and "messy") science can be, and that one must often revise experimental designs as one learns from previous experiments. Tran routinely uses standard methodology for training ANNs, being careful to choose training set, validation set, and test set appropriately, and so forth. His thesis is also an opportunity to learn how challenging it is to write accurately in a way that can be understood by someone who doesn't already understand.

Starting research with me during his freshman year at Richmond, Tran has carried out hundreds of different experiments, of many kinds, usually repeating each experiment 20 times to help reduce the likelihood that a choice in the random numbers that one must typically use in creating a ANN would itself explain the reason for the performance of the ANN. His first experiments were with RBM ANNs, which like all artificial neural networks can be considered to be "real-world" systems when they're implemented on real-world computers. Tran then proceeded to experiments with CNNs. Initially the data we used was our own synthetic data, having several different variants.[2] Then, we moved to briefly using real-world photographs, such as of animals and vehicles, from the public image repository known as CIFAR-10. Then we moved to simpler real-world images of handwritten single-digits and handwritten single let-

[2]At one point, to enable us to increase our number of synthetic images, we put a non-trivial amount of random noise into all the training set, validation set, and test set images. Seeing the high accuracy of the subsequent experimental test set results, Tran conjectured that the ANNs were now learning – in effect – to ignore such noise, and an experimental design change then produced data confirming his conjecture. Thus the use of random noise had resulted in a test set – although by design provably mathematically disjoint from training set and validation set – that was in effect not disjoint from either of those two other sets, hence using the test set (when we had inserted random noise) could not support any justification of a generalization ability.

ters of the English alphabet, from public image repositories initially compiled by a U.S. agency, the National Institute for Standards and Technology.

In terms of computer software, our early experiments used ANNs programmed in the Octave language, with significant programming by Tran. Then we used RBM-related ANNs written in Octave, again with significant programming by Tran.[3] Tran also wrote several software platforms for running experiments and automatically collecting data. The actual experiments reported in Tran's thesis result from slight modifications to Python and Theano programming by a reputable researcher, Michael Neilsen, who provides initial such software on the internet. In terms of computer hardware, we began using Linux systems within the Computer Science program and then supplemented them with classroom iMacs (with the permission of Information Services). For several months we then used a cluster of computers purchased by the Physics Department (thanks to Professor Gilfoyle). We finally used Tran's Windows desktop, built from components including a high-speed Graphics Processing Unit, commonly used by ANN research teams.

The main research question of Tran's thesis can be stated as "Does the best choice of receptive field size for a CNN, with a single convolution layer and a single receptive field size, *depend on the specific application*?" The thesis investigates CNNs having commonly-used "filters" and "pooling", the combination of which might influence the effect of choosing a particular receptive field size. His thesis explains the relevant terminology, reports some experimental results, and suggests the need for further investigation of related questions.

[3]Programming our own software, rather than using standard software like Keras and TensorFlow and Caffe, facilitates making custom modifications. For gradient descent we used standard conjugate gradient software that avoided the need to specify a learning rate, but for RBM pre-training another learning rate is needed and Hinton's how-to guide for RBM gives no algorithm for choosing that rate. We invented and implemented our own algorithm for choosing that rate, an algorithm intuitively more time-efficient than either grid search or random search, and our program also turned on and off the clock repeatedly at appropriate times during that algorithm, to give us a report on the total ANN training time that factored out the time for choosing the RBM learning rate.

References

[1] D. E. Knuth, Selected Papers on Computer Science, New York: Cambridge University Press, 1996; the passage was first published (with slight modification) in his article Computer Science and its relation to Mathematics, *American Scientist*, Vol. 61, Num. 6, pp. 707–713, 1973.

[2] A. Charlesworth, Neural Networking (letter), *American Scientist*, Vol. 102, Num. 5, pp. 323 – 324, 2014.

[3] S. Dehaene, *The Number Sense: How the Mind Creates Mathematics*, Oxford University Press, New York, 1997.

[4] D. Landy, A. Charlesworth, E. Ottmar, Categories of Large Numbers in Line Estimation, *Cognitive Science* Vol. 41, Num. 2, 326-353, 2017.

1 Introduction

This thesis does not assume the reader is familiar with artificial neural networks. However, to keep the thesis concise, it assumes the reader is familiar with the standard Machine Learning concepts of training set, validation set, and test set [1]. Their usage is intended to help ensure that the Machine Learning system can generalize its training from input examples used during its training to "similar" kinds of examples never used during its training.

The concept of a Convolutional Neural Network (CNN) is one of the most successful computational concepts today for solving image classification problems. However, CNNs are difficult and time-consuming to train. Such training requires automatically adjusting parameters to find a choice of parameter values that achieves good classification accuracy. In order to achieve good accuracy, a CNN must sometimes have several million parameters, which can cause the automatic training to require several days on even very powerful current hardware. Moreover, designing a successful CNN can require a wise choice of a dozen or more **hyper-parameters**, a concept further explained in Section 4. This thesis investigates the relationship between a certain hyper-parameter and the CNN's resulting accuracy using that hyper-parameter on different applications. Our main research question is stated in Section 4. It uses terminology that needs to be explained first.

2 Feedforward Neural Network

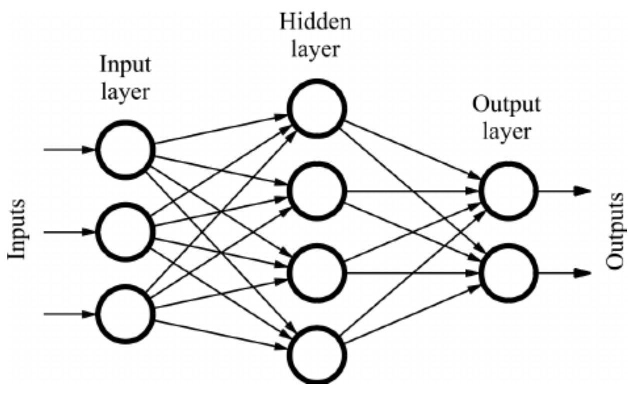


Figure 2.1: A feedforward neural network with one hidden layer [8]

To understand what a CNN is, it is helpful to first understand the simpler concept of a feedforward artificial neural network (ANN) for vision applications. An ANN has multiple connected nodes similar to the network concept in discrete mathematics. The nodes are divided into three main types of layers (Figure 2.1): one input layer, one or more hidden layers, and one output layer. All layers are presented in the form of a one-dimensional array of numbers. In this thesis, we use neural networks to classify grayscale images into different classes. In such an application, the neural net is usually structured as follows:

- The number of nodes in the input layer is equal to the number of pixels in a discretized input image. Each node in the one-dimensional input layer gets the value of one corresponding pixel in the two-dimensional input image.
- Each hidden layer can have one or more nodes.
- The number of nodes in the output layer is equal to the number of classes in the classification problem.
- The information within the neural network is transmitted strictly forward from one layer to the next. This explains the term "feedforward".
- No arc connects two nodes within the same layer.

Typically, all layers are **fully-connected**, except for the input layer. A layer is said to be fully-connected if each node in that layer is connected to all nodes in the immediately preceding layer via arcs, with each arc having its own real-number weight. We call this arc weight a **parameter**; all parameters in typical feedforward neural networks are initially randomly assigned using a normal distribution.

Each node in an ANN also has its own input and output. Except for the nodes in the input layer, the input to a node is the weighted sum of the outputs of the nodes in the previous layer:

$$\mathit{input} = w_1x_1 + w_2x_2 + \cdots + w_nx_n = \sum_{i=1}^{n} w_ix_i$$

Here, w_i represents the value of a weight of an arc, and x_i represents the output value of the node in the previous layer connected via that arc to the node in the current layer being described. The weighted sum then serves as an input to a non-linear activation function to create the output of the described node in the current layer. This output then serves as part of the input to a node in the next layer, if a next layer exists.

3 Convolutional Neural Network

The CNN concept is a modification of the ANN concept. A CNN usually has five types of layers: one input layer, one or more **convolution** layers, one or more **pooling** layers, one or more fully-connected layers, and one output layer. The concepts of an input layer, a fully-connected layer, and an output layer in a CNN are identical to those of an ANN. In this section, we briefly explain how convolution layers and pooling layers work, as well as the components within those layers. Since each actual input image we used in our experiments is two-dimensional, for simplicity, our standard illustrations of a CNN show the input layer as two-dimensional, ignoring the actual one-column-at-a-time mapping between two-dimensions and one-dimension.

Unlike an ANN, in which the hidden layers are fully-connected, a node in a convolution layer of a CNN only connects to a subset of nodes in the preceding layer. This subset of nodes is called a **receptive field**, and in our investigation is square-shaped as illustrated in Figure 3.1. Although the following explanation is specific to the input layer and first convolution layer, the same approach is typically used for multiple adjacent pairs of such layers in a CNN.

