KATHFORD INTERNATIONAL COLLEGE OF ENGINEERING AND MANAGEMENT

Balkumari, Lalitpur

A

Major Project Report

on

"Optical Alpha-Numeric Character Recognition using Convolutional Neural Network"

[Subject Code: CT 755]

A PROJECT REPORT

SUBMITTED TO THE DEPARTMENT OF COMPUTER AND ELECTRONICS & COMMUNICATION ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Project Members

Achalshail Khadka (04/BCT/2072)

Bharat Awasthi (08/BCT/2072)

Bikhyat Adhikari (13/BCT/2072)

Sangam Basnet (33/BCT/2072)

DEPARTMENT OF COMPUTER AND ELECTRONICS & COMMUNICATION ENGINEERING

LALITPUR, NEPAL

AUGUST, 2019

KATHFORD INTERNATIONAL COLLEGE OF ENGINEERING AND MANAGEMENT

(Affiliated to Tribhuvan University)

Balkumari, Lalitpur

"OPTICAL ALPHA-NUMERIC CHARACTER RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK"

A PROJECT REPORT

SUBMITTED TO THE DEPARTMENT OF COMPUTER AND ELECTRONICS & COMMUNICATION ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted By:

Achalshail Khadka (04/BCT/2072)

Bharat Awasthi (08/BCT/2072)

Bikhyat Adhikari (13/BCT/2072)

Sangam Basnet (33/BCT/2072)

Submitted To:

DEPARTMENT OF COMPUTER AND ELECTRONICS & COMMUNICATION ENGINEERING

AUGUST, 2019

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who have helped us by any means in order to complete this project. We would also like to acknowledge with much appreciation the crucial role of our supervisor, **Er. Santosh Giri,** Lecturer, Department of Computer And Electronics & Communication, Kathford International College of Engineering & Management, who supervised us and continuously gave us the feedback and suggestions regarding the project.

A special gratitude we give to our project coordinator, **Er. Jalauddin Mansur,** Lecturer, Department of Computer And Electronics & Communication, Kathford International College of Engineering & Management, whose contribution in stimulating suggestions and encouragement, helped us to coordinate and complete our project.

Last but not least, many thanks go to lecturers of our department who have always been a source of inspiration to us and have been guiding us from beginning of the project to completing this report.

ABSTRACT

Optical Character Recognition (OCR) in the scanned documents has been a well-studied problem in the past. However, when these characters come from the real-world problem, it becomes much more challenging, as there exist many difficulties in these images, e.g.: illumination variance, cluttered backgrounds, and geometry distortions. This project has used a deep learning method based on convolution neural network to recognize the alpha-numeric characters in the scene images. The primary focus of this project was, how to make a compact architecture based on the convolution neural network in order to recognize the characters on the natural scene images. Rigorous experiments were conducted on 175,198 image samples from Kaggle dataset [1] in order to train the network. The data was divided into training, validation and testing sets: 10 % of data was used for testing and 90 % of data was used for training and validation. Among the training and validation set, 85 % was used for training and remaining 15 % was used of validation. Parameters used for training the model were epoch: 30, learning rate: 0.001 and batch size: 256. After training the model, training and validation accuracies were 96.03 % and 91.66 % respectively. After training and validation, the model was tested with the testing dataset and testing accuracy obtained was 85.49 %.

Keywords: OCR, illumination variance, CNN, Adam optimizer

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Program Interface
CNN	Convolution Neural Network
CPU	Central Processing Unit
GPU	Graphical Processing Unit
GUI	Graphical User Interface
IDE	Integrated Development Environment
KNN	K-Nearest Neighbor
LM	Levenberg Marquardt
ML	Machine Learning
NN	Neural Network
OCR	Optical Character Recognition
PDF	Probability Distribution Function
PNN	Probability Neural Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network

1. INTRODUCTION

1.1 Background

Optical Character Recognition (OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text whether from a scanned document, a photo of a document, a scene-photo or from subtitled text superimposed on an image. It is the process of translating images of handwritten, typewritten, or printed text into a format understood by machines. It allows for reduced storage size, editing, indexing, searching, etc. [2]. In this digital era, millions of organizations around the world are still using the traditional paper-based approach for day to day tasks because of having all the past data in the handwritten format. If there was some system to store all the previous data inside the computers in computer readable, sortable and searchable format, then all those organizations would have been digitalized.

1.2 Overview

OCR is a very well-known problem since past two decades. The lack of an efficient OCR system is one of the major reason that millions of the organizations are still unable to digitalize them. If there was an efficient OCR system, most of the organizations would have already stored their past data in computer readable, sortable and searchable format. Several hand filled forms could have been saved in the efficient format inside a computer.

There has been lots of researches and developments in the field of OCR but still, the researchers and the developers have not been able to get the satisfactory accuracy when it comes to recognizing the hand-written text. However, several systems have been developed that are even more than 97% accurate in recognizing the computer printed documents. But they still lag far behind when it comes to recognizing the hand-written texts.

The major reasons that make OCR such a challenging problem are the larger size of document in image format, the noise, distortions and the illuminance variations in the captured image, several languages, different people having different handwritings, the language lexicons, the requirement of high processing costs, etc.

1.3 Problem Statement

Optical Character Recognition falls under Machine Learning and Natural language Processing and has not been so easy till the date due to several complications due to the fact that there are several symbols, characters, lexicons, etc. for a language and everyone has their own style of writing. Moreover, the complications in image itself like the pixel densities, gradient, texture, color etc. makes the character recognition a difficult task. Although there are lots of software developed for the optical character recognition, they provide much less accuracy when it comes to recognizing the handwritten texts due to the fact that everyone has their own handwriting and it's very hard to model each of them.

1.4 Objectives

1.4.1 General Objective

- To develop a system that can recognize the handwritten text in image and to convert the characters in computer readable, searchable and sortable format.

1.4.2 Specific Objectives

- To download the suitable training dataset from online sources and preprocess it for training the machine learning model
- To develop the GUI
- To implement the convolution neural network and train it with the collected training dataset
- To evaluate the performance measurement of implemented CNN
- To integrate all the components of the system and perform integration testing

1.5 Scopes/Applications

OCR is a very well-known problem and has scopes in a variety of fields. One of the major applications of OCR is recording the handwritten data of an organization in a computer in an efficient way. It can also be used for tracing the vehicles from CCTV of the traffic by identifying the number plates. It can be used to scan the handwritten forms and documents and store it in the machine in text format. It can also be integrated with a bank's system to process the cheque automatically without the involvement of human. This system can be extended to be used in post office to scan the zip codes and addresses.

2. LITERATURE REVIEW

2.1 Related Work

The authors of [3] had used Hough transformation technique for Arabic characters recognition. They were able to get about 95% accuracy. They had used Hough transformation for the feature extraction and dynamic programming matching for the classification. The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

Subhash Tatale and Akhil Khare [4] have used KNN (K-Nearest Neighbor) method for recognizing the number plates of vehicles. In pattern recognition KNN is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether KNN is used for classification or regression:

- In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In KNN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

Neural networks with centroid dithering can be implemented to build a high accuracy OCR application [5]. Centroid dithering process involves dithering the centroid of two dimensional input image.

Richard D.Romero, David S.Touretzky and Robert H.Thibadeau in [6] have introduced the use of probabilistic neural networks for optical character recognition. A PNN is a feed forward neural network, which is widely used in classification and pattern recognition problems. In the PNN algorithm, the parent probability distribution function (PDF) of each class is approximated by a Parzen window and a non-parametric function.

According to S. Varshney [7], AI enables machines the human like abilities and human aptitude based intelligence, it has remained one of the most challenging areas in last few decades. Giving machine the power to see, interpret and the ability to read text is one of the major tasks of AI. A lot of work has been done in this field, but still the problem is not solved in its full complexity. A good text recognizer has many commercial and practical applications, e.g. from searching data in scanned book to automation of any organization, like post office, which involve manual task of interpreting text.

2.2 Related Theory

2.2.1 Different Approaches for OCR

The problem of text recognition has been attempted by many different approaches. Template matching is one of the simplest approaches. In this many templates of each word are maintained for an input image, error or difference with each template is computed. The symbol corresponding to minimum error is output. The technique works effectively for recognition of standard fonts, but gives poor performance with hand written characters.

Feature extraction is another approach, in which statistical distribution of points is analyzed and orthogonal properties extracted [7]. For each symbol a feature vector is calculated and stored in database. And recognition is done by finding distance of feature vector of input image to that of stored in the database, and outputting the symbol with minimum deviation. Though this technique gives lot better results on handwritten characters but is very sensitive to noise and edge thickness.

In geometric approach, on the other hand, attempt is made to extract features that are quite explicit and can be very easily interpreted. These features depend on physical properties, such as number of joints, relative positions, number of end points, length to width ratio etc. Classes forced on basis of these geometric features are quite distinct with not much overlapping. The main drawback of this approach however is that this approach depends heavily on the character set, Nepali alphabets or English alphabets or Numbers etc. Features extracted for one set are very unlikely to work for other character set. In contrast to this, neural networks offer complete independence of recognition process and character set. In this neural network is first trained by multiple sample images of each alphabet. Then in recognition process, the input image is directly given to neural network and recognized symbol is outputted. The advantage of neural

networks lies in the domain of the character set that can be very easily extended; one just needs to train the network over new set. Another advantage is that neural networks are very robust to noise. The disadvantage is that a lot of training is required which is very time consuming.

None of the above approaches thus used in their pure form yields good result on handwritten text. So generally hybrid approaches are taken to achieve the desired recognition rates. Although even best of the recognition approaches are able to recognize all those texts that human can. Major reason for this has been wide variation in writing practices and styles of different people and lot of context based information that human brain uses to interpret any text sample. Multi-layer feed forward neural network with back propagation training algorithm can be implemented to recognize the characters with higher accuracy. Gradient descent algorithm and Levenberg Marquardt algorithm can be used and the results can be compared.

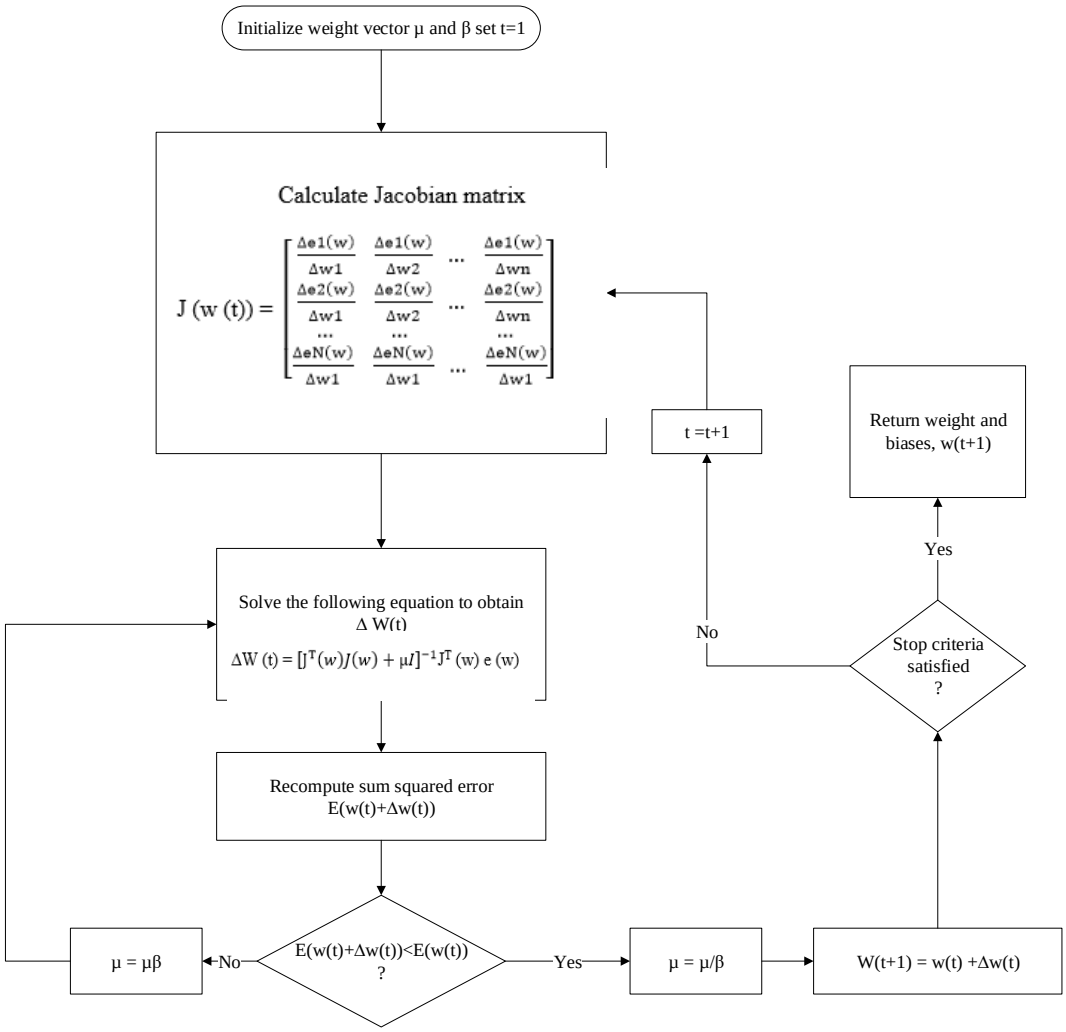

Figure 2.1: Flow chart of LM method

2.2.2 Machine Learning

Machine learning provides computers the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn themselves. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. Machine learning algorithms build a mathematical model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to perform task. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on provided examples. Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

2.2.2.1 Supervised Learning

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output: $Y = f(X)$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems.

- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

According to [8] some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

2.2.2.2 Unsupervised Learning

Unsupervised learning is the training of an ML algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. In unsupervised learning, an ML system is presented with unlabeled, uncategorized data and the system's algorithms act on the data without prior training. The output depends upon the coded algorithms. Subjecting a system to unsupervised learning is one way of testing AI. Unsupervised learning algorithms can perform more complex processing tasks than supervised learning systems. However, unsupervised learning can be more unpredictable. While an unsupervised learning ML system might, for example, figure out on its own how to sort cats from dogs, it might also add unforeseen and undesired categories to deal with unusual breeds, creating clutter instead of order. Hierarchical clustering, Hebbian network, k-means, deep belief nets, etc. are some examples of unsupervised learning.

2.2.2.3 Semi-Supervised Learning

Semi-supervised learning is a class of machine learning tasks and techniques that also make use of unlabeled data for training; typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning resides somewhere in between supervised and unsupervised learnings since both labeled and unlabeled data are used for semi-supervised learning. It is much useful in cases when there isn't enough labeled data to produce an accurate model. Generative models and Heuristic approaches are some examples of semi-supervised learning.

2.2.3 Artificial Neural Network

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites. ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value. Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values [9].

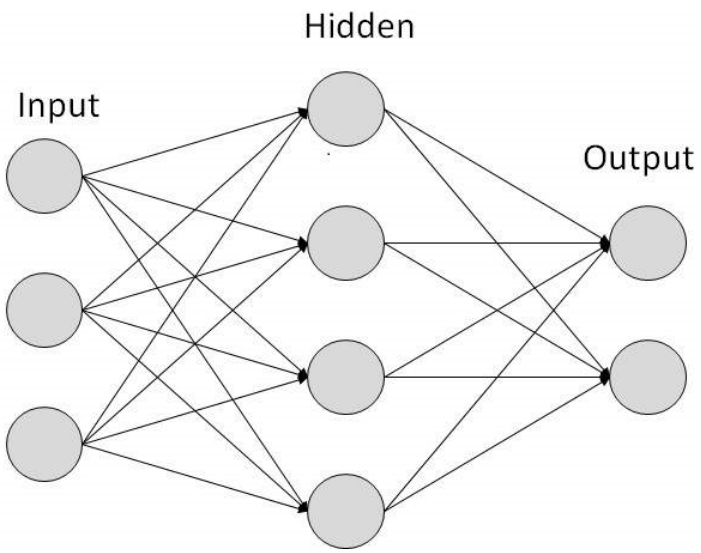

Figure 2.2: Single layer artificial neural network

2.2.4 Error Backpropagation Method

Back propagation is the essence of NN training. It is the practice of fine-tuning the weights of previous layer of neural net based on the error rate. Proper tuning of weights ensures lower error rates, making the model reliable by increasing its generalization. Back propagation is needed while designing an NN. In the beginning, weights and biases are initialized randomly and input is fed into the network. Then the deviation of the calculated output from the actual output is calculated as error. Then the error at output layer is used to adjust the weights and biases of the neurons at output layer. Then error at output layer is reflected back to the previous layer to adjust the weights and biases at previous layer. It is done in following steps:

- **Calculate the error and correct the parameters at output layer:**

  Calculate the deviation of model's output from actual output as error. Then error gradient of the neurons in output layer is calculated as:

  $$\delta_k = y_k(p)\,(1-y_k(p)).e_k(p)\text{ , where } e_k(p) = y_{d,k}(p) - y_k(p)$$

  Then the weight corrections at the output layer is calculated as:

  $$\Delta w_{j,k} = \alpha \,.\, y_j(p),\, \delta_k(p)$$

  Then the weight corrections are used to update the weights at output layer as:.

  $$w_{j,k}(p+1) = w_{j,k} + \Delta w_{j,k}(p)$$

  The whole process repeats until the error at output layer is below the defined maximum value of error that is acceptable.

- **Propagate the error back to the previous layer**:

The error gradient in the previous hidden layer is calculated as:

$$\delta_j(p) = y_j(p)[1-y_j(p)].\left[\sum_{k=1}^{I}\left(\delta_k(p).w_{j,k}(p)\right)\right]$$

Then the value of error gradient is used to calculate the weight corrections at the previous hidden layer. Weight correction is calculated as:

$$\Delta w_{i,j} = \alpha \,.\, x_i(p),\ \delta_j(p)$$

After calculating the weight corrections, the weights at the previous hidden layer are corrected as:

$$w_{i,j}(p+1) = w_{i,j} + \Delta w_{i,j}(p)$$

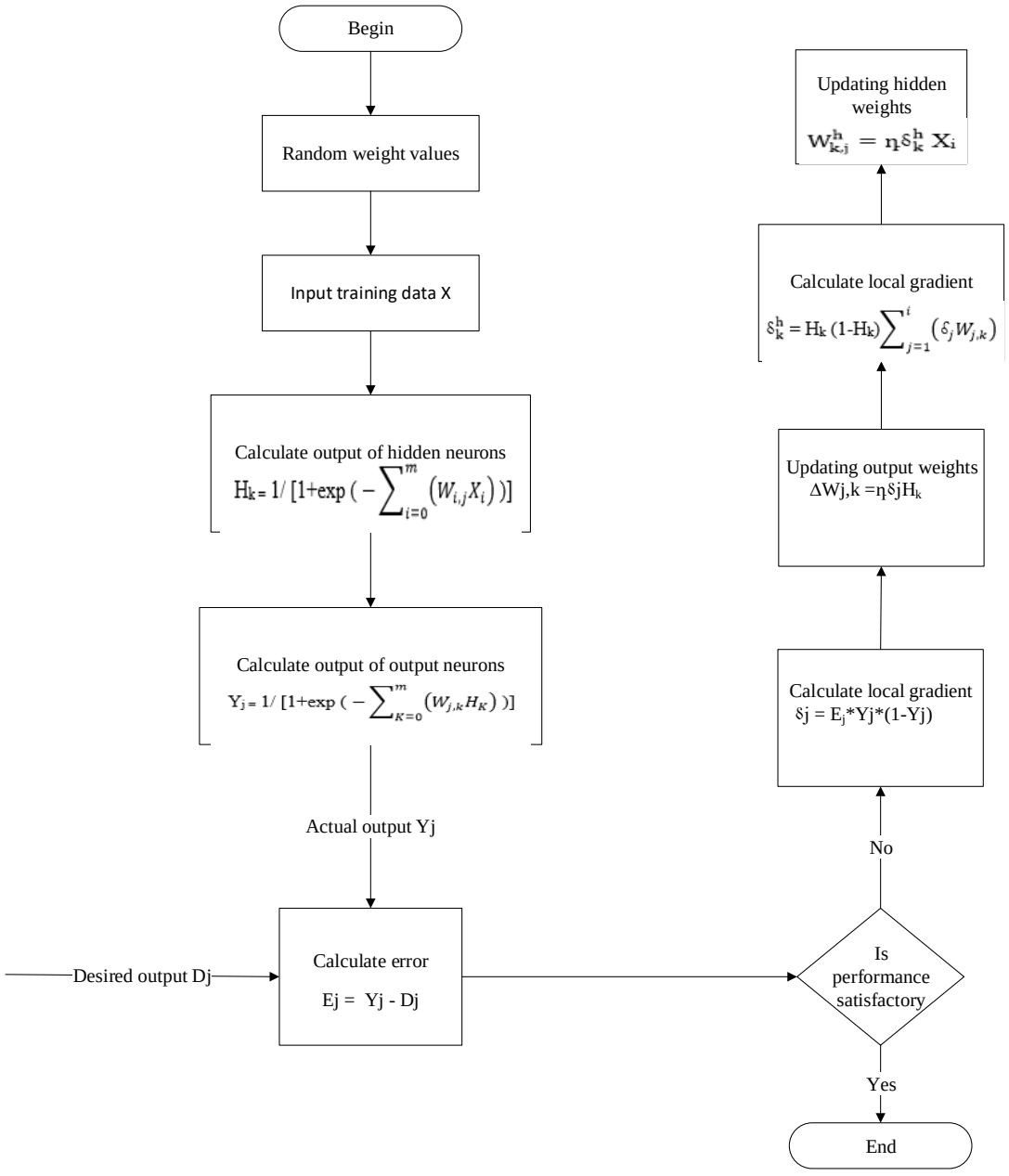

Figure 2.3: Flow chart of error back propagation

2.2.5 Convolutional Neural Network

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such field overlaps to cover the entire visual area [10].

Figure 3.2 below shows an example of a convolution neural network to classify handwritten digits. There are two convolution layers having 5x5 filters in each and followed by a flattened layer. There are ten neurons at the output layer to classify the digits from 0 to 9.

Source: towardsdatascience.com

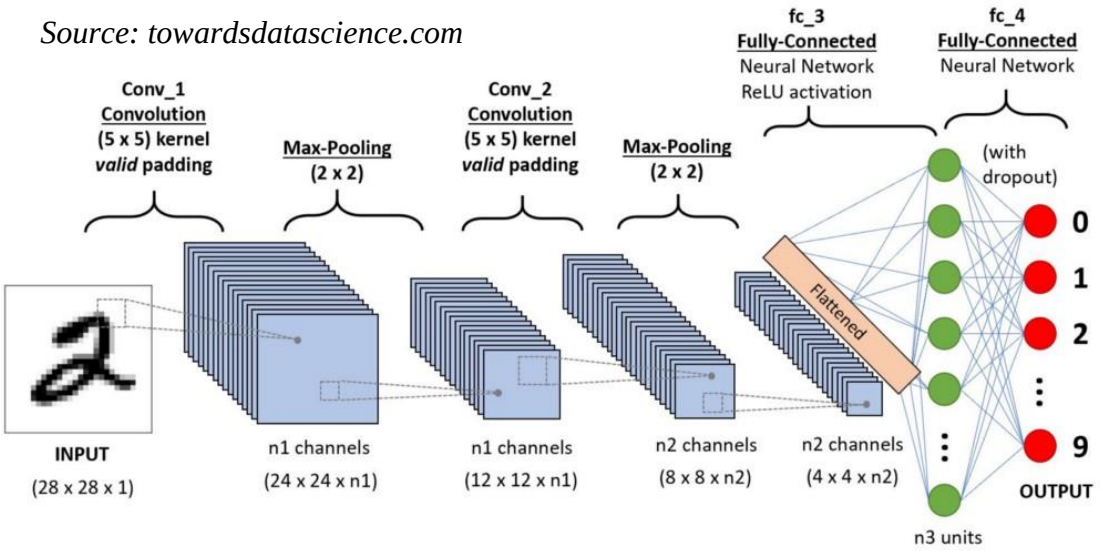

Figure 2.4: An example of convolution neural network

- **Kernels:**

  Kernels are also known as the filters. They are learnable parameters of Convolution Neural Network as the convolution neural networks fall under supervised learning. Each kernel has a bias associated with it. Kernels are

nothing but either 2D or 3D matrices of weights depending on either you are implementing a 2D Convolution or a 3D Convolution Network. Initially, these weight values are initialized randomly and are gradually learnt over several forward and backward propagations. Generally used kernels are square matrices of odd sizes like 1x1, 3x3, 5x5, 7x7 and so on.

- **Convolution Operation:**

  Convolution Operation is nothing but just the matrix dot multiplication. The idea is to take a kernel of same or smaller size than the input matrix and slide it over the input matrix and add the products from the elementwise multiplication of each elements of kernel and input. If same padding is used, then the result from the convolution operation has the same dimension as the input matrix. The mathematical formula for convolution operation has been shown in equation 3.1 below:

$$h^{n}_{i,j} = \sum_{k=0}^{k=w-1} \sum_{l=0}^{l=w-1} X_{i+k,j+l} \times W^{n}_{k,l} + b^{n} \quad \text{(3.1)}$$

  Where,

  n is the number of filters

  i is the number of rows in feature matrix

  j is the number of columns in feature matrix

  w is the size of the filter

  $h^{n}_{i,j}$ is the $(i,j)^{th}$ element of feature matrix for n^{th} filter

  W^{n} is the n^{th} filter matrix

  X is the input matrix

  b^{n} is the bias for n^{th} filter

- **Pooling:**

  The idea behind using pooling in convolution layer is that more features can be extracted along with reducing the input size. After convolving each kernels from a convolution layer to the input, all the resulting matrices are passed to a pooling layer. The pooling can either be average pooling or max pooling. If max pooling is used, then for each feature matrix from convolution layer, elements having maximum value within the defined window is selected. Even a 2x2 max pooling can reduce half the size of the feature matrix.

- **Rectified Linear Unit:**

  A Rectified Linear Unit or simply a ReLU is an activation function that eliminates the values lower than the defined minimum accepted value. It is used

in Convolution layer to introduce some nonlinearities to the feature matrices. Equation 3.2 and Figure 3.3 represent a ReLU function.

$$y = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \quad \text{(3.2)}$$

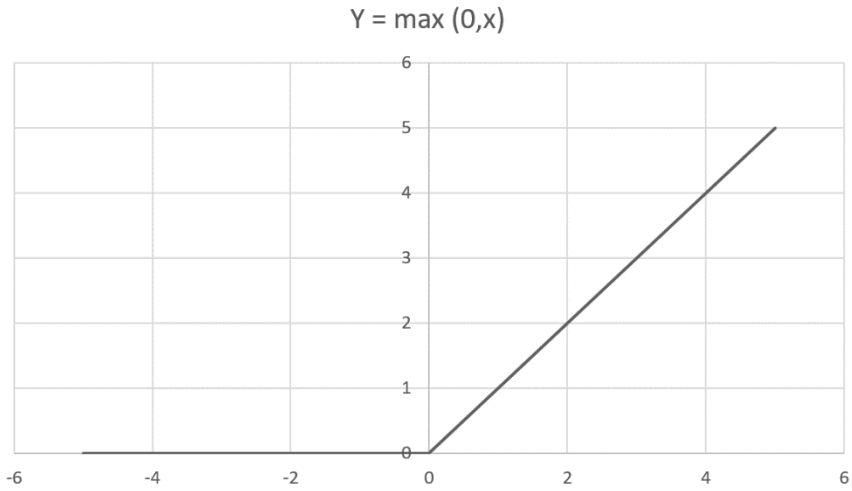

Figure 2.5: ReLU activation

Softmax:

Softmax is also known as softargmax or normalized exponential function. It is an activation function that is used at the output layer of a Convolution Neural Network. Its task is to represent the weight values of each of the neurons at the output layer in probabilistic form where sum of the probabilities for each of the neuron is one. In probability theory, the output of a softmax function can be used to represent a categorical distribution. The basic idea is to activate the output neuron that has the highest probability. It is widely used in several classification and regression problems.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=0}^{k} e^{y_j}} \; for\; i = 0,1,2 \ldots., k \quad \text{(3.3)}$$

2.2.6 Implemented Training Algorithm

Step 1: Read an image and its class

Step 2: Convert raw image to its pixel array

Step 3: Normalize each pixels: Normalized_pixel = Orignal_pixel/255

Step 4: Append the array of Normalized_pixel and class label to the dataset list

Step 5: Repeat step1 to step4 for all the images.

Step 6: Randomly initialize all weights and biases.

Step 7: Select a batch of size batch_size from dataset list

Step 8: Feed the batch to the 1st Convolution Layer

Step 9: Perform the convolution:

$$h^{n}{}_{i,j} = \sum_{k=0}^{k=w-1} \sum_{l=0}^{l=w-1} X_{i+k,j+l} \times W^{n}{}_{k,l} + b^{n}$$

Step 10: Use max pooling on the feature matrix with widow size (2,2)

Step 11: Apply ReLU activation on max pooled matrices:

$$y = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$$

Step 12: Feed the output features from 1st Convolution Layer to 2nd Convolution Layer

Step 13: Perform Step9 to Step11 for each output feature from 1st Convolution Layer

Step 14: Flatten all the output feature matrix from 2nd Convolution Layer

Step 15: Forward Pass to the Fully Connected Layer. For each neurons at Flatten Layer,

Calculate: $y^{i}_{j} = w^{i}_{jk}\, x_{k}{}^{i} + b_{j}$

Step 16: Again forward pass to the output layer

Step 17: Apply softmax activation at the output layer. $S(y_i) = \frac{e^{y_i}}{\sum_{j=0}^{k} e^{y_j}}$

Step 18: Calculate cross entropy loss:

$$\sum_{k=1}^{M} y_{o,k} \log_2(p_{o,k})$$

Step 19: Adjust the weights and biases.

Step 20: Propagate the error back to previous layer.

Step 21: Repeat step19 to step20 for all previous layers except for input layer.

Step 22: Repeat step8 to step21 for all next batches.

Step 23: Save the model to a json file format

Step 24: Save the weights and biases to an hdf5 file format.

Step 25 Stop

2.2.7 Technical Details

2.2.7.1 System Environment

The system was built, trained, validated and tested on a windows 10 home version 1803 OS build 17134.885 64 bit operating system based personal computer having intel core i5-6200 2.3GHZ CPU and 1700 MHZ 8 GB DDR3 RAM.

2.2.7.2 Python

Python is a general purpose high level object oriented machine independent programming language. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including object oriented, functional, and procedural programming. Several built in and third party libraries, packages and frameworks make python a powerful tool for scientific, ML, AI and web programming. Python is highly focused on readability of programs. The system was built with 64 bit Python 3.7.4 for windows.

2.2.7.3 Pycharm

Pycharm is an IDE used in computer programming specifically for python language. It provides code analysis, a geographical debugger, an integrated unit tester, integration with version control systems. The system was developed using 64 bit jetBrains PyCharm Edu 2019.1.1.

2.2.7.4 Numpy

Numpy is a general purpose array processing package in python. It provides a high performance multi-dimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with python. Numpy version 1.16.4 was used in the system for storing and processing one dimensional and multi-dimensional arrays to normalize the images and to store and read data to and from files.

2.2.7.5 Matplotlib

Matplotlib is a library for making 2D plots of arrays in python. It is primarily written primarily in pure python though it makes heavy use of numpy arrays and other extension code to provide good performance even for large arrays. Matplotlib version 3.1.1 was used in this system to plot images, filters, training and validation accuracy and loss graphs and the confusion matrix.

2.2.7.6 Scikit Learn

Scikit Learn is a library in python that provides many unsupervised and supervised learning algorithms. It is written in python and built on top of numpy, pandas and matplotlib. It is widely used in data mining and data analysis. The system has used scikit learn version 0.21.2 to calculate the confusion matrix during the model testing and to plot the confusion matrix for visualization.