Starting from the top left region of the input layer, conceptually the receptive field is slid horizontally and/or vertically over a number of pixels, called a **stride length**, to get another receptive field. Conceptually, this sliding of receptive field to get the next receptive field is done until reaching the bottom right corner of the input layer. All nodes within a single receptive field are connected to one node in the first hidden layer, and that is done for each receptive field separately.

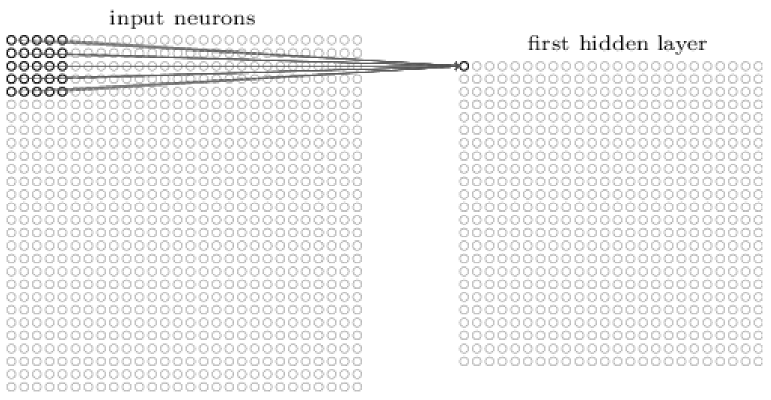


Figure 3.1: Illustration of a receptive field [7]. In this example, the input layer is a 28-by-28 square of nodes, and a receptive field is a 5-by-5 region (darker color) in the input layer. (For illustrative simplicity, only 5 of the actual 25 arcs that connect the receptive field and a node in the first hidden layer are shown.)

Merely for expository simplicity, we consider receptive fields of size 5 x 5 here. Figure 3.1 and 3.2 indicate that, for such a receptive field, each of the 24 x 24 nodes in the first hidden (and convolution) layer is connected to its corresponding receptive field via 25 arcs (which have, in turn, 25 corresponding weights). Moreover, all of the 24 x 24 nodes use exactly the same corresponding 25 weights, a concept called **weight sharing**. The intended purpose of weight sharing is to extract position-independent features from the preceding layer. In image classification, a feature is a piece of information, such as a line or a corner within an image, which could help classify what kind of image it is. In a standard feedforward ANN, an extracted feature is usually position-dependent; for instance, an ANN detecting a line at one position of the input may not be able to detect such a line that appears at another position. Weight sharing is intended to promote position-independence by forcing every one of the 24 x 24 nodes in the convolution layer to extract the same feature regardless of where the feature appears in the input image. To extract more than one feature from the input, multiple sets of 25 weights are used. Each such set of weights is said to define a **filter**; a 24 x 24 set of nodes that uses a single filter is said to form a **feature map**. Each convolution layer in a typical CNN uses multiple feature maps, and our experiments used 20 of them, regardless of the size of the receptive field used.

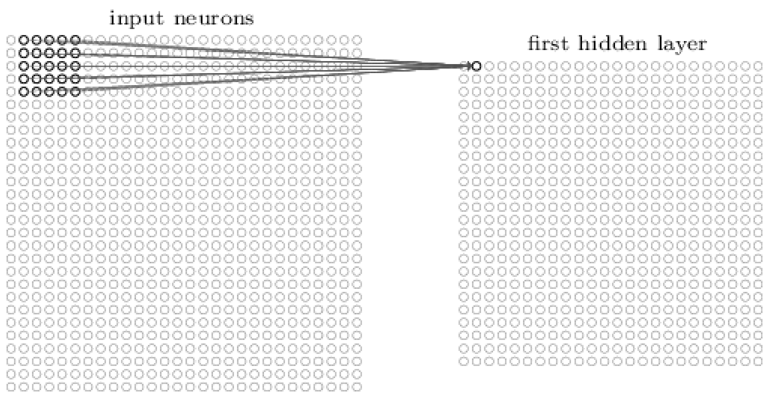


Figure 3.2: Illustration of a stride length [7]. From Figure 3.1, we move the receptive field one pixel to the right (and also, eventually, downward). In the illustration, the stride length is equal to 1. After the receptive field reaches the bottom right corner of the input layer, we have established connections to 24 x 24 nodes in the convolution layer.

A **pooling** layer is sometimes added after each convolution layer. A pooling layer "subsamples" over the convolution layer. The type of subsampling we use in our CNNs is called max-pooling. Figure 3.3 illustrates how max-pooling works when a convolution layer is pooled into 2-by-2 regions, so that each such region considers 4 nodes.

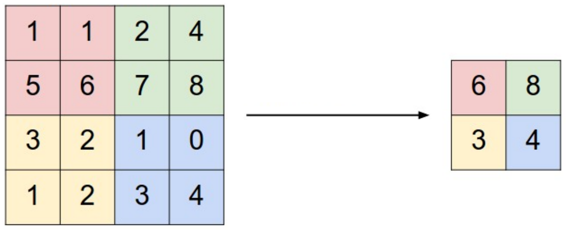


Figure 3.3: Illustration of max-pooling [9].

Weight sharing and pooling are intended to help reduce the total number of parameters in a CNN from those used in fully-connected feedforward neural networks. Reducing the number of parameters in a CNN is one way to seek to enhance the CNNs ability to generalize to images not shown to the CNN during its training. Reducing the total number of parameters also allows saving space when storing the CNN, and could speed up training.

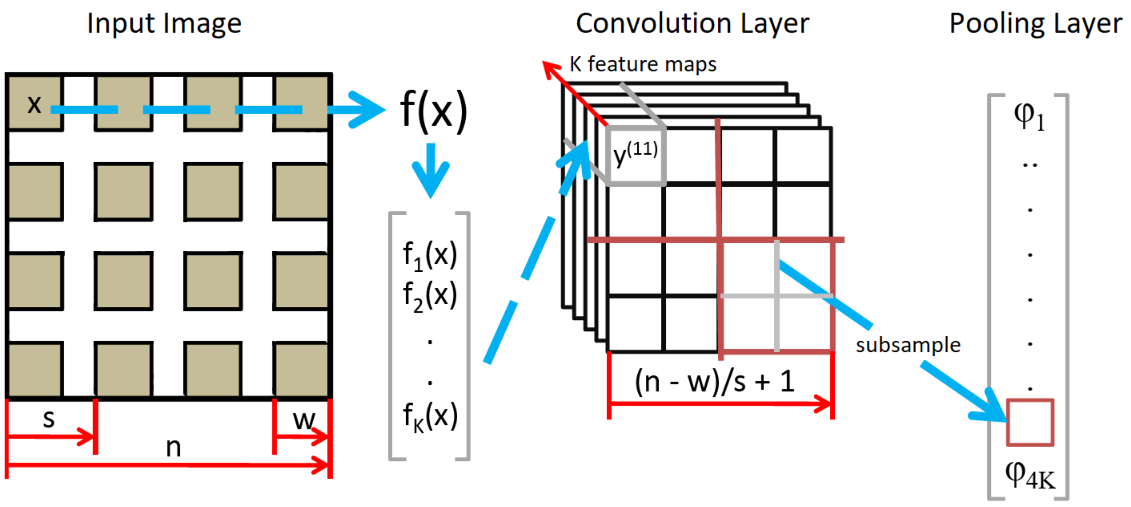


Figure 3.4: Example of a convolution layer and a pooling layer [2]. In this figure, x represents a single receptive field of size w-by-w, s is a stride length, n is the number pixels in one dimension of the input image. x is fed through an activation function f(x) to get y, which is the output value of a node in the convolution layer that connects to receptive field x. A 2-by-2 subsampling is applied to the convolution layer to get each corresponding φ in the pooling layer. Note that unlike Figure 3.3, the pooling layer in this example is represented in one-dimension.

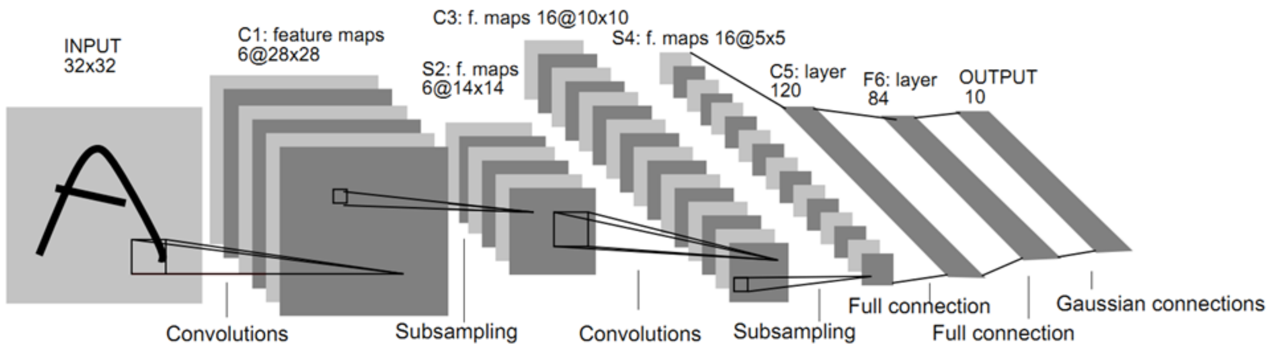


Figure 3.5: Example of a complete convolutional neural network [4].