2.2.7.7 Tkinter

Tkinter is the standard python interface for GUI programming. The name Tkinter comes from Tk Interface. Using tkinter provides fastest and easiest way to create the GUI applications. The GUI of the system has been built using tkinter.

2.2.6.8 Keras

Keras is a high level neural networks API, written in python and capable of running on top of Tensorflow. It allows for easy and fast prototyping, supports both CNN and RNN and runs seamlessly on CPU and GPU. The system has used keras version 2.2.4 to implement, train, validate and test the network.

3. PROJECT METHODOLOGY

3.1 Project Workflow

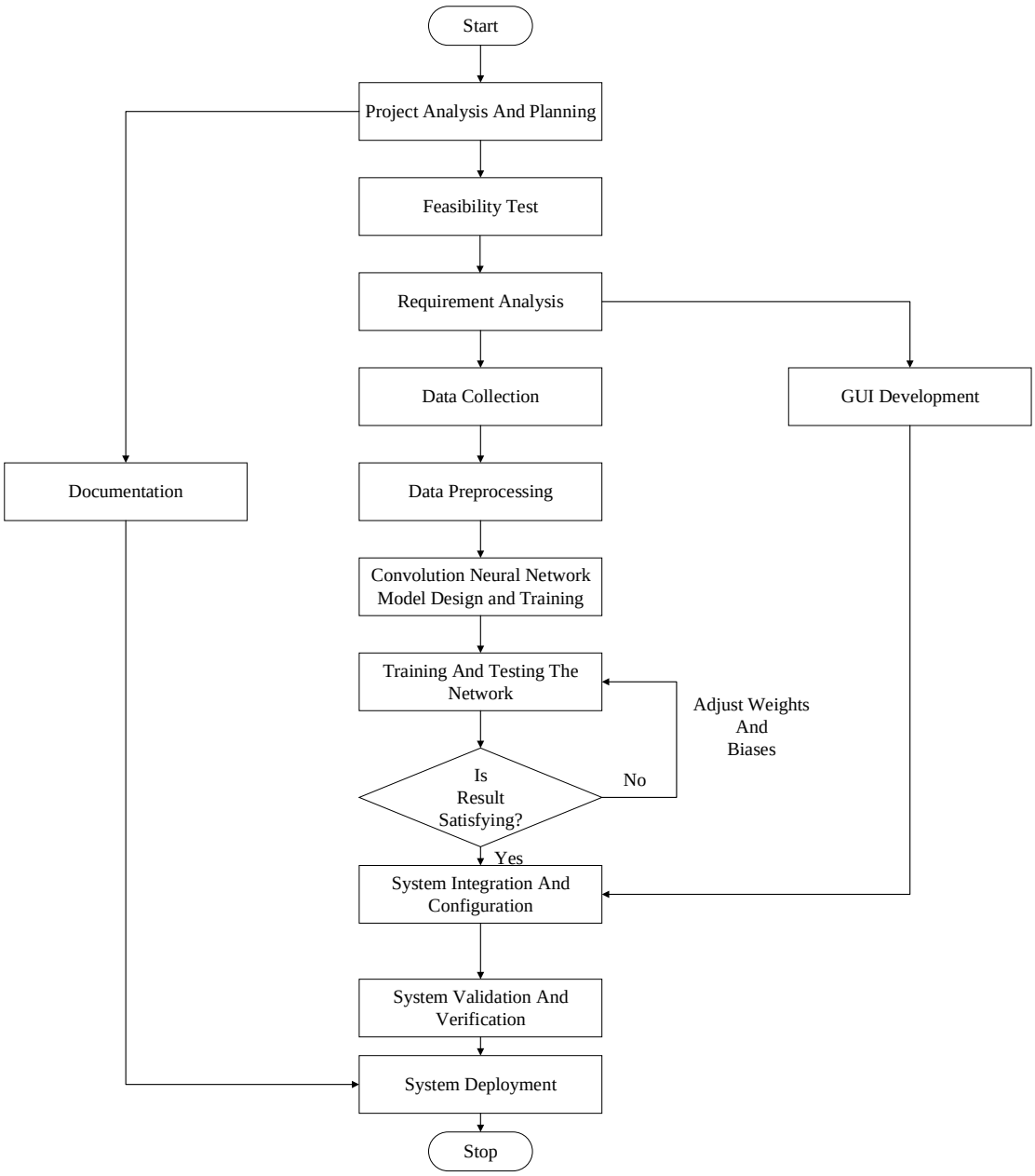

Figure 3.1: Project workflow diagram

- **Project Analysis and Planning:**

  This was the first step of the project. We started analysis of the project. We researched some papers and articles useful for the project. We did make a proper planning and schedule to complete the project. We created a Gantt chart to guide

us through the other phases of the project in order to stick with the time available with us to complete the project.

- **Feasibility Test:**

  During this step, we researched more whether doing this project was feasible or not. The project passed technical feasibility because it does not require any special kind of technologies. The project seemed to be legally feasible since it does not violate any laws and does not require any legal permissions. There was enough time to complete this project so the project passed the schedule feasibility. The project has high scope and it does not require any special skills to use the system and hence it passed the operational feasibility. The project did not require investment of high amount of money and hence passed the economic feasibility.

- **Requirement Analysis:**

  It was one of the major phase since requirements are the basic foundation to guide any project through the rest of the remaining phases. We formulated the problem and defined the requirement of our project during this phase.

- **System Design:**

  During this phase, the blueprint or the architecture was designed. We created the use cases and designed the use case diagram. The algorithm of the system was also developed. The block diagram of the system was also designed during this phase. And we finally designed the neural network model for our project.

- **GUI Development:**

  During this phase, we developed the GUI suitable for the project as well as user-friendly for the users. The major concern was the usability of the system.

- **Data Collection:**

  We searched for data on several online resources. The one from Kaggle seemed to be suitable for our project so we downloaded that data set for the project. However, the dataset was not balanced and it did need some preprocessing.

- **Data Preprocessing:**

  During preprocessing of the data set, we converted the available image data into a list 2D array of the pixel values ranging between 0 to 255 and the corresponding class labels. Background and foreground of the images were inverted and the pixel values were normalized in the range of 0 and 1.

- **Convolution Neural Network Implementation:**

  During this step, neural network was implemented in Python using some scientific and neural network libraries, packages and APIs. We converted the algorithm and the blueprint into the executable source code.

- **Training and Validating the Network:**

  During this step, the network was trained and validated with the preprocessed dataset. 85 % of the preprocessed dataset was used to train the model and rest 15 % of the data was used to validate the model. Adam optimizer was used to optimize the model by minimizing the cross entropy.

- **System Integration and Configuration:**

  During this step, all the components of the system were integrated to form a single program. The integration testing was also performed to ensure that the system as a whole works fine.

- **Documentation:**

  Documentation was initiated from the very beginning of the project. It was updated regularly at each phases of the project completion. The final result of the system for training, validation and testing were also recorded in the document.

3.2 Deployment Model

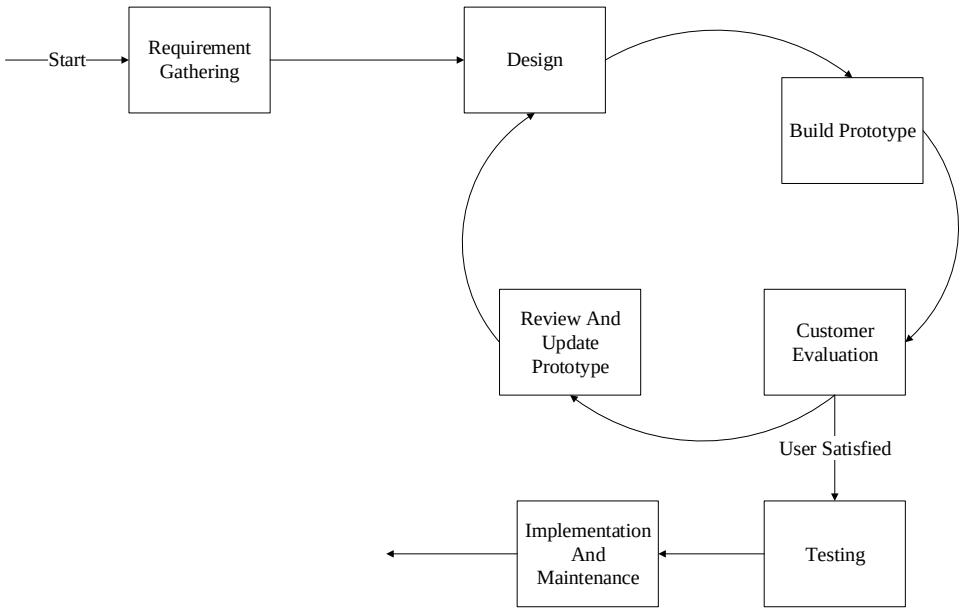

Figure 3.2: Prototype software development model

The system was developed using porotype software development model. The model has been shown in figure 4.2 above. It consists of several iterations refining the prototype.

During the requirement gathering, the requirement of the system was analyzed and required data set was collected from the online sources. Problem was formulated and the feasibility studies were performed for technical, operational, legal, economic and schedule feasibilities. The collected data set was preprocessed as per the need.

Then a quick design of the system was built. After designing the system, a working porotype was developed. Then it passed several iterations for evaluating, refining and reconstructing the prototype. After the satisfactory refined prototype, the prototype was engineered for the engineering product. The system was then built based on the final refined prototype

3.3 Architecture of The Implemented CNN

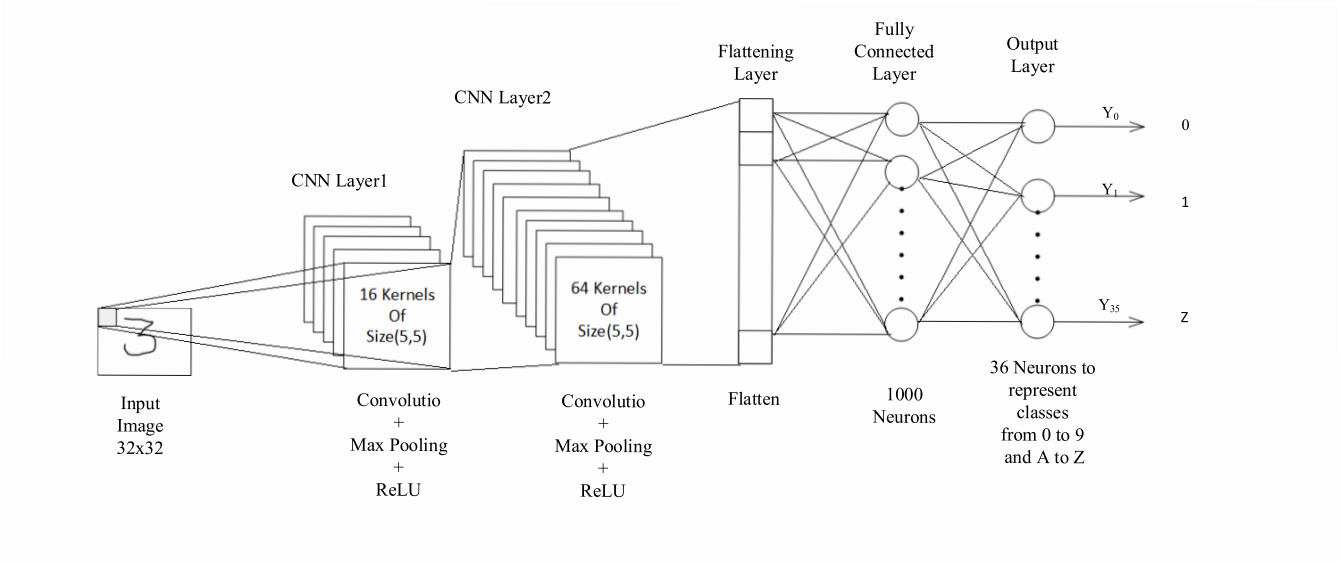

Figure 3.3: Architecture of the implemented CNN

The network has two convolution layers, one flatten layer, one fully connected layer and one output layer. Each convolution layer consists of convolution operation along with max pooling and ReLU activation. The first convolution layer has 16 kernels of size (5, 5) and the second convolution layer has 64 kernels of size (5, 5). In each convolution, valid padding is used with stride of size (1, 1). The max pooling has been used with window size of (2, 2). All the outputs from the second convolution layer are flattened in the flattening layer. The flattening layer is connected to the fully connected layer. This intermediate fully connected layer consists of 1000 neurons.

The activation from these neurons are fed to the neurons at output layer. There are 36 neurons at the output layer; representing one class each (0 to 9 and A to Z). At the output layer, the softmax function has been used to convert all the output values to the probability distributions.

So the number of parameters to be learned are:

At 1st convolution layer: $16 \times 5 \times 5 + 16 \; = \; 416$

At 2nd convolution layer: $16 \times 64 \times 5 \times 5 \; + 64 \; = \; 25{,}664$

At Dense Layer: $4096 \times 1000 + 1000 = 4{,}097{,}000$

At Output Layer: $1000 \times 36 + 36 = 36{,}036$

Total number of learnable parameters = 416 + 25664 + 4097000 + 36036

= 4,159,116

4. SYSTEM DESIGN

4.1 Block Diagram

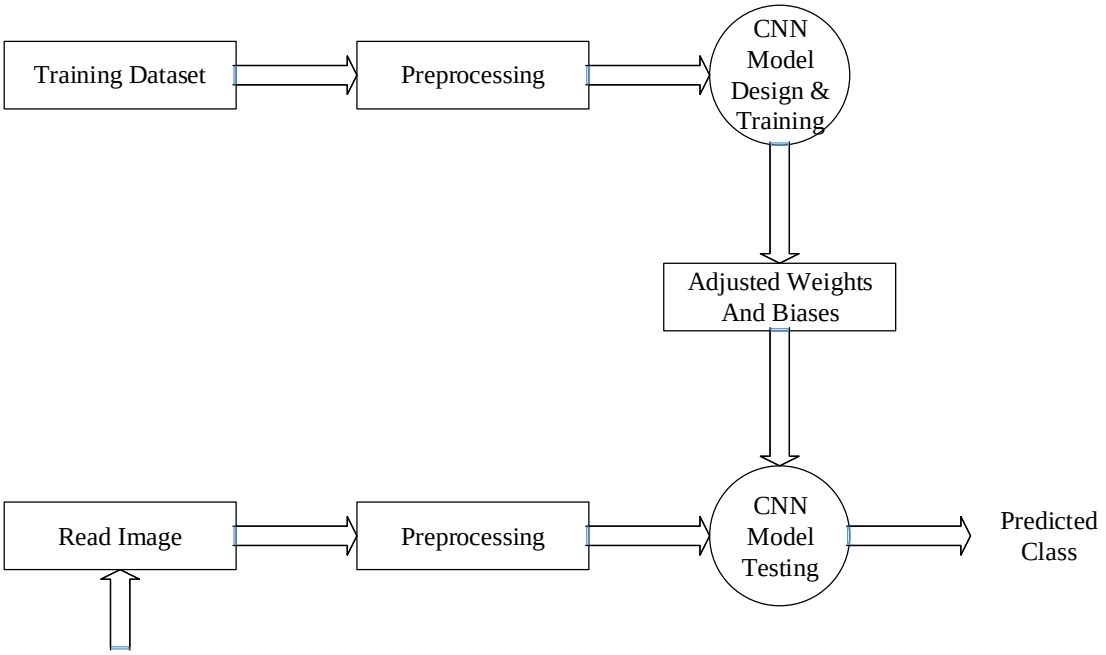

Figure 4.1: Block diagram of the system

Figure 5.1 above shows the block diagram of the system. The system works in two phases: Training and Implementation.

During the training phase, the system reads all the images from training dataset and the images are preprocessed. The preprocessed images are fed into the CNN model. After the completion of training process, the weights and biases are saved and architecture of the model is also saved.

During the implementation phase, the input image is read. After reading input, it is pre-processed. The pre-processed input along with the trained weights and biases of the model are fed into the trained model. The trained model then predicts the class of input image.

4.2 Use Case Diagram

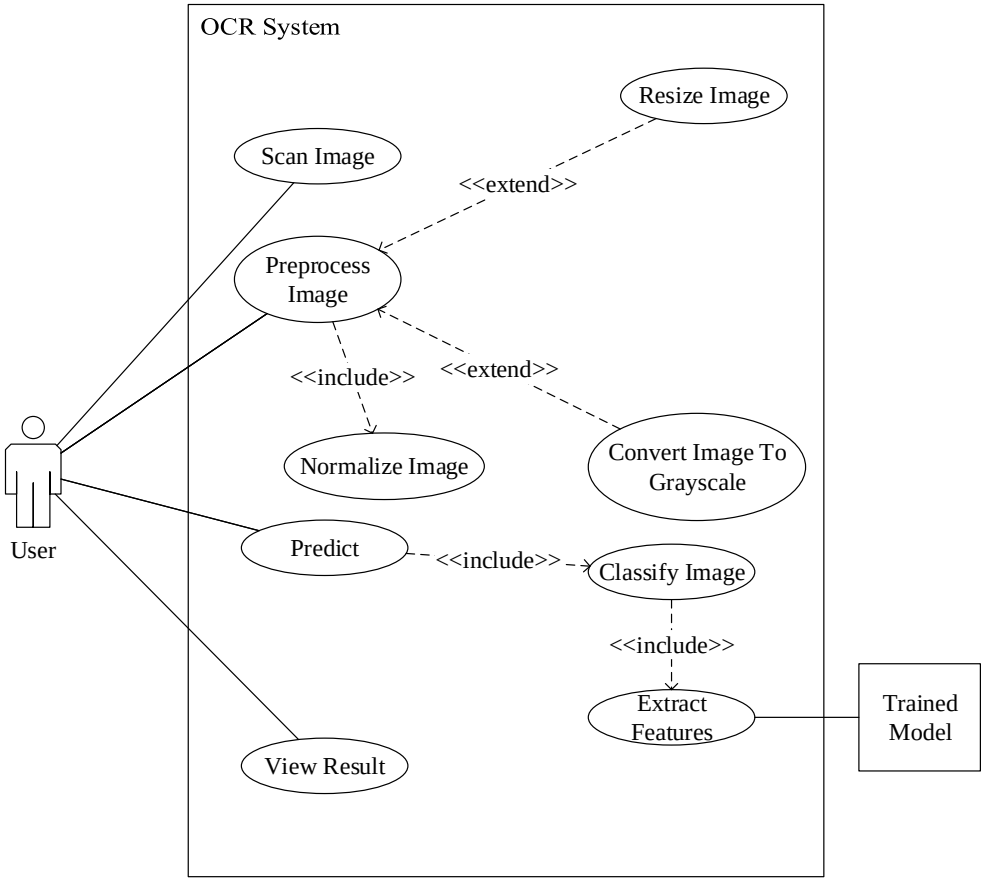

Figure 4.2: UML use case diagram of the system implementing trained model

User first gives an image as input to the system. The input images is preprocessed i.e. converting RGB image to a grayscale image, normalizing the pixel values of the image and resizing the input image to 32x32 dimension. User can then select to classify the image which further extracts the feature and performs classification using the trained model. User selects view result to see the predicted class of the input image.

4.3 State Chart Diagram

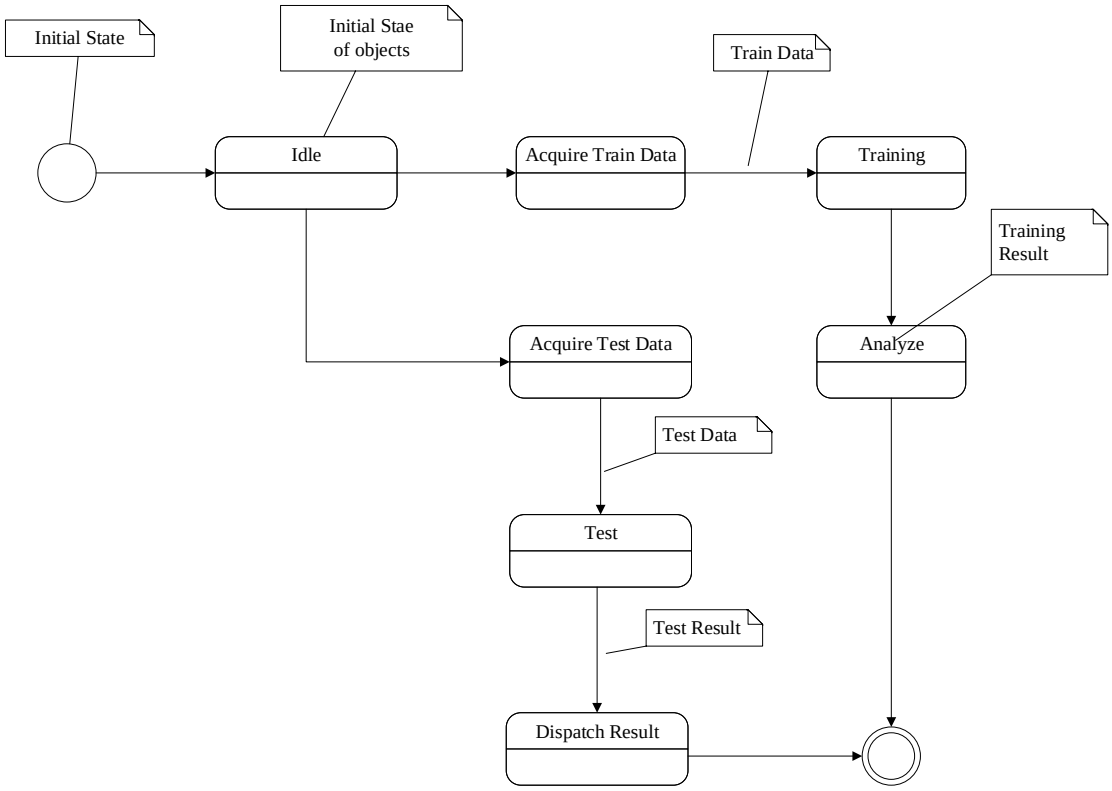

Figure 4.3: State chart diagram for training and testing

Initially, the system rests at idle state. The system can process either for training process or for testing process through acquiring data.

- **Training:**

  The system acquires the raining data and proceeds for training. The system applies the training algorithm on preprocessed training data set and performs defined number of iterations. The training results can be then analyzed. The system then terminates

- **Testing:**

  The system acquires the test data, performs the preprocessing and proceeds for testing. The model fits the test data set to the model and predicts the classes of each of the image in the test data set and finally dispatches the result. The system then terminates.

5. RESULTS AND DISCUSSION

The model was trained on Kaggle handwritten data set consisting of the handwritten images of numerical digits and upper and lower case characters. However, the dataset was not balanced. So only 175,198 images were selected from the data set. Then the selected data set were split into two sets: 90 % of the selected data as training and validation set and the remaining 10% as testing set. From the training and validation set of 157,095 images, 85 % of the set i.e. 133,530 samples were used for training the network and the remaining 15 % i.e. 23,565 samples were used for the validation. Batches of 256 samples were fed into the network with learning rate of 0.001 for the training. With cross entropy loss function and adam optimizer, the training and validation process took 5 hour 31 minute and 35 second for 30 epochs. The resulting training accuracy was 96.03 % and validation accuracy was 91.66 %. The training accuracy starts to saturate from 26th epochs as seen in model accuracy plot in figure 5.1. After 28th epochs accuracy doesn't increase much and hence, the model was trained for 30 epochs only. The validation accuracy fluctuates in between 90 to 92 percent throughout all the epochs.

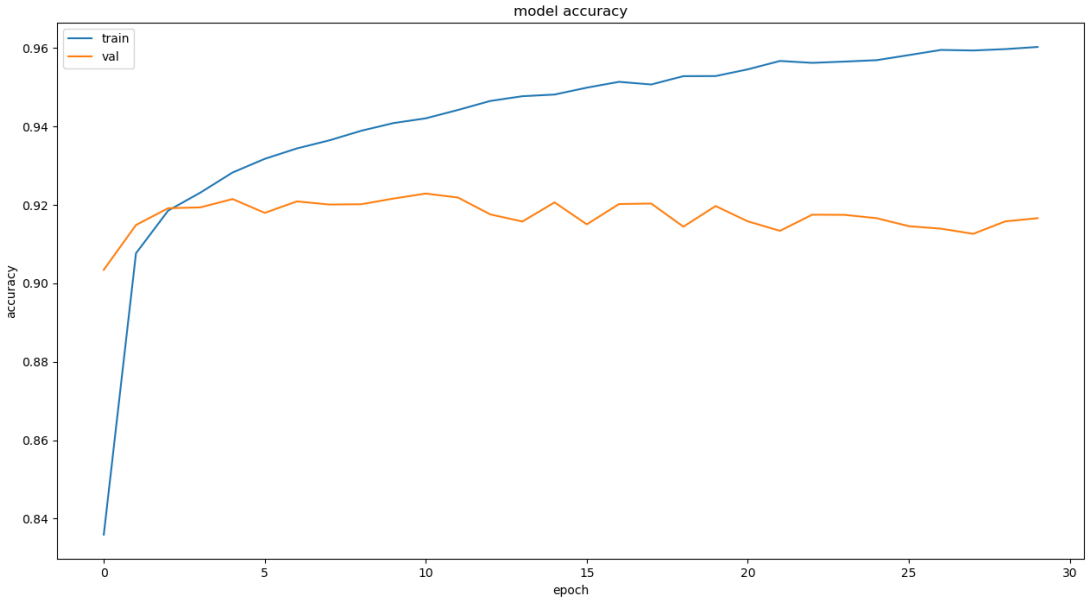

Figure 5.1: Model training and validation accuracy plot

Figure 5.2 shows the training and validation loss plot for the model. The network was designed to lower the loss and as seen in the loss plot, the training accuracy was initially around 0.6 and gradually decreased 0.14 after 30 epochs. The training loss starts to saturate from 26th epoch as seen on the loss plot. However, validation loss doesn't decrease much. In fact, validation loss decreases gradually upto 5 epochs and then starts to increase till 30 epochs. Validation loss rises finally to 0.38 from the initial value of 0.285 as seen on the model loss plot.

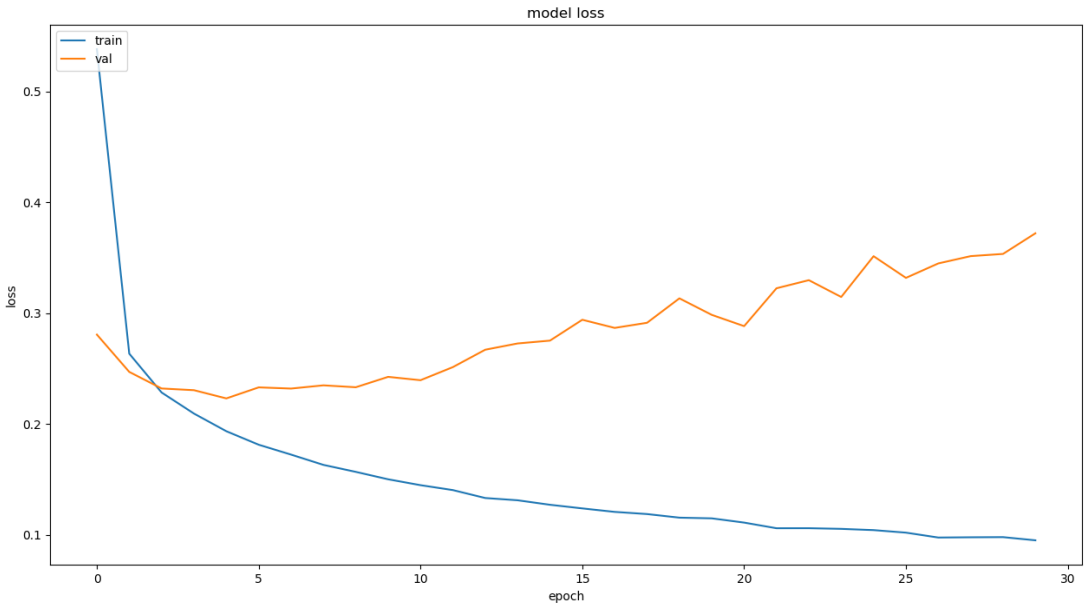

Figure 5.2: Model training and validation loss plot

The resulting model was tested against the test samples consisting of 18.104 image and the confusion matrix was calculated and plotted for the test set. The resulting test accuracy was 85.49 %. From the confusion matrix shown in figure 5.3 below, the model seems to be confused in between some alpha-numeric characters like 1 and I, 1 and l, 1 and L, I and l, 9 and p, Z and 2, Y and 4, g and 9, S and 5 and so on. This is because the characters in these pairs look much similar to each other. If there was enough distinction in between these characters, then model should have surely been able to recognize them successfully.

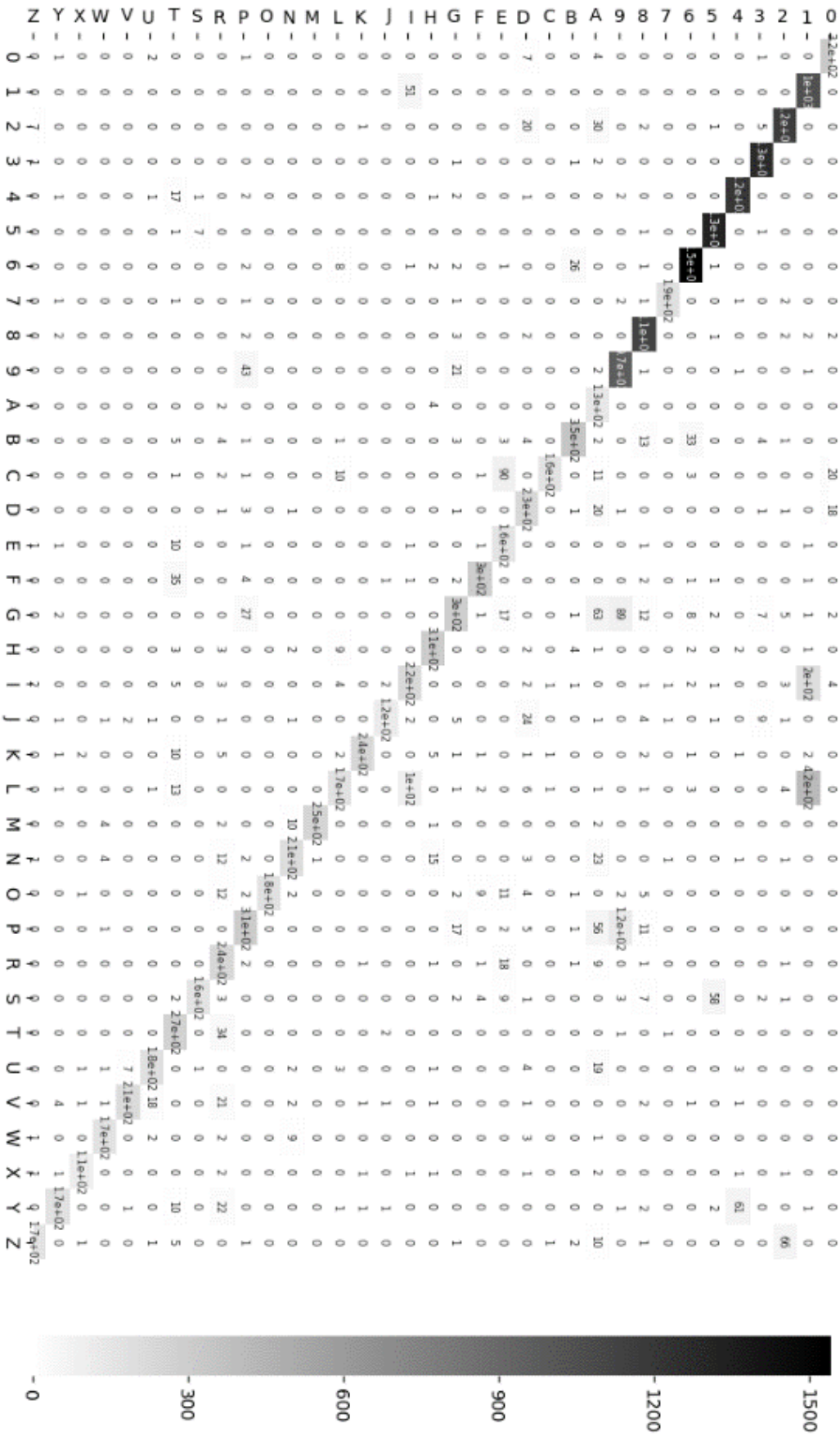

Figure 5.3: Confusion matrix of the model for test data set

6. CONCLUSION

The convolution neural network implemented in python was trained with the Kaggle handwritten dataset. The data set was divided into training, validation and testing sets among which 90 % data was in training and validation set and the remaining 10 % was in testing set. Further, the training and validation set was split into 85 % for training and remaining 15 % for validation. The model was trained in python environment in 64 bit windows 10 operating system running on 2.3 GHZ intel core i5-6200 CPU with 8 GB of 1700 MHZ DDR3 RAM. To complete 30 epoch, the system took 5 hour and 32 minutes. After the training and validation process, the system gave 96.03 % training accuracy and 91.66 % validation accuracy. The system was then tested with the test data set. The system was able to give 85.49 % of test accuracy. So, the system is quite good at recognizing the handwritten texts and performed well during training and testing process. However, the model has some confusion in between some alpha-numeric characters like 1 and I, 1 and l, 1 and L, I and l, 9 and p, Z and 2, Y and 4, g and 9, S and 5, etc. This is because the way they are written in English language. After completing this project, we came to conclusion that CNNs are very efficient and powerful for the image based classification and their implementation can really provide a higher accuracy.