4 Our Main Research Question

As mentioned in the introduction section, one problem confronting the designer of a CNN is how to choose hyper-parameters. Examples of hyper-parameters are: number of convolution layers, number of fully-connected layers, receptive field size (or sizes, when there is more than one convolution layer and each convolution layer uses a different receptive field size), stride length, number of feature maps, pool size (or sizes), type of pooling, etc. Unlike parameters, hyper-parameters cannot be learned by standard training algorithms such as "backpropagation" (not explained in this thesis). Currently, the most popular methodologies in use are algorithms that involve searching through a space of possible hyper-parameter values. Because of the large number of hyper-parameters, this searching process could require much time and other resources. Many engineers rely on heuristic "rules of thumb" (related to the passage from a 2014 issue of *American Scientist* quoted in the Preface). These rules of thumb can be quite vague (they are more akin to rough recipes than to algorithms) and do not always apply well to new problems. The importance of finding better ways to choose hyper-parameters is well-known. A better scientific understanding of them might help.

With that background in mind, here is our main research question:

> **Does the best choice of receptive field size for a CNN, with a single convolution layer and a single receptive field size, *depend on the specific application*, even in the presence of filtering and pooling?**

Intuition behind our main research question:

The intuition described in the current discussion ignores the fact that a CNN's input layer is only one-dimensional rather than two-dimensional. In addition, the intuition could be further undermined because of ignoring the effect of filters and pooling.

Imagine viewing an enlarged image of a handwritten digit through a square hole cut-out of a large piece of cardboard covering the image, where the square is like a receptive field size. Your task is to guess the digit. As you slide the cardboard across the image, you get more clues, but you are not allowed to recall the *locations* of the clues. Given that restriction, some digits might be easier to recognize than others. Figures 4.1 to 4.3 illustrate the intuition.

To address our main research question, we keep all hyper-parameters constant except for the receptive field size, and compare test set accuracies of different receptive field sizes on different applications. In details, we used the following hyper-parameter values for our CNN, taken from a system largely programmed by Machine Learning researcher Michael Nielsen:

Number of convolution layers	1
Stride length	1
Number of feature maps in the convolution layer	20
Number of pooling layers	1
Pooling size	2 x 2
Activation functions	rectifier
Type of pooling	max
Number of fully-connected layers	1
Number of nodes in the fully-connected layer	100
Number of backpropagation iterations	60

In this table, the "rectifier" used is the function $f(x) = max(0, x)$, where x is the input to a node.

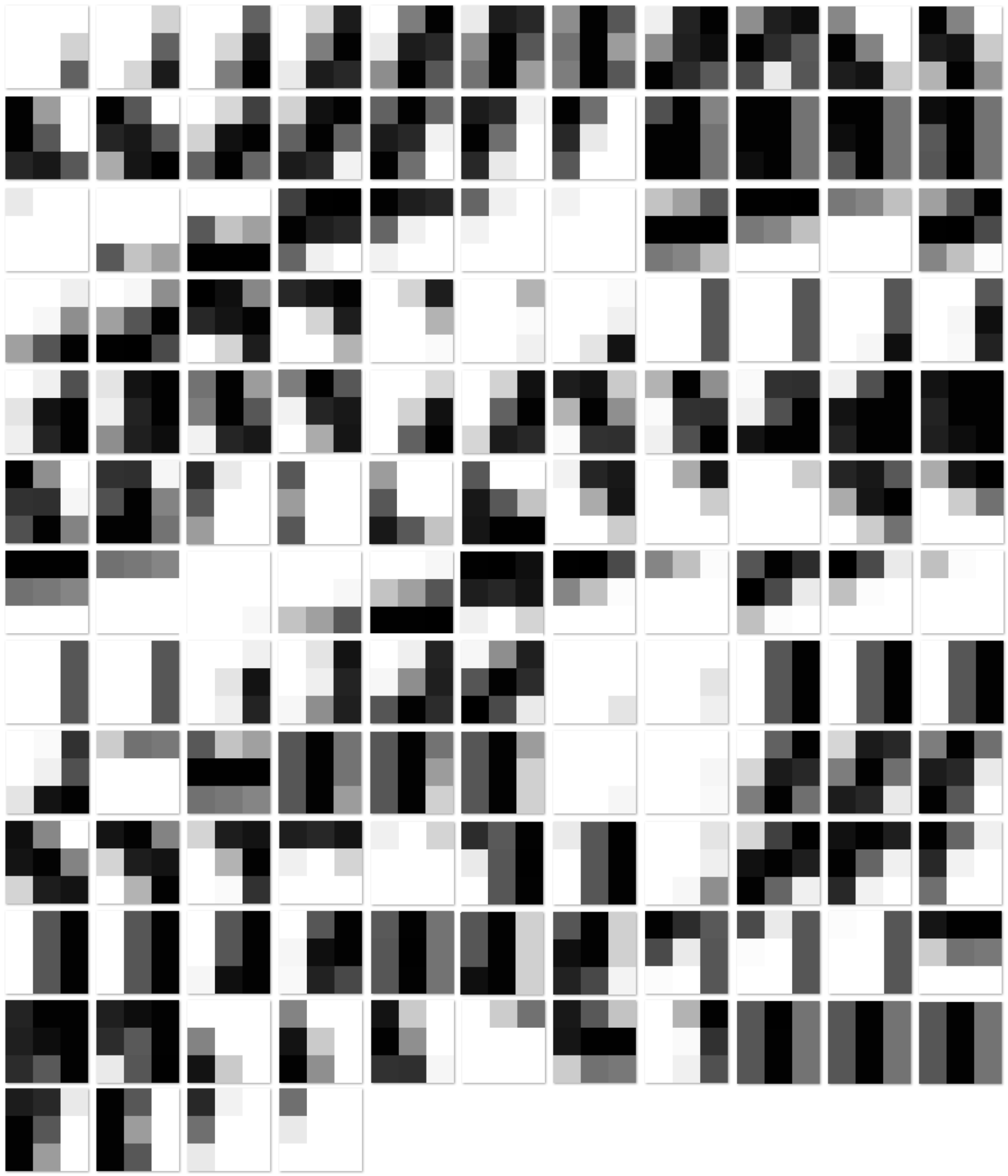


Figure 4.1: Suppose you are given a 28-by-28 grayscale image of a handwritten digit 9 (you do not know that yet). Suppose you use a cardboard sheet with a small cutout of size 3-by-3, and slide it across the image using a stride length of 1. Figure 4.1 shows all 136 not-fully-white receptive field sub-images that you could get. Without positional information, these receptive fields fail to capture necessary features that could help recognize the digit 9.