7. LIMITATIONS AND FUTURE WORK

7.1 Limitations

We had the limited data samples. If there were more data samples, the model could result a better accuracy. Due to the limited resources, we could not make the network denser. If we could add more layers and make the network denser, we might have been able to get even a better result. There was also the time limitations, if we did not have to stick within the time boundary, we could optimize the network for a larger epoch like 100, 10000 or even 10,000 which could more refine the weights and biases in the network.

7.2 Future Work

In the future, the model can be retrained on a large dataset to yield even better accuracy. The model can further be retrained for higher epochs to refine the weights and biases. Also the learning rate can be fine-tuned to increase the performance accuracy. The network can be reconstructed increasing the number of filters in the convolution layers and the number of neurons in the hidden layers and adding more convolutional and hidden layers resulting in denser network for the better prediction results.

REFERENCES

[1] Vaibs, "HandWritten_Character | Kaggle," 9 September 2018. [Online]. Available: https://www.kaggle.com/vaibhao/handwritten-characters. [Accessed 5 January 2019].

[2] P. Subashini, "Optical Character Recognition using Artificial Neural Networks," in *ResearchGate*, Berlin, Germany, 2007.

[3] M. Fakir, M. M. Hassani and C. Sodeyama, "On The Recognition of Arabic Characters Using Hough Transform Technique," *Malaysian Journal of Computer Science,* vol. 13, no. 2, pp. 39-47, 2 December 2000.

[4] S. Tatale and A. Khare, "Character Recognition And Transmission of Characters Using Network Security," *International Journal of Advancesin Engineering & Technology,* vol. 1, no. 5, pp. 351-360, November 2011.

[5] H. I. Avi-Itzhak, T. A. Diep and H. Garland, "High accuracy optical character recognition using neural networks with centroid dithering," *EEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 17, no. 2, February 1995.

[6] R. Romero, D. Touretzky and R. Thibadeau, "Optical Chinese character recognition using probabilistic neural networks," *Pattern Recognition,* vol. 30, no. 8, pp. 1279-1292, August 1997.

[7] S. Varshney, . R. Chaurasiya and Y. Tayal, "Optical Character Recognition using Neural Network," *International Journal of Emerging Technology and Advanced Engineering,* vol. 4, no. 8, p. 5, August 2014.

[8] J. Brownlee, "Supervised and Unsupervised Machine Learning Algorithms," 16 March 2016. [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. [Accessed 14 June 2019].

[9] "Artificial Intelligence Neural Networks," [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm. [Accessed 12 June 2019].

[10] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way," 15 December 2018. [Online]. Available:

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed 12 June 2019].

APPENDICES

A: Sample train and test images before and after pre-processing

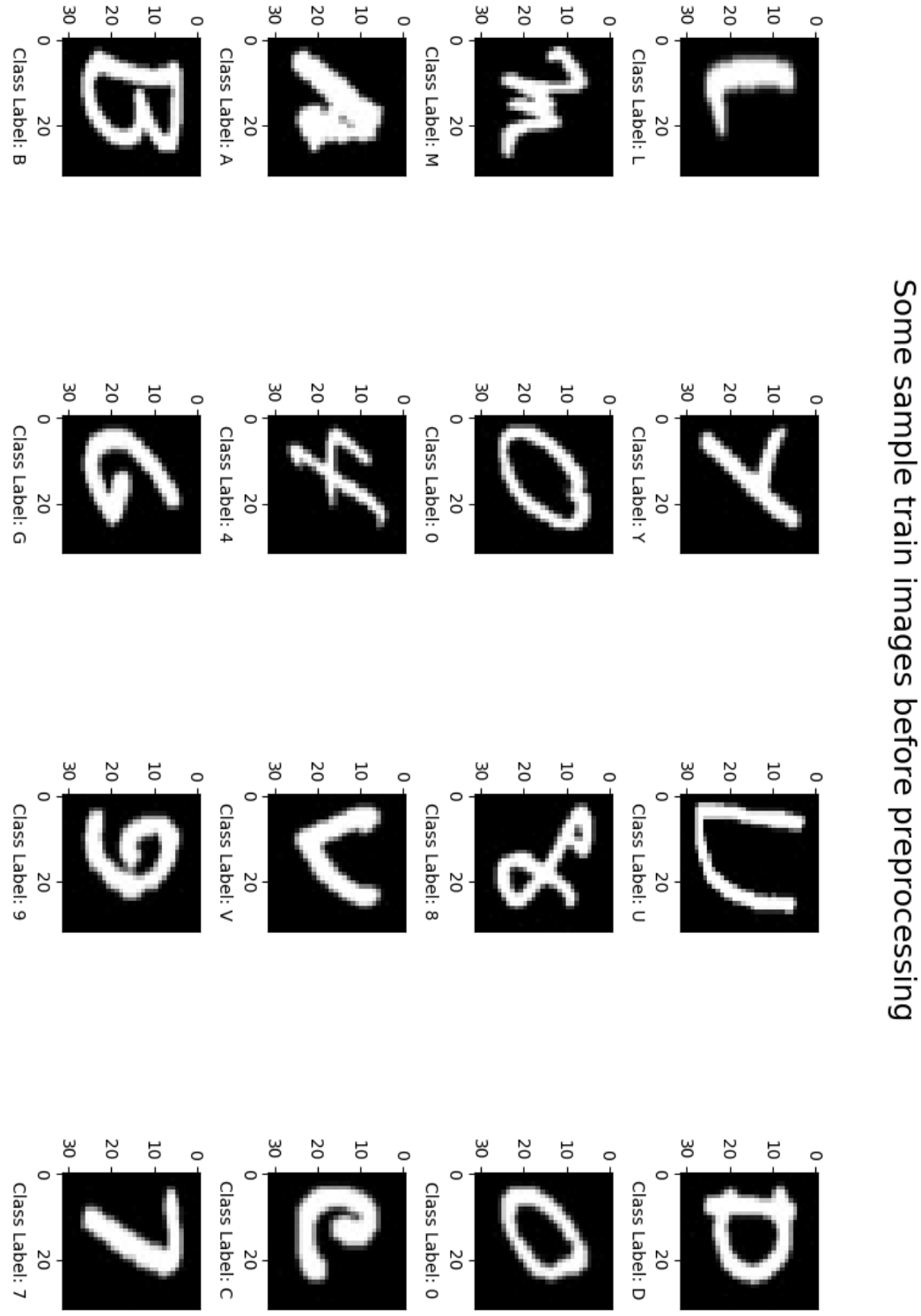

Figure A1: Sample train images before preprocessing

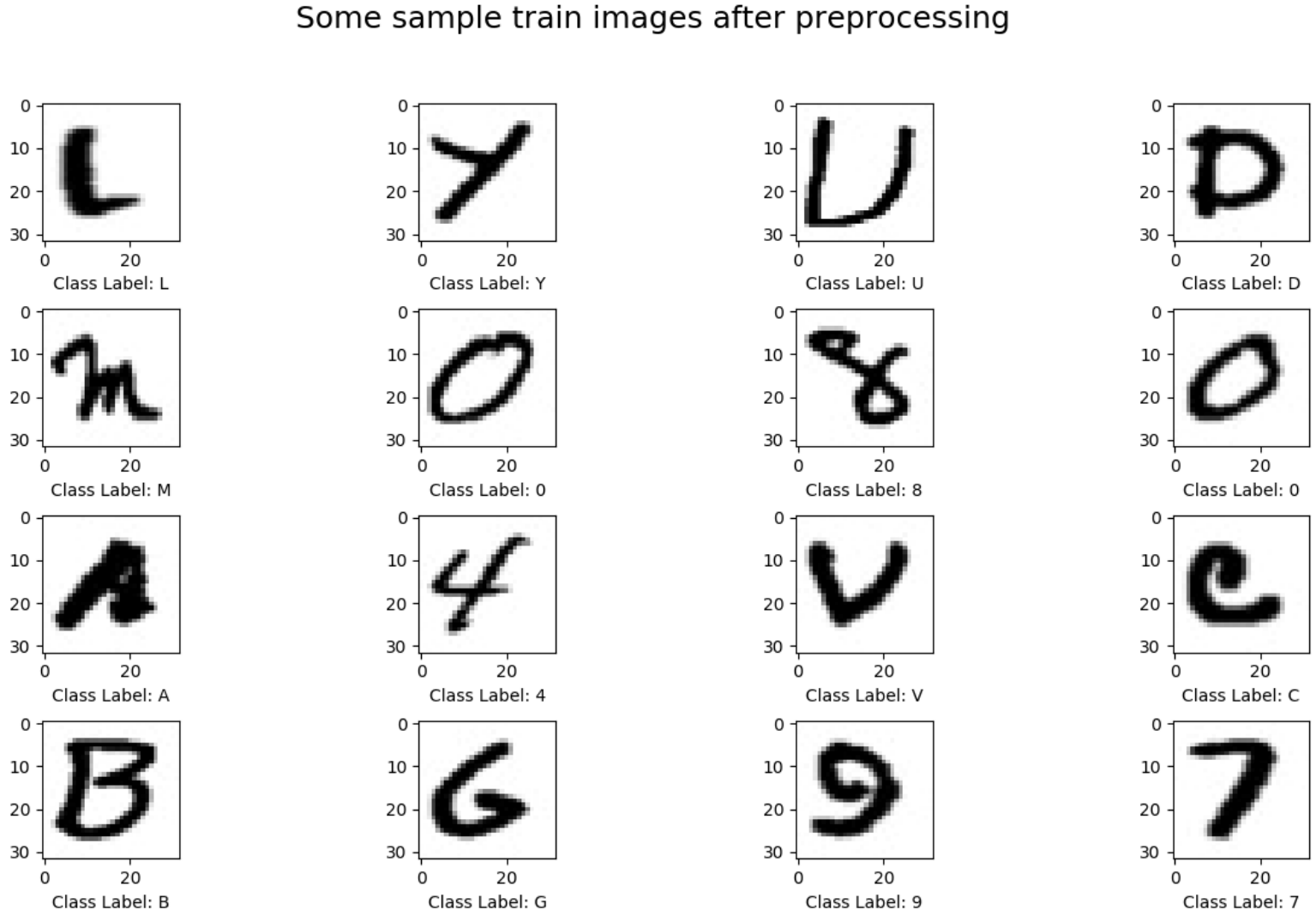

Figure A_2: Sample train images after preprocessing

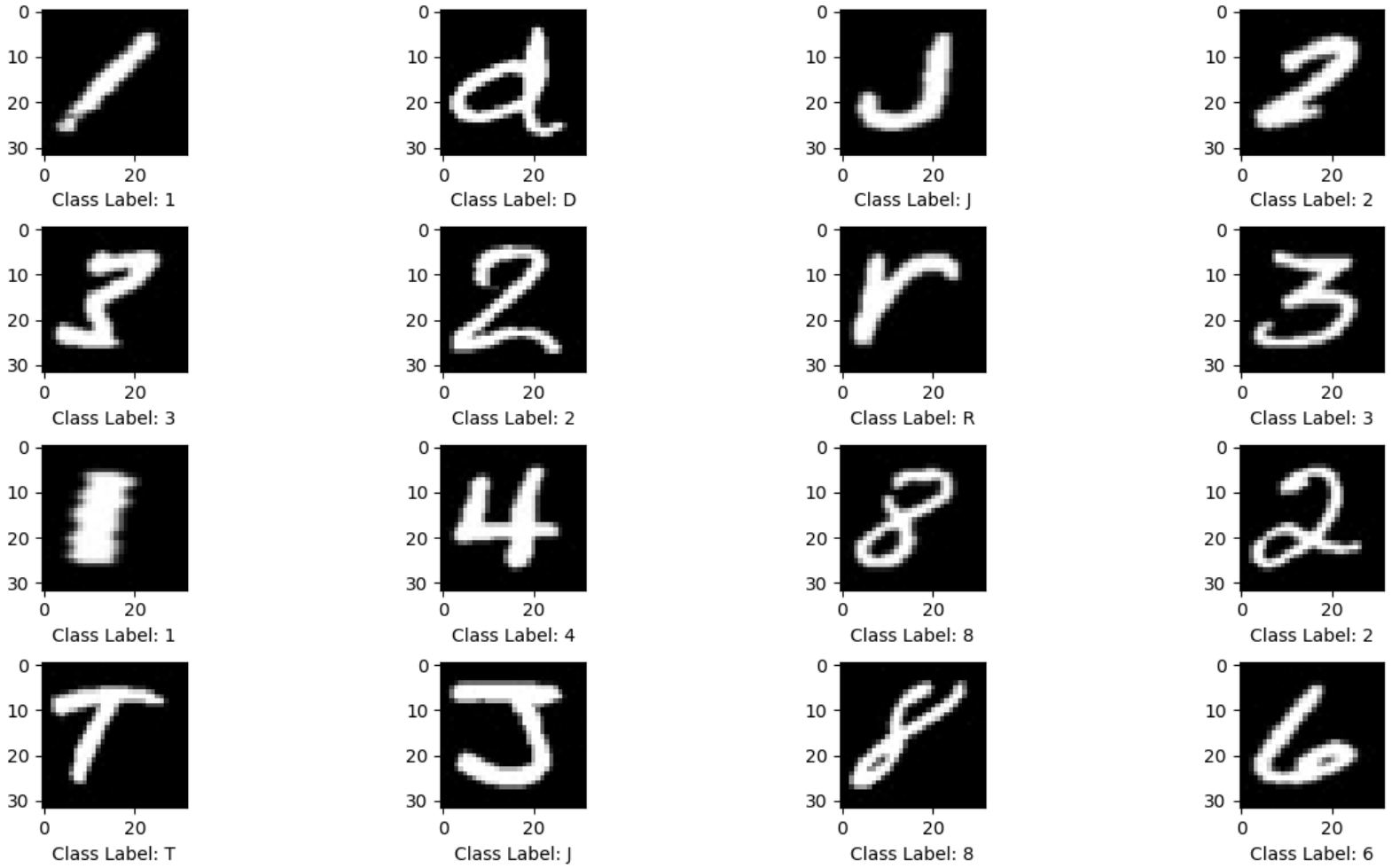

Figure A_3: Sample test images before preprocessing

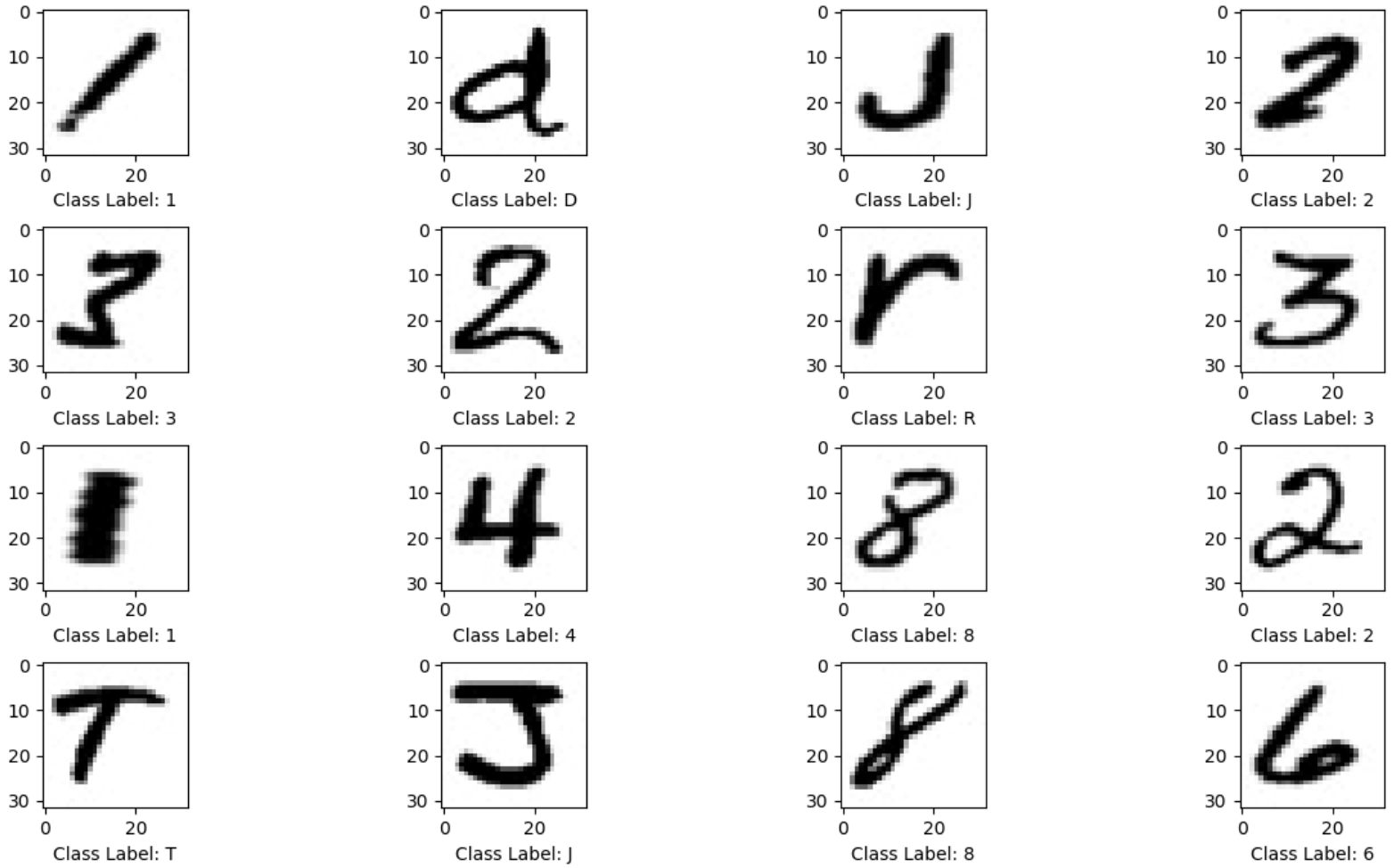

Figure C2: Sample test images after preprocessing

B: Visualization of filters obtained from convolution layers

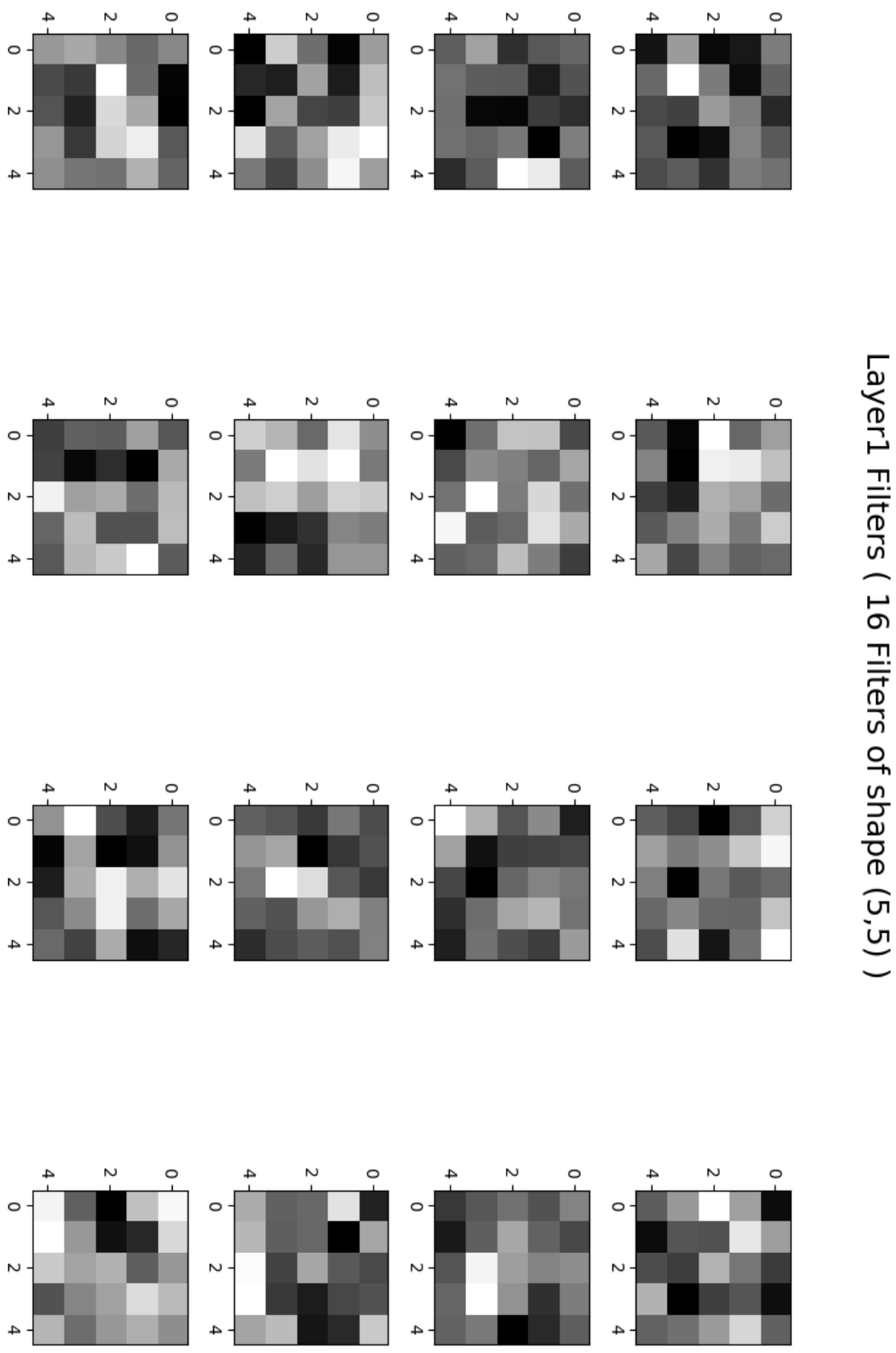

Figure B₁: Visualization of filters of first convolution layer

Figure B_2: Visualization of filters of second convolution layer for first output channel from first convolution layer

Figure B_3: Visualization of filters of second convolution layer for second output channel from first convolution layer

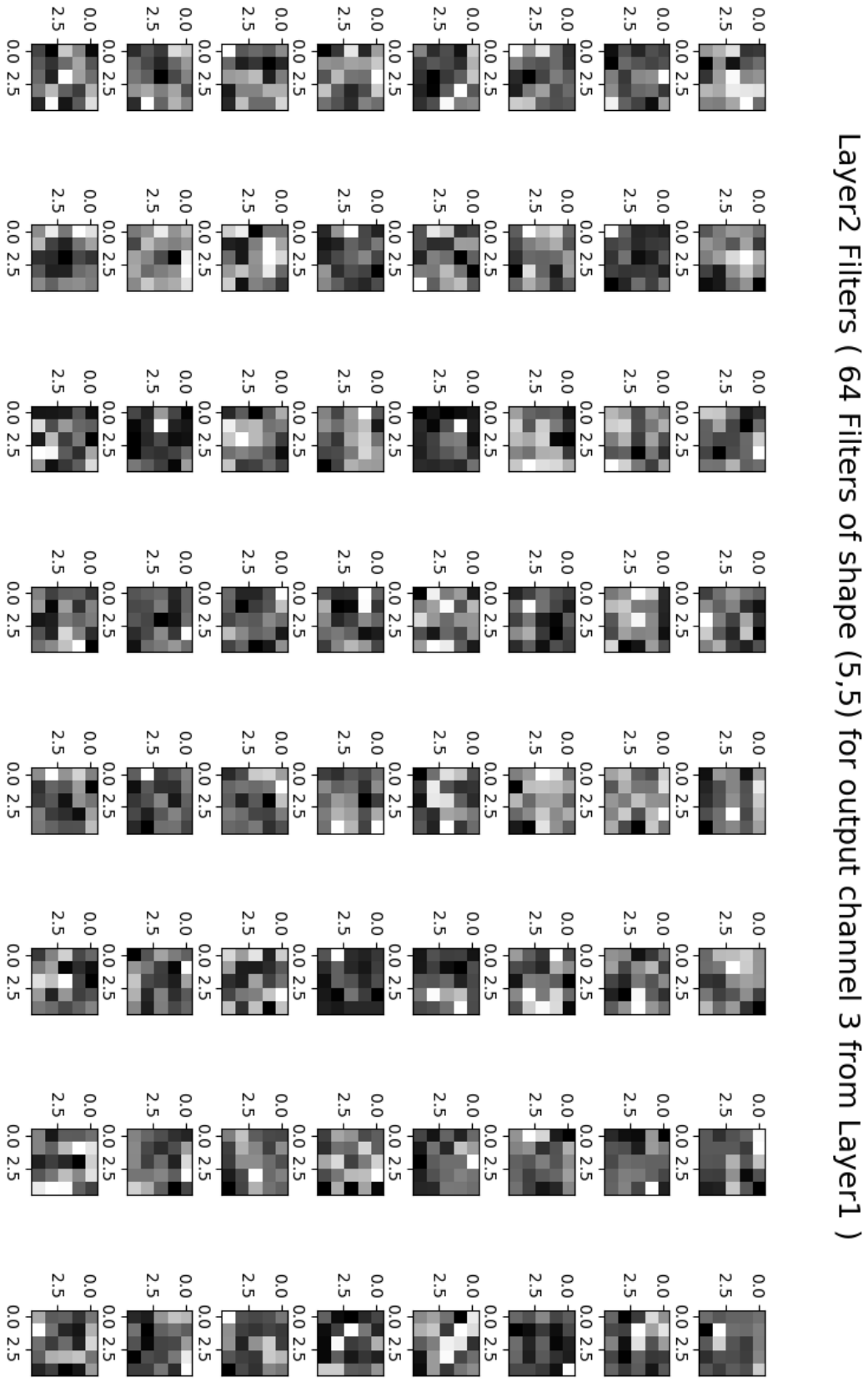

Figure B4: Visualization of filters of second convolution layer for third output channel from first convolution layer

Figure B_5: Visualization of filters of second convolution layer for fourth output channel from first convolution layer

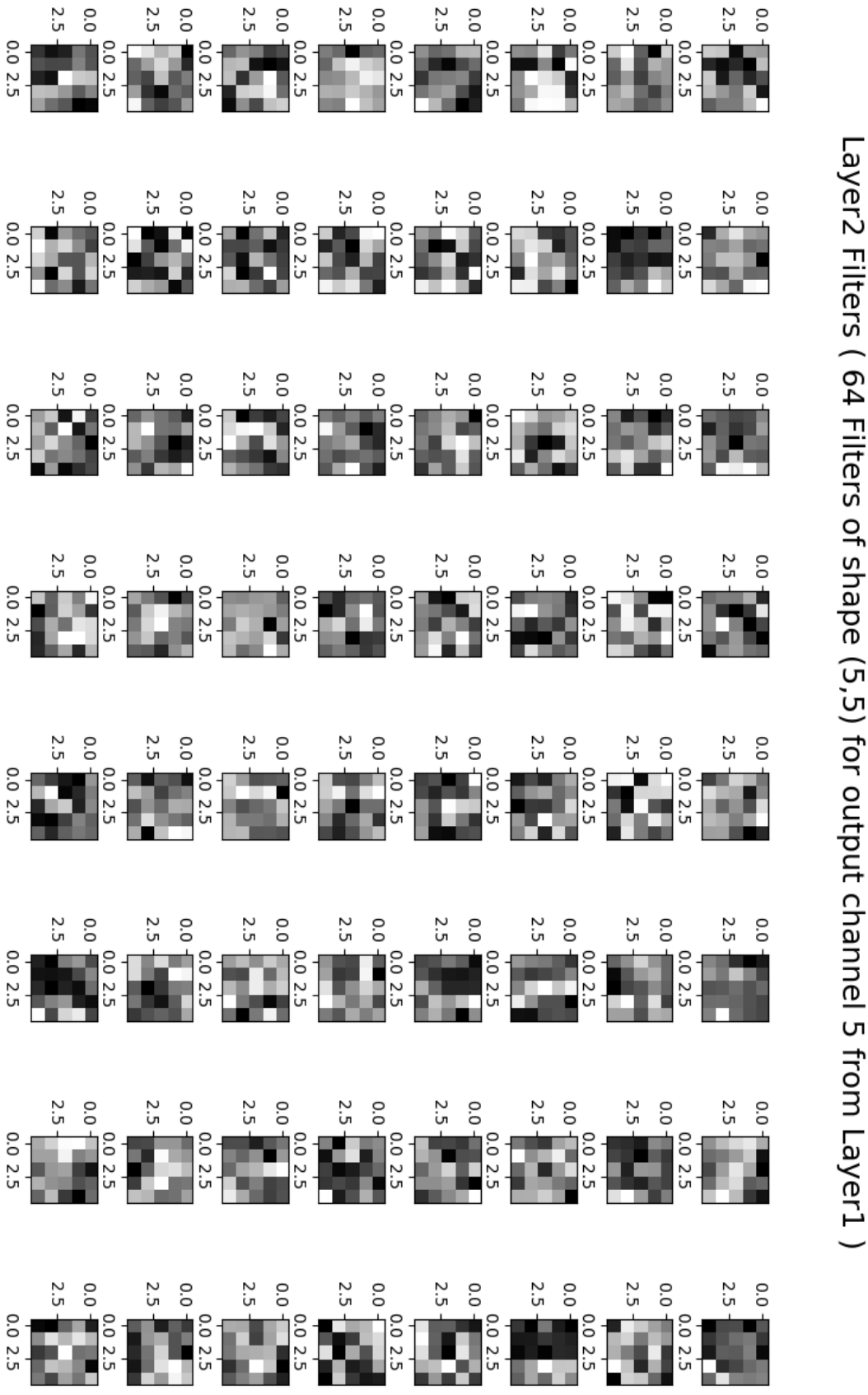

Figure B6: Visualization of filters of second convolution layer for fifth output channel from first convolution layer

Figure B7: Visualization of filters of second convolution layer for sixth output channel from first convolution layer