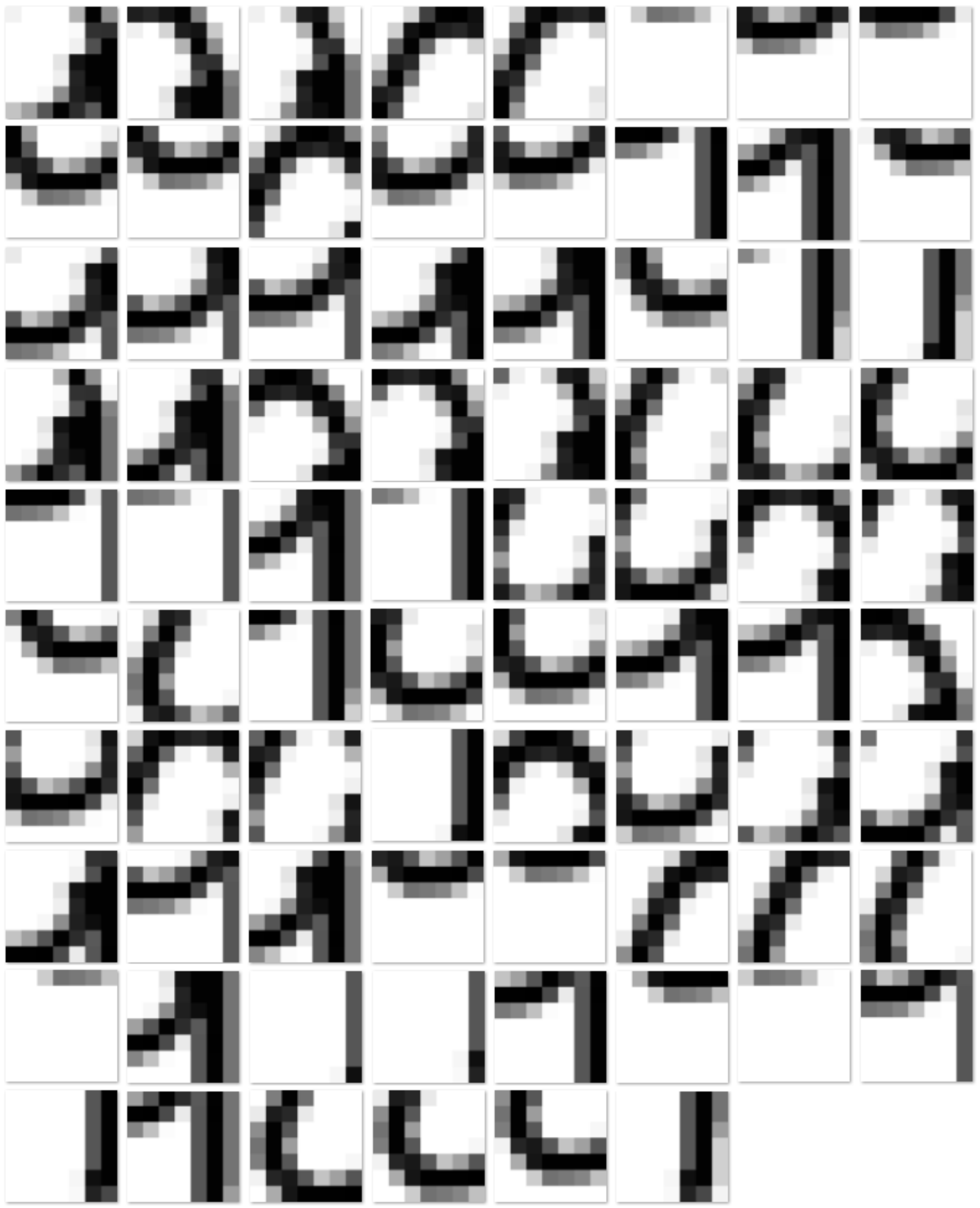


Figure 4.2: Suppose you switch to a new cardboard with a larger cutout of size 7-by-7, and slide it across the image using a stride length of 1. Above are all 78 not-fully-white receptive field sub-images that you could get. Intuitively, it seems that the digit 9 is now easier to recognize.

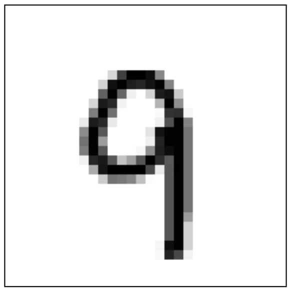


Figure 4.3: Picture of the 28-by-28 handwritten digit 9 used in Figure 4.1 and 4.2, taken from the MNIST data set (explained in Section 5) [3]. Roughly 80% of it is fully-white.

5 Images Used in Experiments

The data we used for our experiments are based on the NIST Special Database 19, one of the biometric databases compiled by the National Institute of Standards and Technology (NIST). The NIST Special Database 19, which we call NIST-19, contains over 800,000 grayscale images of handwritten digits, lower-case letters, and upper-case letters [6]. Classification of handwritten digits and letters is an important problem that appears in numerous practical applications. The United States Postal Service, for example, successfully deployed a postal-address interpretation system using neural networks that helps save a lot of time and resources [10].

Each image has size 128-by-128, with each pixel having an integer value between 0 and 255, inclusive. The handwritten digits and letters in NIST-19 were obtained from two sources: Census Bureau employees in Suitland, Maryland (source 1), and Bethesda high school students (source 2).

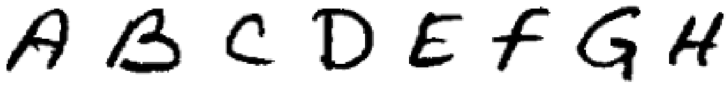


Figure 5.1: Examples of 8 images from NIST-19 [6].

In our experiments, NIST-19 was divided into two smaller data sets. The two data sets are described in this section.

5.1 The Standard MNIST Data Set

The main data set we used for our research is the standard MNIST data set widely used in Machine Learning experiments. MNIST is a subset of NIST-19 that only includes handwritten digits. The examples in MNIST were selected and processed by a team led by Yann LeCun of New York University [3].

The MNIST data set contains 70,000 grayscale images of handwritten digits, 0 through 9. These 70,000 images are usually divided into smaller subsets in different ways by different researchers. We used the version of MNIST divided by a Machine Learning researcher named Michael Nielsen, consisting of 3 disjoint subsets: 50,000 images in the training set, used for choosing the parameters, 10,000 images in the validation set, used for choosing the hyper-parameters, and 10,000 images in the test set, used for evaluation of the resulting system's ability to generalize beyond the examples used in its training. (Parameters and hyper-parameters for CNNs are explained in Section 2 and Section 4). From the original 128-by-128 grayscale images in NIST-19, Yann Lecun's team used the following procedure to adjust the images:

> "The original black and white (bi-level) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field." [4]

As a result, each image in MNIST has size 28-by-28, with each pixel having an integer value between 0 and 255 inclusive. Figure 5.2 shows 100 examples of such images:

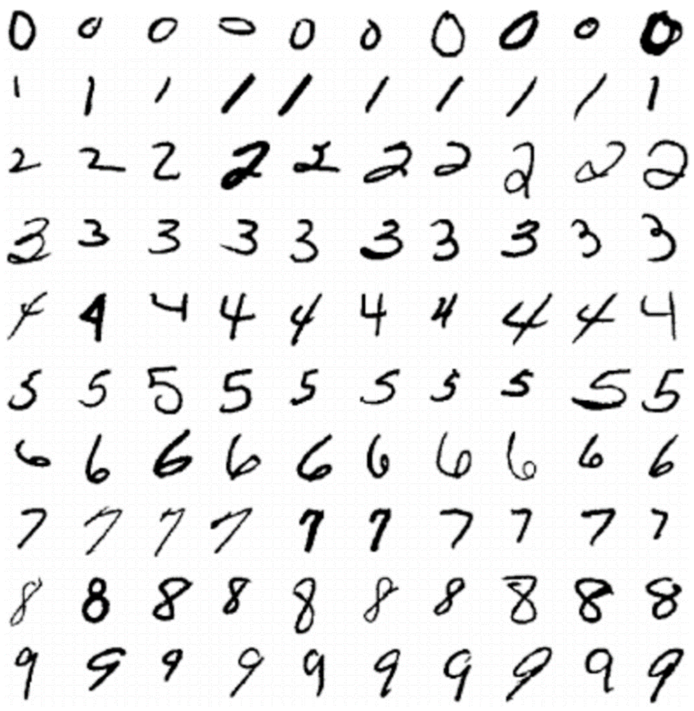


Figure 5.2: Examples of 100 images from MNIST [5].

Some digits in the original MNIST have more examples than others. In our experiments, we **balanced** the number of images of each digit. To see the importance of such balancing, consider the following simplified illustration of a binary classification problem in which the CNN only needs to classify two digits: digit 0 and digit 1. If there had been 8000 images of digit 0 in the test set and just 2000 images of digit 1, then a system that would classify every single image as digit 0 would not demonstrate any substantial ability to classify individual images. Yet such a system would achieve 80% accuracy (8000 correctly classified images out of 10000), substantially better than the 50% accuracy achieved by random guessing. To achieve balancing for this illustration, exactly 2000 images - the minimum of 8000 and 2000 - should be randomly chosen from the 8000 images of digit 0. Similarly, if the problem is to classify 10 digits in MNIST, and the numbers 8000 and 2000 in the illustration had been something like 3000 images of digit 0, 2900 images of digit 1, 3100 images of digit 2, etc., similar balancing should be applied, replacing the 10 numbers 3000, 2900, 3100, etc., by the minimum of those 10 numbers.

The experiments later described in Section 7 involve two ways of dividing MNIST into smaller data sets. Within both, we applied balancing.

5.2 Our NIST-9 Data Set

The second data set that we used is called NIST-9. After obtaining experimental evidence related to our Main Research Question on images of handwritten digits using the MNIST data set, we sought to broaden experimental evidence using other kinds of images. In particular, we selected images of nine upper-case letters from NIST-19 to create a data set, which we call NIST-9, as indicated in the following table:

E	Has similar shape to "3"
I	Has similar shape to "1"
L	Has similar shape to "7"
M	M is very roughly upside-down W, similar to how 6 is very roughly upside-down 9
N	Has similar shape to "2"
O	Has similar shape to "0"
P	Has similar shape to "6" or "9"
S	Has similar shape to "5"
W	W is very roughly upside-down M, similar to how 6 is very roughly upside-down 9

We did not include a letter similar to either digit 8 or digit 4. (There are two different kinds of handwritten 4's: one with a closed top, and one with an open top; see Figure 5.3.)

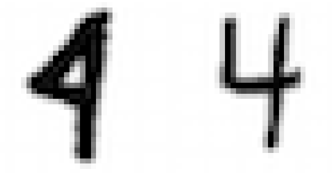


Figure 5.3: Two kinds of handwritten 4's [3].

We applied balancing as described in Section 5.1. We observed that “W” had the fewest handwritten upper-case letters (5026 in total) obtained from source 1, and “E” had the fewest handwritten upper-case letters (365 in total) obtained from source 2. As a result, for each of the 9 capital letters, we randomly selected 5026 images from source 1, and 365 images from source 2, for a total of 5026 + 365 = 5391 images of each digit. Of these 5391 images of each capital letter, we divided them into three disjoint sets: training set (4/6 of the images), validation set (1/6 of the images), and test set (1/6 of the images), as shown in the following table:

Training set	3592 images of each capital letter
Validation set	898 images of each capital letter
Test set	898 images of each capital letter

In addition, we used the same processing procedure by Yann LeCun, described in Section 5.1, to center and resize the images to size 28-by-28. Due to time limitation, this thesis does not further discuss the capital letter images.

We originally considered letter “B”, since it had similar shape to digit “8”, but the number of handwritten “B” images in NIST-19 was very small. Carrying out the necessary balancing with such a small number of images would have resulted in few images in the test set. In turn, that could have risked obtaining experimental results that are statistically significant but not substantial. Section 6 will explain the distinction between being statistically significant and having substantial effect in the context of the investigations this thesis pursues.

6 Why High Statistical Significance Can Be Insufficient

As mentioned in the last paragraph of Section 5, a relatively small test set can cause misleading statistical significance. This section describes one of our early exploratory experiments that demonstrates such a misleading statistical phenomenon.

The early experiment used the standard MNIST data set. We created what we call “one-versus-one” data sets, each of which included images of two different digits. For example, a “2-versus-3” data set only contained images of handwritten digits 2 and 3, partitioned in the following way:

	Digit 2	Digit 3
Training set	2484 images	2550 images
Validation set	495 images	515 images
Test set	516 images	505 images

Recall that Section 5 explains the important concept of balancing the data used in experiments. The numbers of images of each kind of digit given in the above chart were not balanced as carefully as the image sets we used in our later experiments, but we do not think such lack of balancing explains the phenomenon described in this section. After training a CNN with one convolution layer on this 2-versus-3 data set, we obtained the following results on the test set:

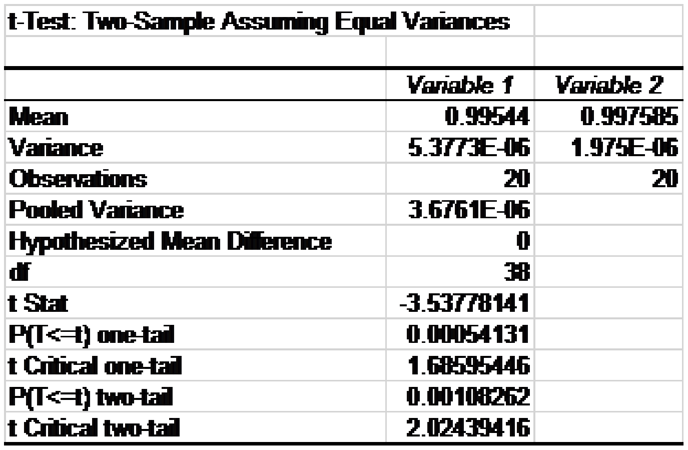
		Digit 2	Digit 3
Receptive field size 5	Mean accuracy	99.54%	99.67%
	Standard deviation	0.23%	0.17%
Receptive field size 5	Mean accuracy	99.76%	99.52%
	Standard deviation	0.14%	0.14%

For expository simplicity, we shall refer to a square receptive field using only the length of one of its side. For instance, a 5-by-5 receptive field will be said to have “size 5”. In the table above, mean accuracy represents the average accuracy on the test set over 20 runs of the experiments. Multiple runs are required due to the random initialization of weights described in Section 2. (The use of 20 in “20 runs” and “20 feature maps” is merely coincidental.) The classification accuracy was calculated by dividing the number of correctly classified images of a digit by the total number of images of that digit. For example. 99.54% accuracy means that on average, 513.63 images of digit 2 were correctly classified out of 516.

In this exploratory experiment, we noticed that increasing the receptive field size resulted in a “flip” phenomenon:

The accuracy for digit 2 increased, and the accuracy for digit 3 decreased!

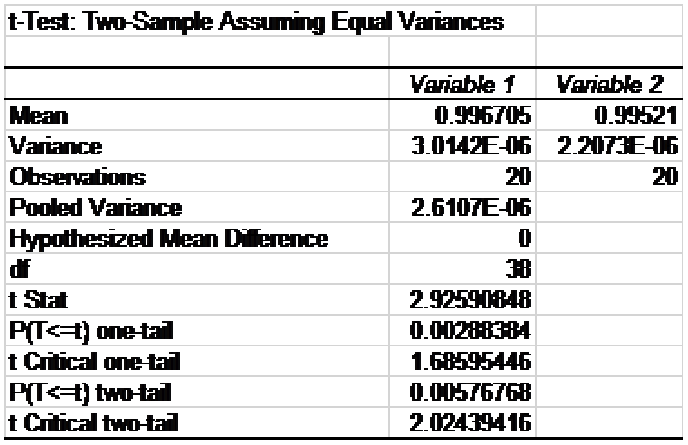
We performed a statistical test to compare the mean accuracies when the receptive field size increased for each digit. Because the sample standard deviations were of similar magnitude, we used the two-sample t-test assuming equal variances. The following was the result for digit 2, obtained from Excel's Data Analysis tool:



t-Test: Two-Sample Assuming Equal Variances		
	Variable 1	*Variable 2*
Mean	0.99544	0.997585
Variance	5.3773E-06	1.975E-06
Observations	20	20
Pooled Variance	3.6761E-06	
Hypothesized Mean Difference	0	
df	38	
t Stat	-3.53778141	
P(T<=t) one-tail	0.00054131	
t Critical one-tail	1.68595446	
P(T<=t) two-tail	0.00108262	
t Critical two-tail	2.02439416	

Figure 6.1: t-test result for comparing mean accuracies of classifying digit 2 when the receptive field size is 5 (Variable 1) and when the receptive field size is 7 (Variable 2).

Here, Variable 1 represented the accuracies when receptive field size was 5, and Variable 2 represented the accuracies when receptive field size was 7. In using the t-test, the null hypothesis assumed digit 2's mean classification accuracies were equal when the receptive field size was 5 and 7. A small p-value (0.00054131) of less than 0.001 indicated very strong evidence against the null hypothesis, meaning the difference in accuracies when the receptive field size was 5 and 7 was statistically significant. We obtained a similarly small p-value (0.00288) for digit 3:



t-Test: Two-Sample Assuming Equal Variances		
	Variable 1	*Variable 2*
Mean	0.996705	0.99521
Variance	3.0142E-06	2.2073E-06
Observations	20	20
Pooled Variance	2.6107E-06	
Hypothesized Mean Difference	0	
df	38	
t Stat	2.92590848	
P(T<=t) one-tail	0.00288384	
t Critical one-tail	1.68595446	
P(T<=t) two-tail	0.00576768	
t Critical two-tail	2.02439416	

Figure 6.2: t-test result for comparing mean accuracies of classifying digit 3 when the receptive field size is 5 (Variable 1) and when the receptive field size is 7 (Variable 2).

Therefore, we noticed that *the "flip" phenomenon was very statistically significant!*

In addition to the t-test, we also calculated the effect size to measure the difference between the accuracies when changing receptive field size from 5 to 7. The effect size quantifies how substantial the difference in accuracies in such "flip" phenomenon is. We used the popular Cohen's d formula to calculate the effect size:

$$\frac{M_1 - M_2}{\sqrt{\frac{SD_1^2 + SD_2^2}{2}}},$$

where M_1 and SD_1 are the mean accuracy and standard deviation when the receptive field size is 5, and M_2 and SD_2 are the mean accuracy and standard deviation when the receptive field size is 7. Results showed that the difference in mean accuracies when the receptive field size changed from 5 to 7 was substantial: 1.15 for digit 2 and 0.96 for digit 3. The above evidence might seem to show that the "flip" phenomenon was a very significant, substantial, and surprising discovery. But we had been cautious, doing the exploratory experiments using only half the available images, so that we could do separate "actual experiments". The later experiments did not show the same specific flip phenomenon at all.

We thus discovered a limitation to both statistical significance and Cohen's measure of effect size. Because there were only 516 images of digit 2 and 505 images of digit 3 in the test set of the exploratory experiments, the number of misclassified images was very small:

		Digit 2	Digit 3
Receptive field size 5	Average number of misclassified images	2.37	1.67
Receptive field size 5	Average number of misclassified images	1.24	2.42

A total of roughly 2 misclassified images and a difference of roughly 1 image when we increased the receptive field size from 5 to 7 for both digit 2 and 3 were, intuitively speaking, not big enough to guarantee an actual phenomenon. (Also, the legibility of the handwriting of the digits in the particular test set could have been a factor.) This is a downside of having a small number of misclassified images, because any discovered phenomenon could be misleading if one considers only statistical significance and effect size.