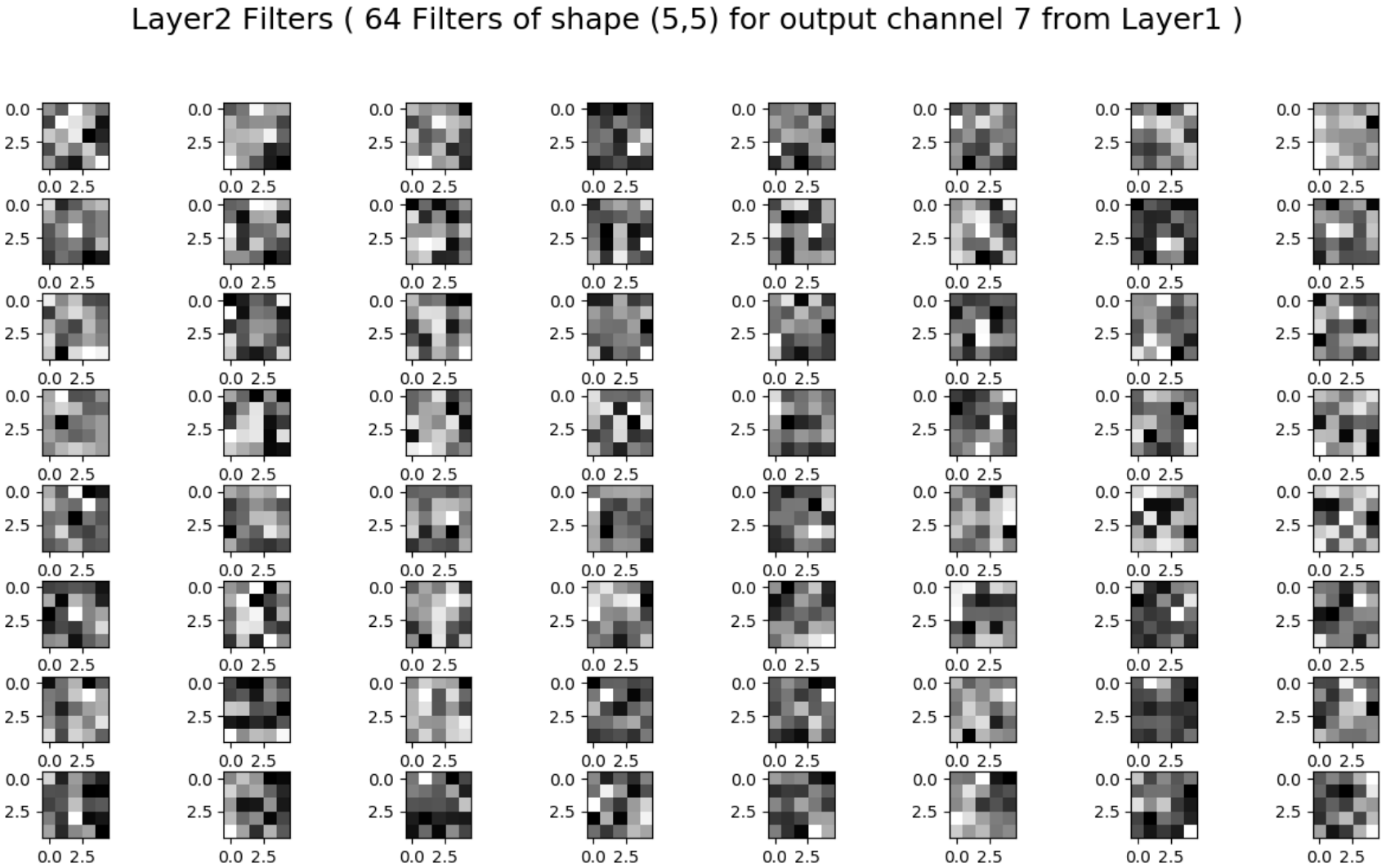

Figure B_8: Visualization of filters of second convolution layer for seventh output channel from first convolution layer

Figure B_9: Visualization of filters of second convolution layer for eighth output channel from first convolution layer

Figure B_{10}: Visualization of filters of second convolution layer for ninth output channel from first convolution layer

Figure B_{11}: Visualization of filters of second convolution layer for tenth output channel from first convolution layer

Figure B_{12}: Visualization of filters of second convolution layer for eleventh output channel from first convolution layer

Figure B_{13}: Visualization of filters of second convolution layer for twelvth output channel from first convolution layer

Figure B_{14}: Visualization of filters of second convolution layer for thirteenth output channel from first convolution layer

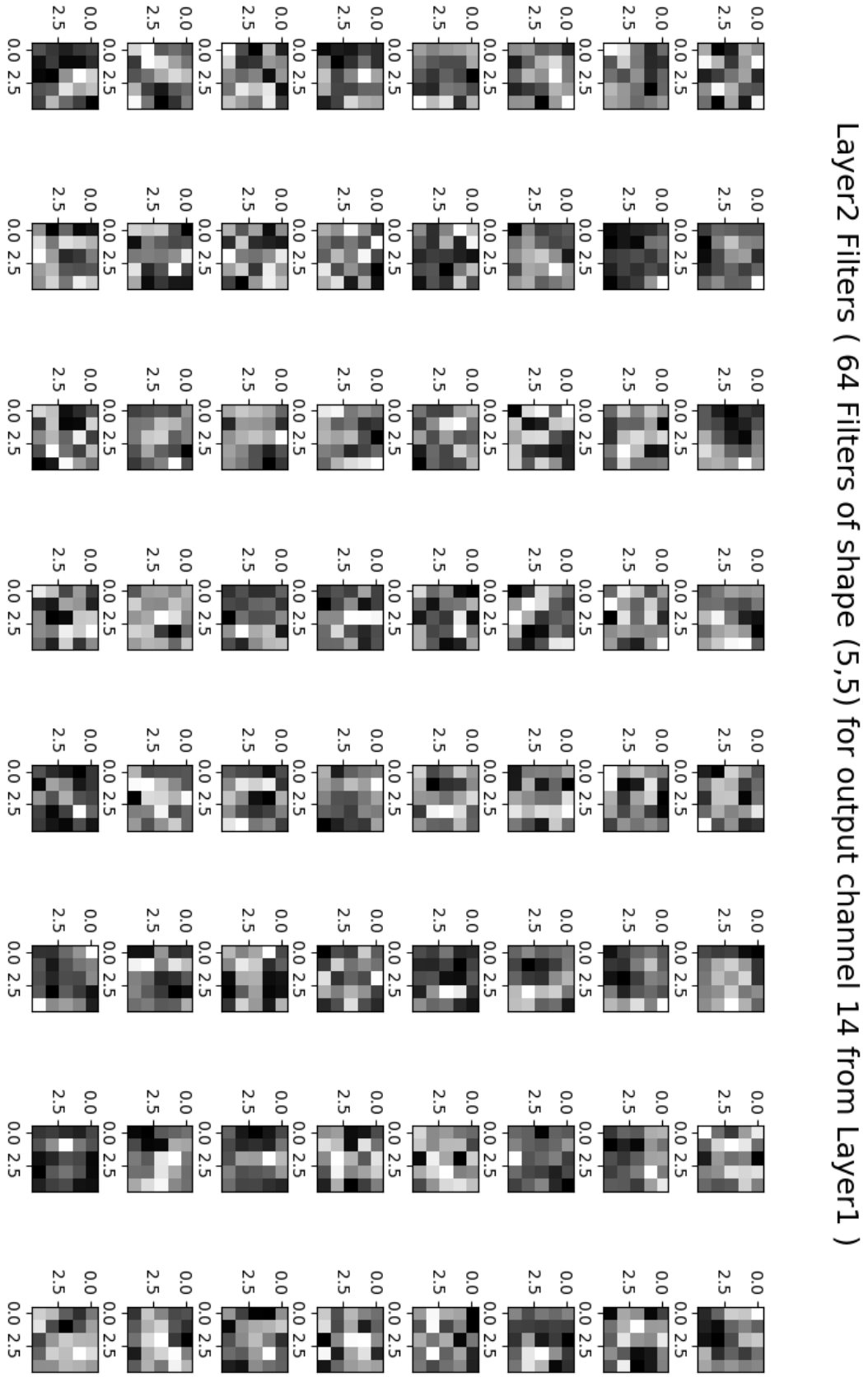

Figure B_{15}: Visualization of filters of second convolution layer for fourteenth output channel from first convolution layer

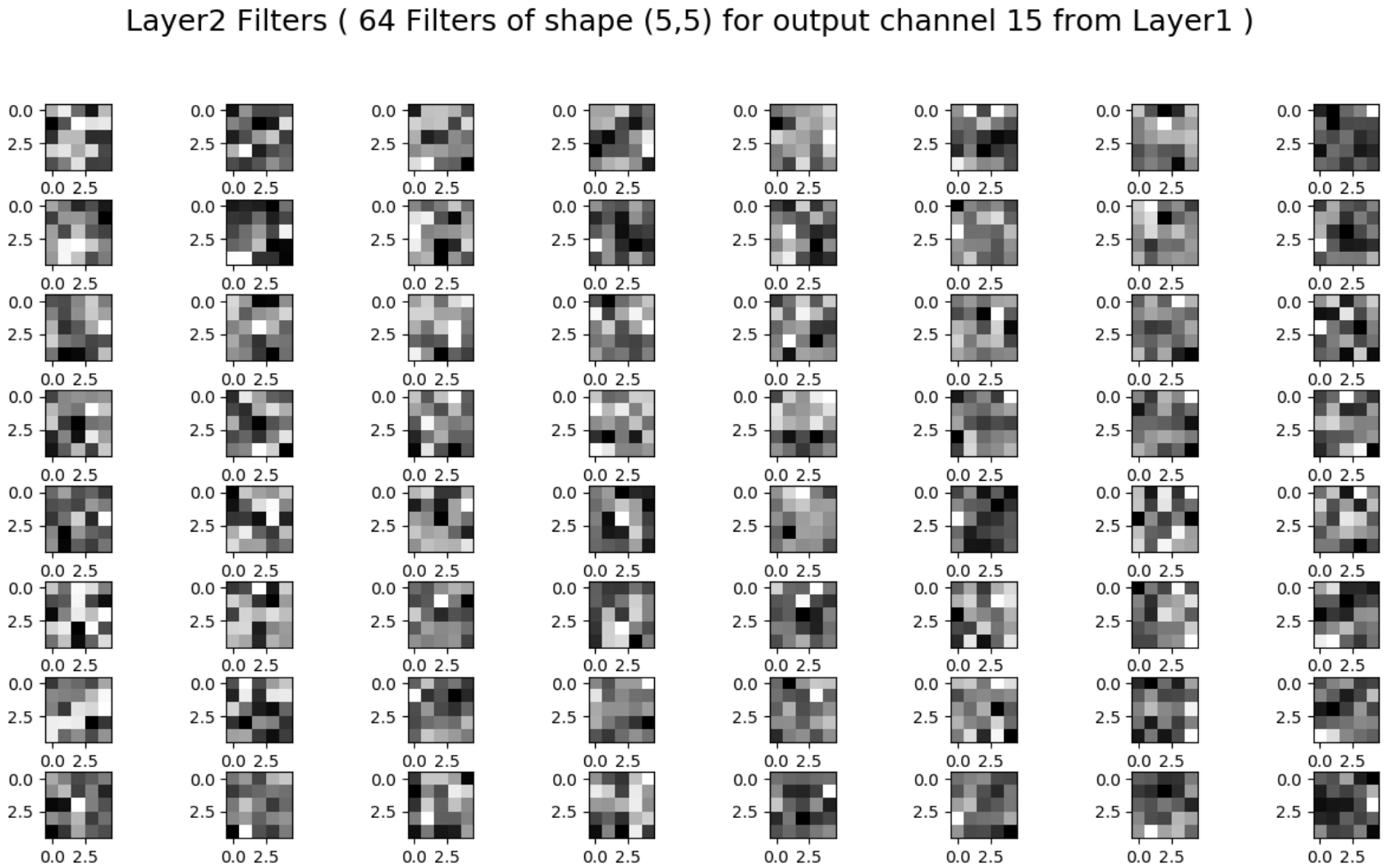

Figure B_{16}: Visualization of filters of second convolution layer for fifteenth output channel from first convolution layer

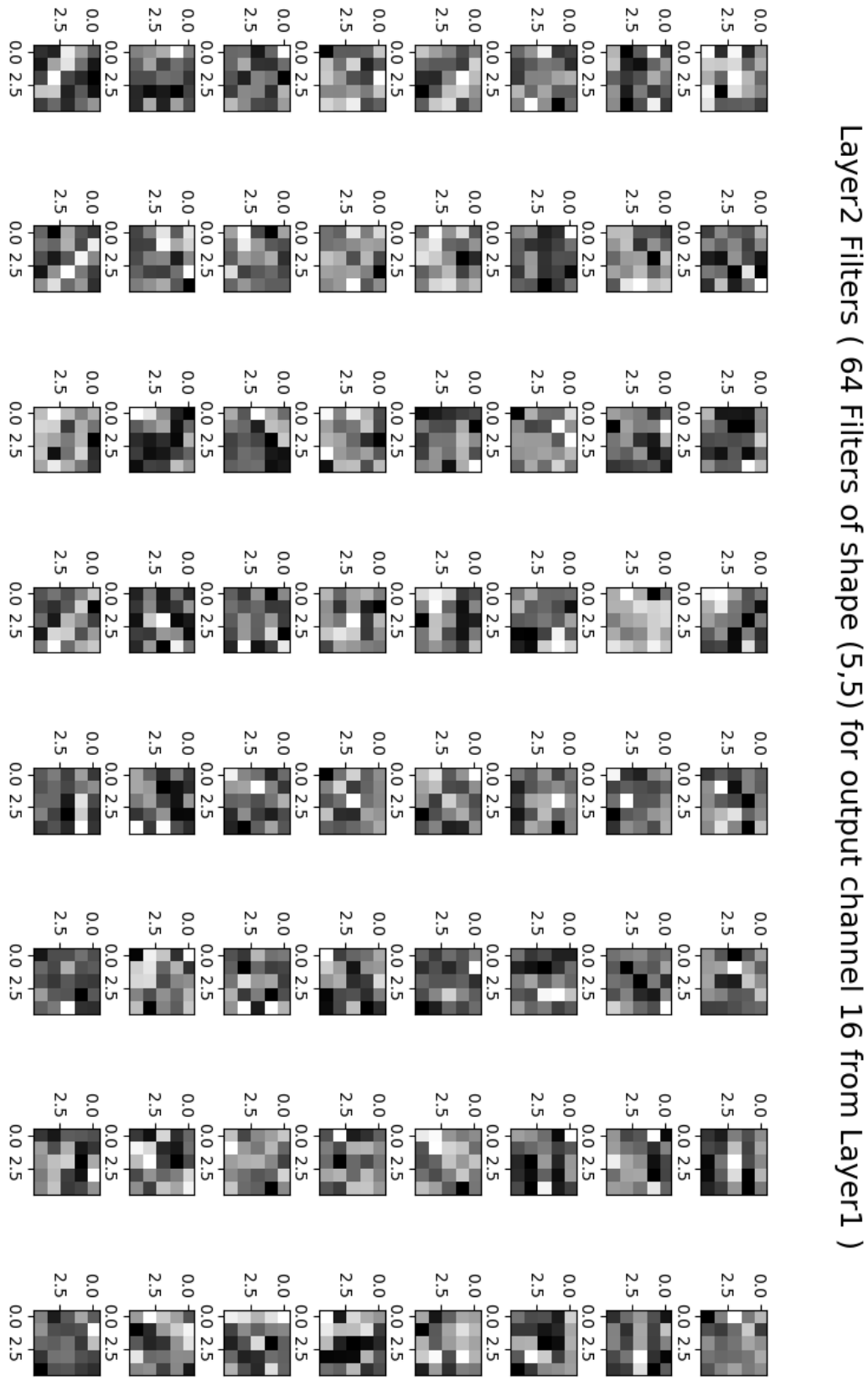

Figure B_{17}: Visualization of filters of second convolution layer for sixteenth output channel from first convolution layer