In our later experiments, described in Section 7, we strived to eliminate this problem by designing our scientific experiments to *avoid a small number of misclassified images.* We accomplished that by using a fairly large test set as well as a fairly small training set. Intuitively speaking, the large test set would mean that even high classification accuracy (but less than 100% accuracy) by the Machine Learning system would result in many misclassified images. Likewise, intuitively speaking, a small training set would cause the CNN to avoid achieving very high accuracy on the test set. While this design contrasts sharply with the standard *engineering* goal of CNNs (to achieve high classification accuracy), it helped us to conduct a *scientific* investigation of CNNs that could lead to better understanding of the CNNs' nature.

7 MNIST Experiments

As stated in Section 4, our main research question is:

> **Does the best choice of receptive field size for a CNN, with a single convolution layer and a single receptive field size, *depend on the specific application*, even in the presence of filtering and pooling?**

Recall from the Preface how scientists often choose the simplest possible experimental design to sharpen insight about a focused research question. With that in mind, we designed experiments even simpler than the normal MNIST classification challenge. Rather than having 10 different categories to classify, we had just 2. For example, for each of the 45 pairs of digits, so we would present just two different kinds of digits as input to the CNN, and the resulting binary classification task would be: "which of the two digits is the input image?" We called such experiments "one-versus-one" experiments, the focus of the current section. (Although this thesis does not present its results, we also preliminarily investigated, for each single digit, the binary classification task of determining whether or not the input was that single digit or a different digit regardless of what that other digit might be. We called such experiments "one-versus-others" experiments.)

To create the one-versus-one data sets for our experiments, we designed and followed this procedure:

- Step one: from the 70,000 images in the original MNIST, divide them into three disjoint data sets. DataSet 1 contains 30,000 images (3,000 images of each digit). DataSet 2 contains 30,000 images (3,000 images of each digit) that do not appear in DataSet 1. The rest of the images not in DataSet 1 and DataSet 2 are set aside.

- Step two: For the 30,000 images in DataSet 1, randomly divide them into three disjoint sets: 10,000 images in the training set (1,000 images of each digit), 10,000 images in the validation set (1,000 images of each digit), and 10,000 images in the test set (1,000 images of each digit). Do the same division for DataSet 2.

- Step three: For the 10,000 images in the training set of DataSet 1, randomly select 500 images (50 images of each digit), and discard the rest of the images in the training set. After this step, DataSet 1 contains 20,500 images in three sets as follow:

Training set	500 images	50 images of each digit
Validation set	10000 images	1000 images of each digit
Test set	10000 images	1000 images of each digit

Do the same random selection for DataSet 2.

Explanation for step three: As explained in the Preface, the goal of this thesis research is a scientific, rather than an engineering, goal. We seek a sharper understanding of the role of receptive field size and how that might depend on the specific classification application. Previous experiments we conducted showed that using a full training set of 10000 images enabled the resulting trained CNNs to perform very well on the test set, frequently reaching 99% to 100% classification accuracy (from 0 to a mere 20 misclassified images out of 2000 images in the test set). While such performance satisfies the usual engineering goal of using CNNs to obtain high accuracy, the number of misclassified images is not substantial enough to permit sharp scientific conclusions, for the important reason given in Section 6. Thus, we deliberately used a small number of images, just 50, in the training set to make the classification problem more challenging and thereby decrease the accuracy on the test set, potentially producing more substantial scientific results of the kind this research project seeks.

- Step four: For each set within DataSet 1 (training set, validation set, and test set), group images of each digit by pair. For example, in the training set, group 50 images of digit 0 and 50 images of digit 1 to create a new training set of 100 images, and then randomize the resulting set. Because there are 10 different digits, this results in C(10, 2) = 45 pairs of digits, representing 45 different "one-versus-one" data sets. Each of these 45 data sets contains:

Training set	100 images	50 images of each digit
Validation set	2000 images	1000 images of each digit
Test set	2000 images	1000 images of each digit

Do the same pairing for DataSet 2.

After creating the one-versus-one data sets, we performed 45 different types of exploratory experiments each using a corresponding one of the 45 data sets obtained from DataSet. Later we performed actual experiments on the 45 data sets obtained from DataSet 2.

7.1 Exploratory Experiments

We trained the CNNs on each of the 45 one-versus-one data sets in the exploratory experiments. On each of the 45 data sets, 14 different CNNs were trained, each having a different receptive field size that was an odd integer between 1 and 27 inclusive. (Recall that since our receptive fields were each square, for simplicity we give the receptive field size as the side length of the square.) Due to the nature of random initialization of weights (described in Section 2), we trained each CNN 20 times to get 20 different results and summarized them by calculating means and standard deviations of the classification accuracies on the test set.

Because the 45 classification problems might vary in difficulty, to act as a feasible way to factor out from consideration the various levels of difficulty of the applications, we added a normalization step before analyzing the results:

- There were 14 different mean accuracies for each application, since there were 14 different receptive field sizes that we tried. Out of the 14 accuracies, we chose the best mean accuracy and (to facilitate a normalization process) considered that to be the best accuracy the CNN could achieve.
- For each application, we wanted to find the best corresponding receptive field size. Therefore, for each application, we normalized its 14 mean accuracies relative to the highest mean accuracy out of the 14. For example, if the 2-versus-4 application had a highest accuracy of 98% with receptive field size 5, and also had an accuracy of 95% with receptive field size 11, the 98% would be "normalized" to 100%, and the 95% would be "normalized" to 95% / 98% = 96.94%.

The following is a plot of the results. The plot shows 45 curves, one for each application. Each curve shows the height of the 14 normalized mean accuracies for the curve's corresponding application.

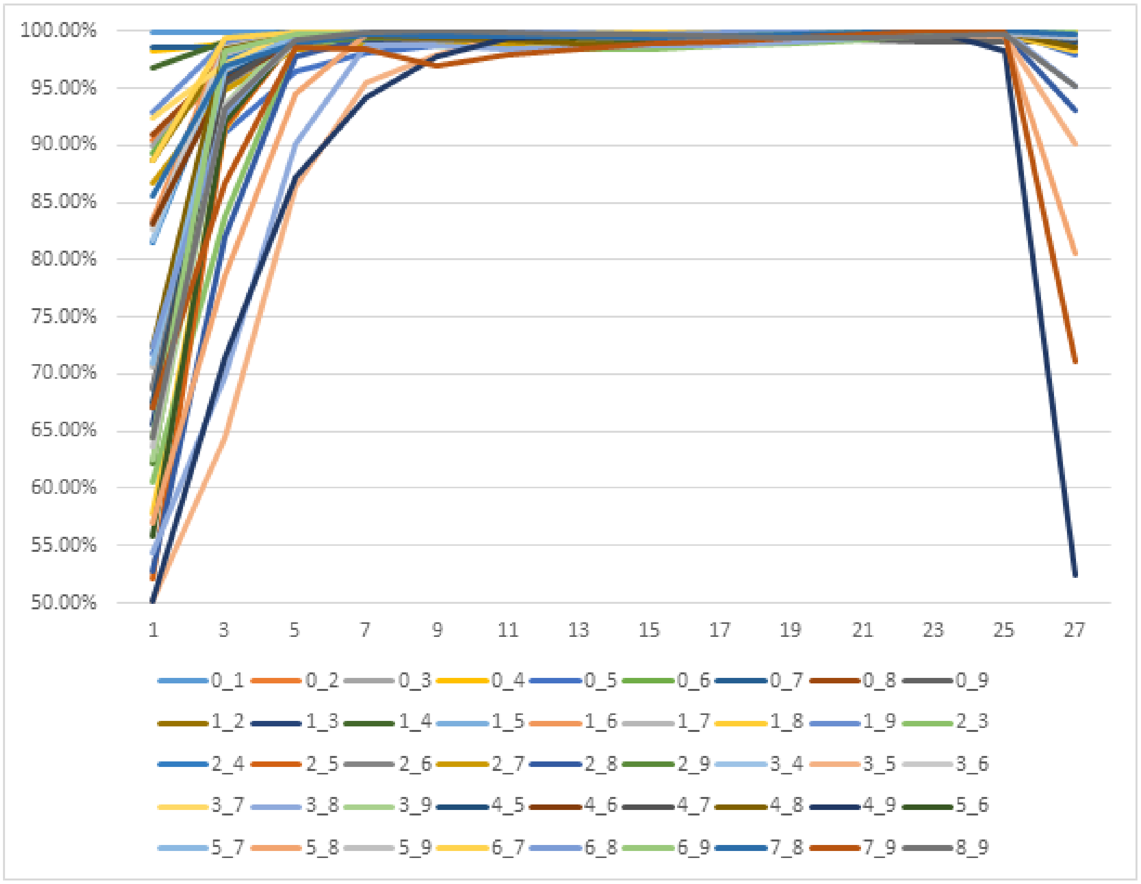


Figure 7.1: Normalized test set accuracies for all 45 applications.

The values on the y-axis are the normalized classification accuracies. The values on the x-axis are the 14 different receptive field sizes used in our experiments.

From the results, we observed:

- For all applications, the normalized accuracies had an upward trend, or at least non-decreasing, as the receptive field size increased, until receptive field size 25 was reached.
- The two applications 3-versus-5 and 4-versus-9 required large receptive field sizes, 11 or larger, to achieve normalized accuracies close to 100%.

- Meanwhile, the three applications 0-versus-1, 0-versus-4, and 0-versus-7 had close-to-highest normalized accuracies even with receptive field size 1.

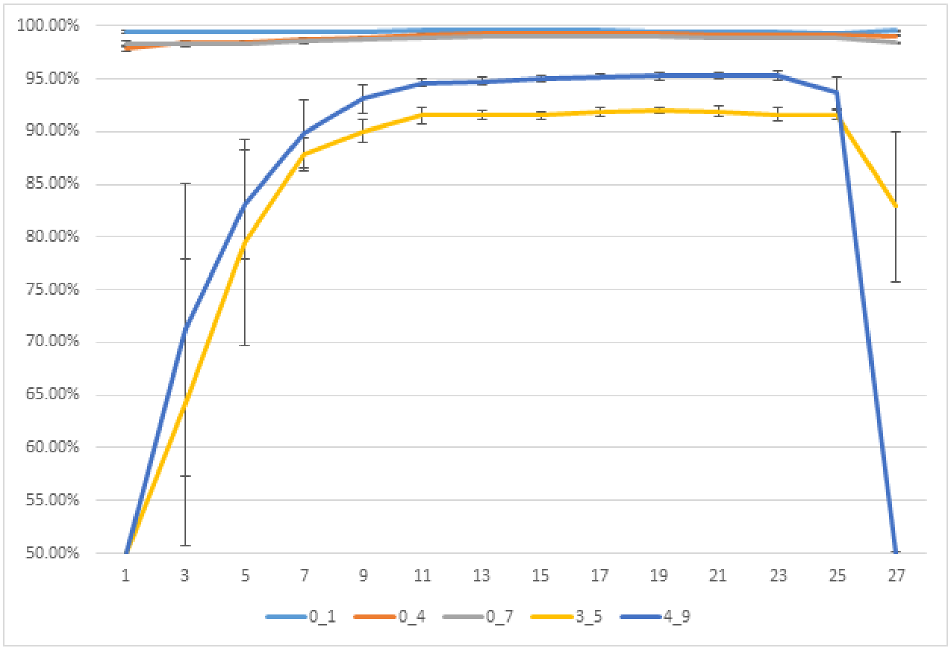


Figure 7.2: Un-normalized test set accuracies of five selected applications. The error bars show one standard deviation above and one standard deviation below the mean accuracies.

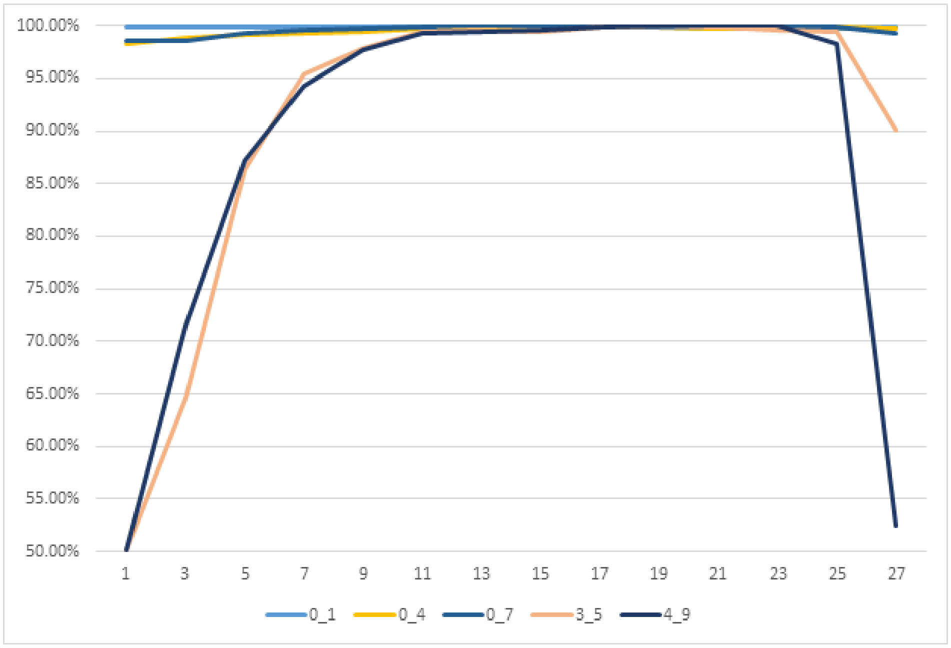


Figure 7.3: Normalized test set accuracies of the five applications in Figure 7.2.

Observations: Even though our CNNs used 20 feature maps and max pooling, which could influence the effect of receptive field sizes, we observed a phenomenon similar to the intuition described in Section 4.

- For the three applications 0-versus-1, 0-versus-4, and 0-versus-7, the digits in each pair share no apparent similarity. This might allow the CNNs to classify well with only a small receptive field size, if the intuition is correct.
- For applications 3-versus-5 and 4-versus-9, a bigger receptive field size might be required because the digits in each pair share some similar features, illustrated in Figure 7.4.

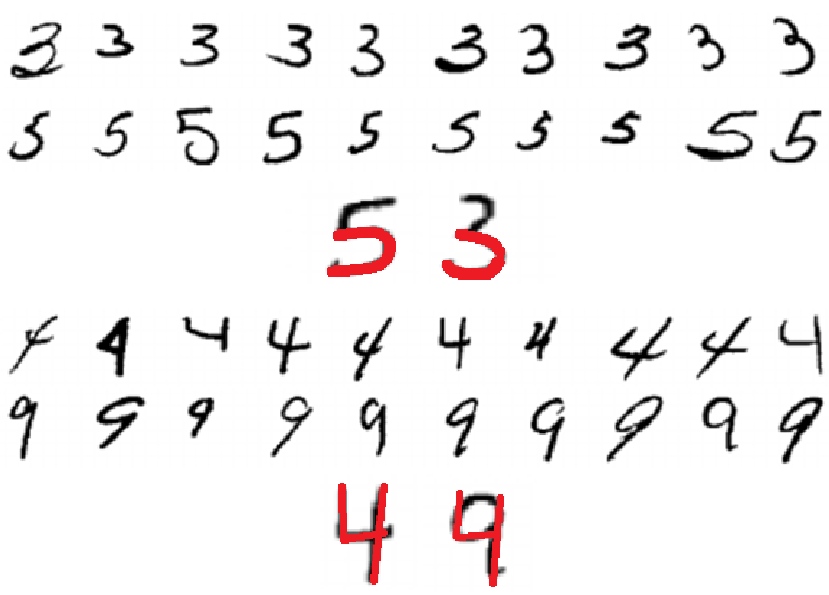


Figure 7.4: Example of several images of handwritten 3, 5, 4, and 9 from MNIST. The red curves/lines show similar patterns between 5 and 3, and 4 and 9.

Section 7.2 discusses the same types of experiments as we did in our exploratory experiments, but on a (totally) disjoint data set. The purpose was to confirm if our observations were robust.

7.2 Actual Experiments

We used the 45 one-versus-one data sets obtained from data set 2, as mentioned at the beginning of Section 7, and performed the same experiments as described in Section 7.1. After normalization, we obtained the following results:

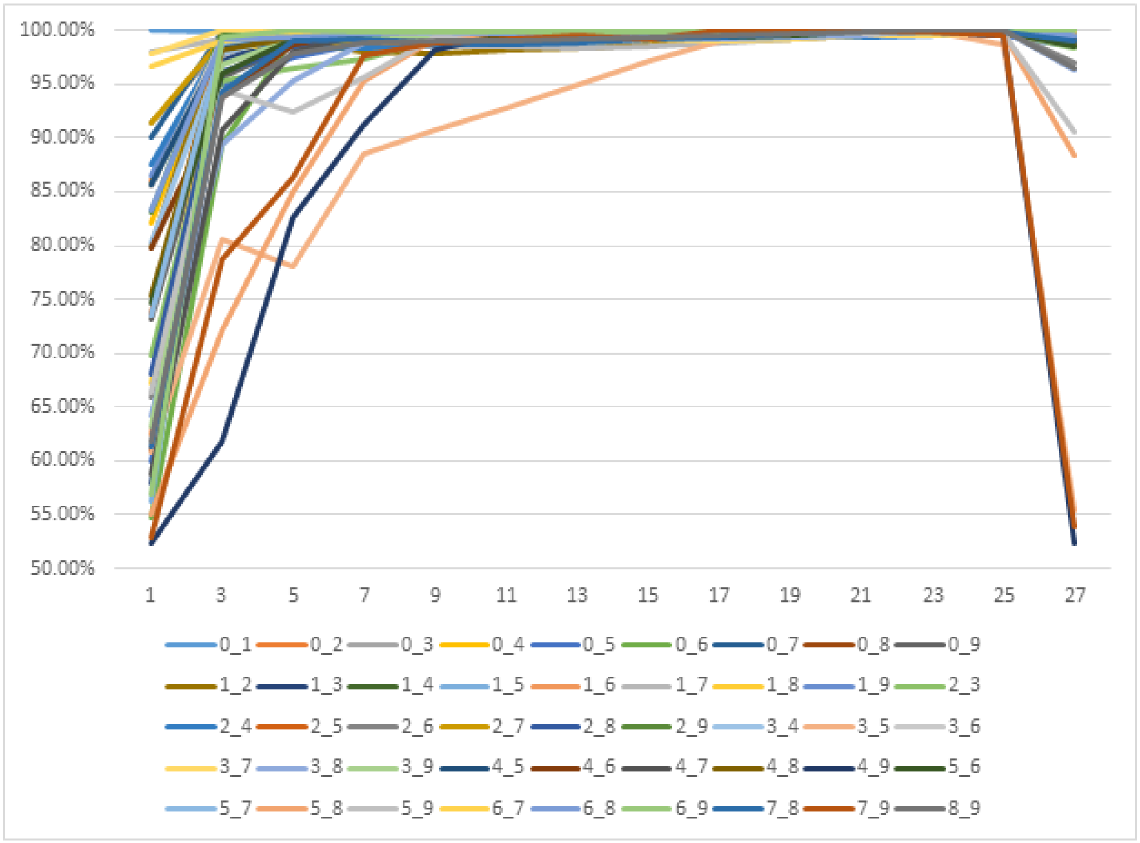


Figure 7.5: Normalized test set accuracies for all 45 applications.

Each curve in the graph above represents the normalized mean test set accuracy on a one-versus-one data set. The values on the y-axis are normalized classification accuracies. The values on the x-axis are the 14 different receptive field sizes.

We observed the following:

- Again, for all applications, the accuracies had an upward trend as the receptive field size increased, and only stopped at receptive field size 25.

- Again, the two applications 3-versus-5 and 4-versus-9 required large receptive field sizes, 11 or larger, to achieve normalized accuracies close to 100%. In addition, the application 7-versus-9 also required large receptive field sizes, 13 or larger, despite only needing receptive field size 5 to achieve a good normalized accuracy in the exploratory experiment.

- Again, the application 0-versus-1 had close-to-highest normalized accuracies even with receptive field size 1. Both applications 0-versus-4 and 0-versus-7 now required receptive field size 3 to perform well, which was still a small receptive field size.

- The three applications 1-versus-7, 3-versus-7, and 6-versus-7, which previously needed at least receptive field size 3 to reach good normalized accuracy in the exploratory experiment, now only needed receptive field size 1. We can see no similarities between 3 and 7, and 6 and 7, but one can argue that 1 and 7 share a similar straight line pattern.

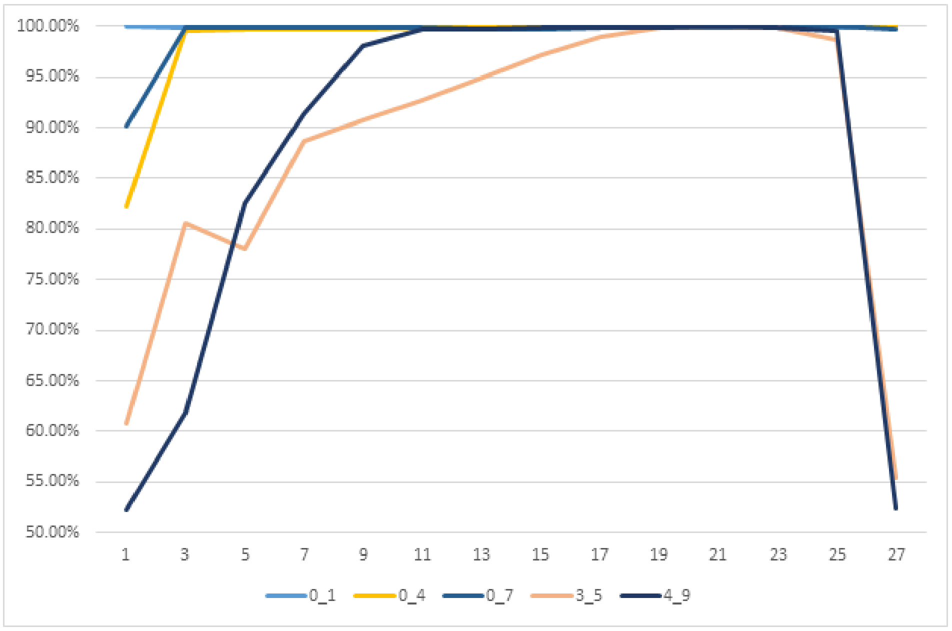


Figure 7.6: Normalized test set accuracies for the selected five applications.

Because of random initialization of weights, we can somewhat explain the difference in 7-versus-9 results between the exploratory and actual experiments, and how 1-versus-7 only requires a small receptive field size. However, 0-versus-1, 0-versus-4, 0-versus-7, 3-versus-5, and 4-versus-9 show consistent results between the experiments. Given these similar observations on a completely different data set, we have evidence suggesting that: when an application has “very non-similar” (not defined in this thesis) images, a small receptive field size is sufficient (even in the presence of 20 feature maps and max pooling) to reach a classification accuracy close to the best accuracy for the application; in contrast, when an application involves images that share some similarities, a large receptive field size appears to be required. Recall the intuition related to our main research question, stated in Section 4.

8 Topics for Future Research

- Hyper-parameter choices cannot usually be made independently of one another. But, for instance, if there are 12 hyper-parameters to choose and the range of choice for each is individually limited to just 10 choices, then there are 10^{12} (which is a trillion) choices among the hyper-parameters, and each such choice should perhaps be supported by at least 20 experiments, and that would need to be multiplied by the number of different applications examined, to experimentally investigate the relation between choices of hyper-parameters and the Machine Learning application. Experimental results obtained using CNNs whose hyper-parameter values are randomly chosen could lack sufficient clarity, since there could be undesirable high variance within such results due to factors irrelevant to our main research question. To be tractable experimentally, simplifying restrictions seem necessary. The current research has investigated receptive field size independently of all other hyper-parameters, where those other hyper-parameters had been chosen by another investigator (Michael Nielsen) to achieve relatively high accuracy. It might intuitively seem that, among all the hyper-parameters, the choice of number of filters and the choice of pooling size might be most interdependent with the choice of receptive field size. A somewhat simple new investigation (requiring perhaps only 11 times as many experimental runs rather than a trillion or more times as many) could address the following question: For each receptive field size, what is the effect on normalized accuracy of reducing the number of filters from 20 down to 0 (using a step size of 2 from one filter size to the next)?

- Experiments for the one-versus-one task need to be redesigned to provide even more robust substantial (in terms of actual number of misclassified images) as well as statistically significant results.
- Our preliminary one-versus-others experiments, described briefly and parenthetically earlier in Section 7, need to be similarly refined.
- We have also done preliminary experiments with handwritten capital letters, rather than digits. Those experiments need to be similarly refined.
- What is the best way to define the phrase "best choice" in our main question?
- If training time per application was limited to a particular amount of time, would that provide sharper insight?
- How might one devise a metric that considers both training time and some notion of accuracy?
- Could image "similarity" be precisely defined in a useful way, independent of using a particular Machine Learning methodology?
- An ultimate goal would be not just a yes/no answer to our main research question, but (if the answer is "yes") answers to questions such as (in a scientific sense): precisely what is it about an application that requires a larger receptive field? A better understanding of that kind of question might ultimately lead to an engineering improvement in which the receptive field size for an application could be chosen without requiring constant trial-and-error Machine Learning.

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, H. Lin, *Learning From Data*, AMLBook, 2012.

[2] A. Coates, H. Lee, A. Ng, “An Analysis of Single-Layer Networks in Unsupervised Feature Learning”, *Journal of Machine Learning Research*, 2011.

[3] Y. LeCun, C. Cortes, C. J. C. Burges, *The MNIST database of handwritten digits*. URL: http://yann.lecun.com/exdb/mnist/

[4] Y. LeCun et al., “Gradient-Based Learning Applied to Document Recognition”, *Proceedings of the IEEE*, 1998.

[5] C. Liu et al., “Handwritten digit recognition: Benchmarking of state-of-the-art techniques”, *Pattern Recognition*, 2013.

[6] National Institute of Standards and Technology, *NIST Special Database 19*. URL: https://www.nist.gov/srd/nist-special-database-19.

[7] M. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[8] R. Quiza, J. P. Davim, “Computational Methods and Optimization”, *Machining of Hard Materials*, 2011.

[9] Stanford University, *CS231n: Convolutional Neural Networks for Visual Recognition*, 2017. URL: http://cs231n.github.io/convolutional-networks/.

[10] United States Postal Service, *Postal Facts 2016*, 2016. URL: https://about.usps.com/who-we-are/postal-facts/postalfacts2016.pdf.