C: Some samples of truly classified and miss-classified test images

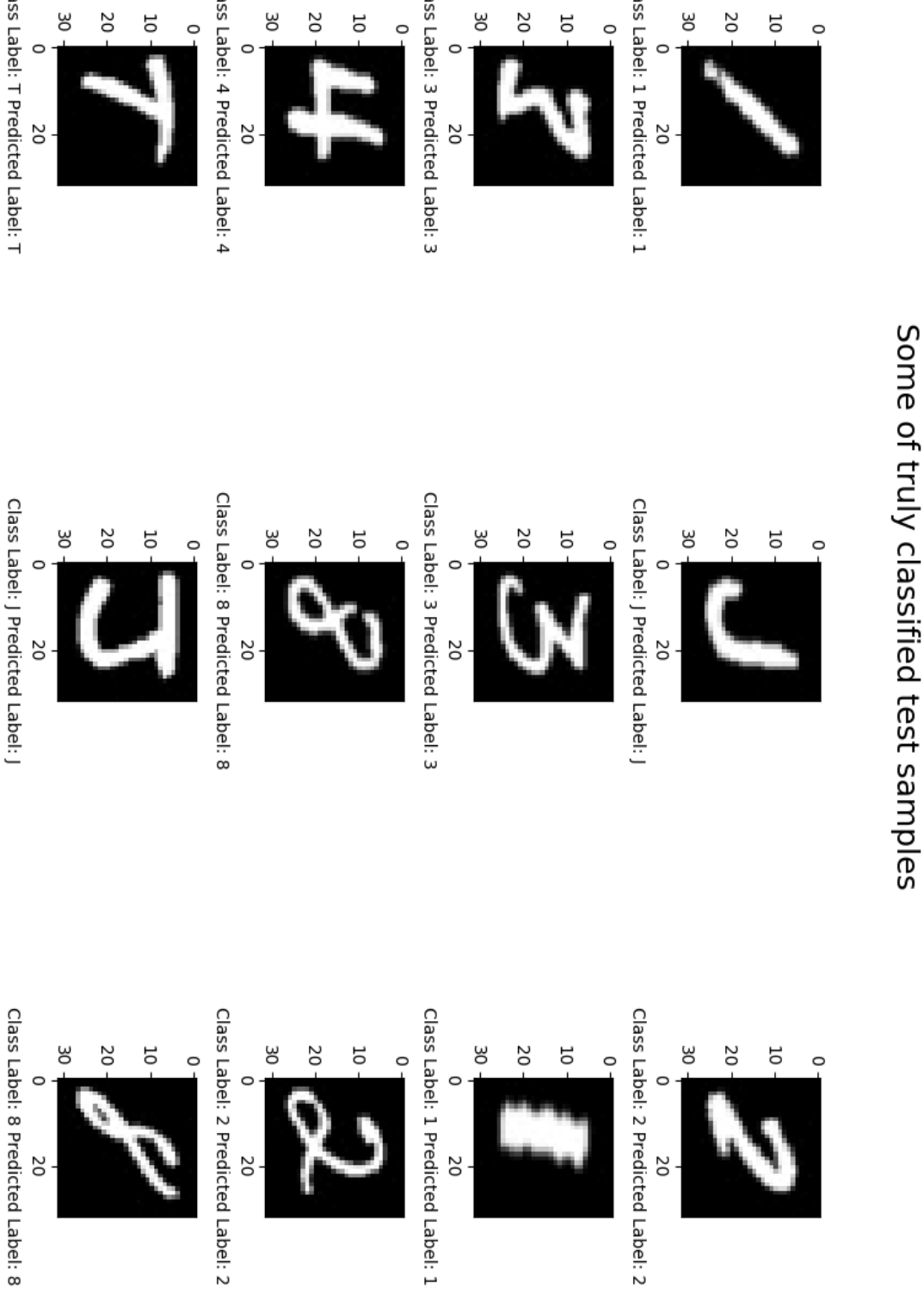

Figure C_1: Some truly classified test samples

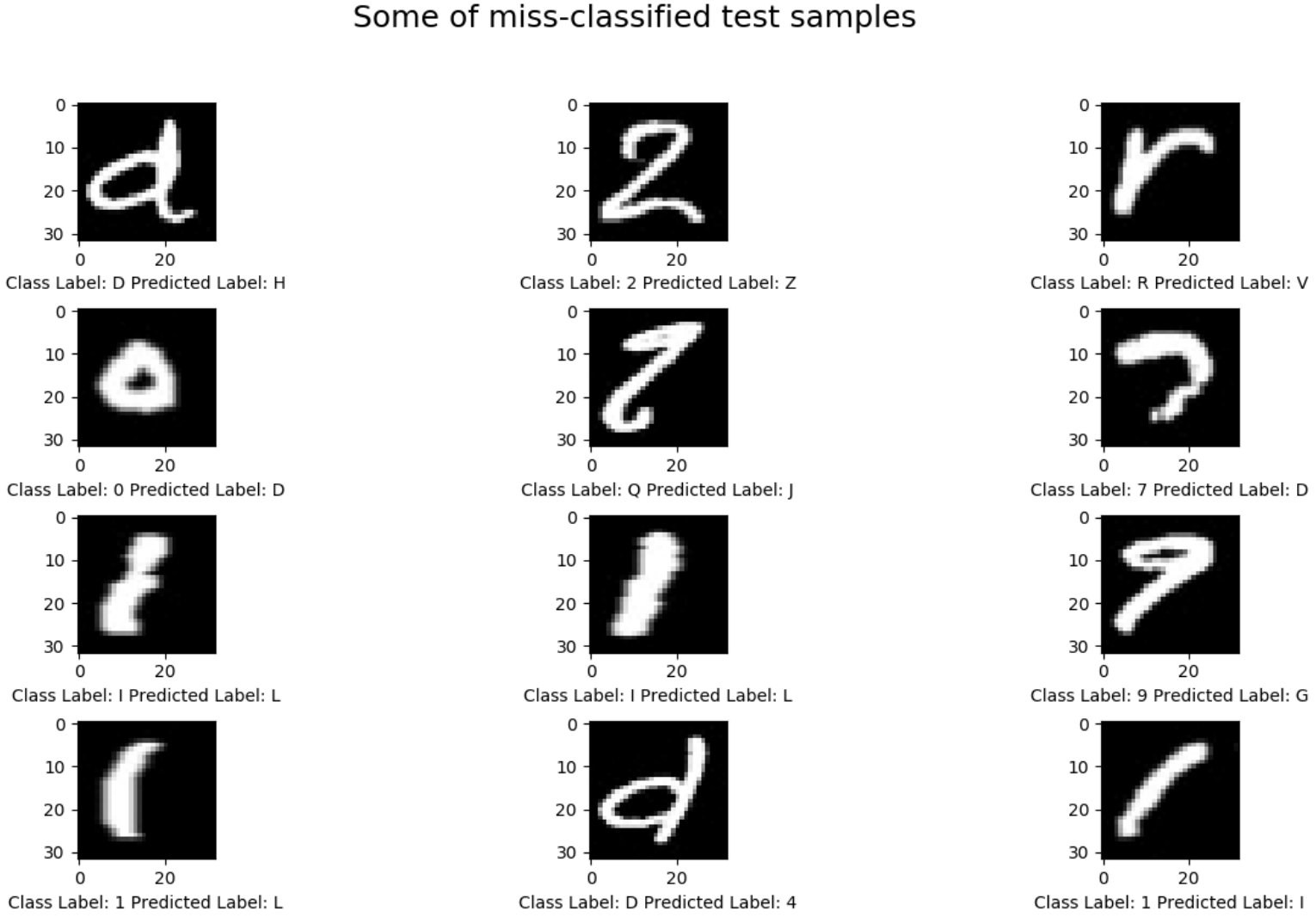

Figure C_2: Some miss-classified test samples

D: Some sample runs on own test data

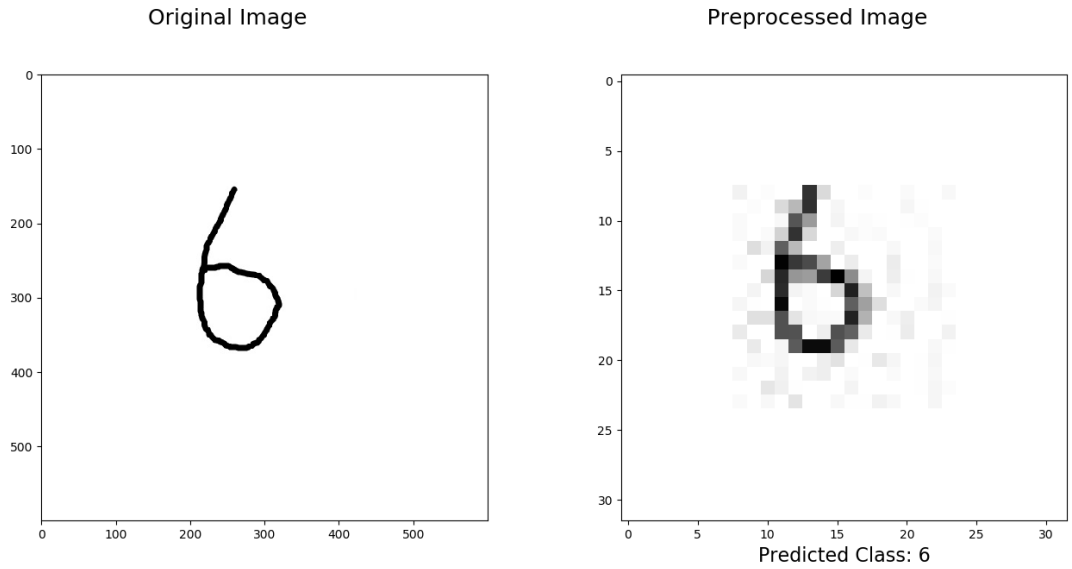

Figure D_1: Sample run on own image data1

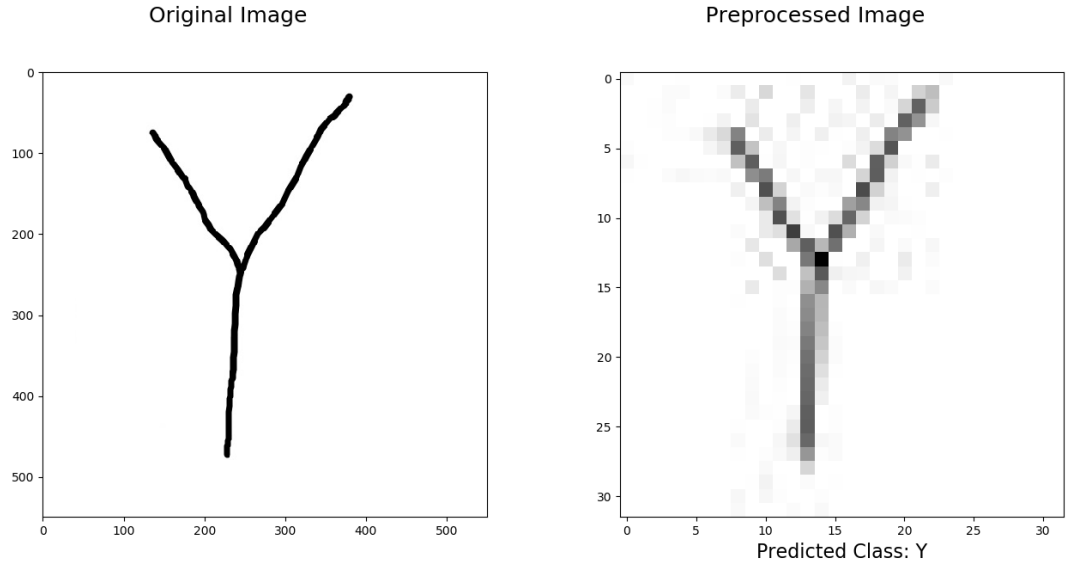

Figure D_2: Sample run on won image data2

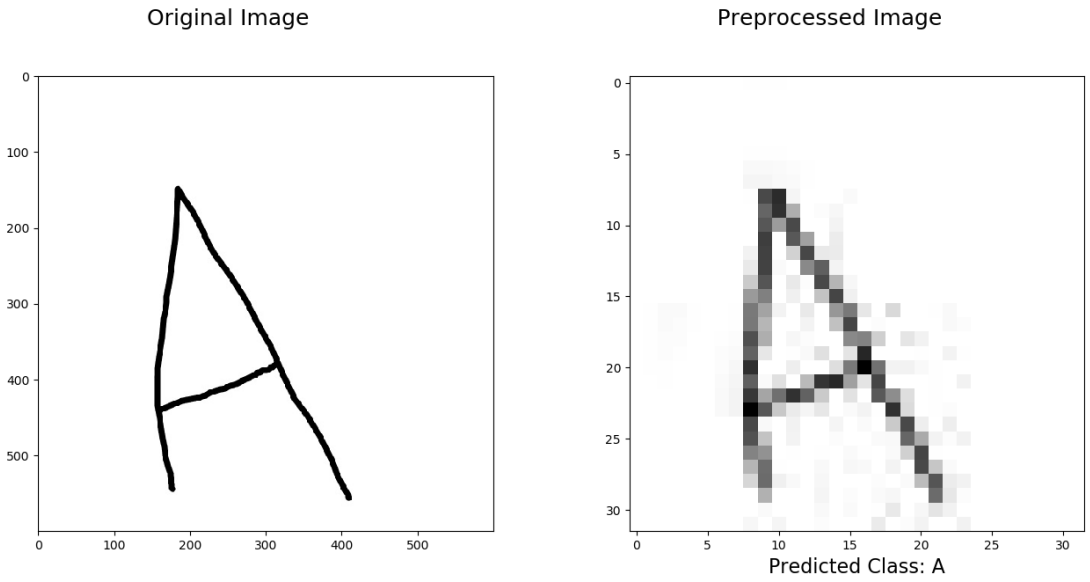

Figure D_3: Sample run on won image data3

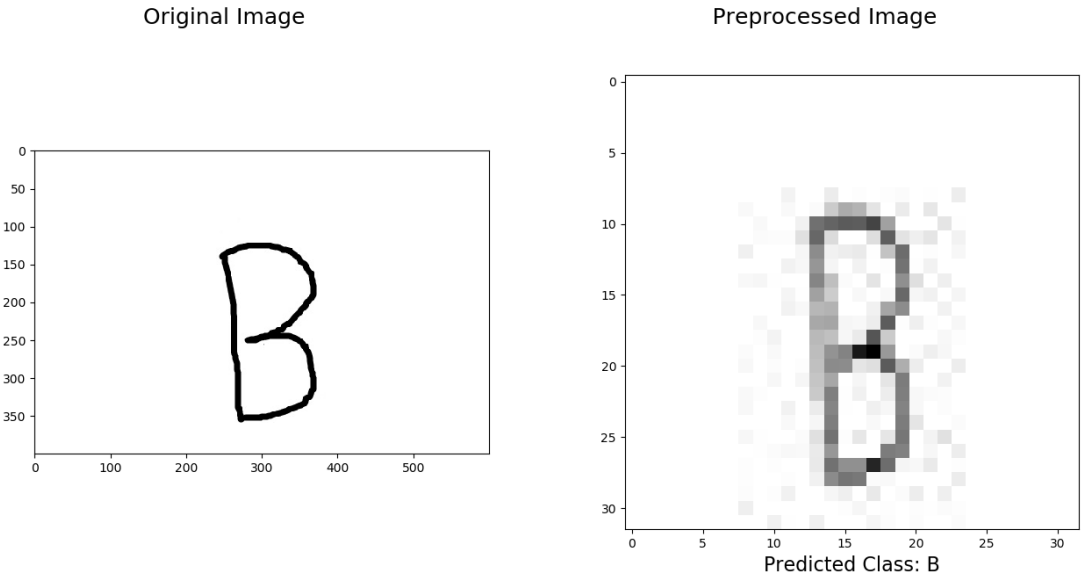

Figure D4: Sample run on won image data4

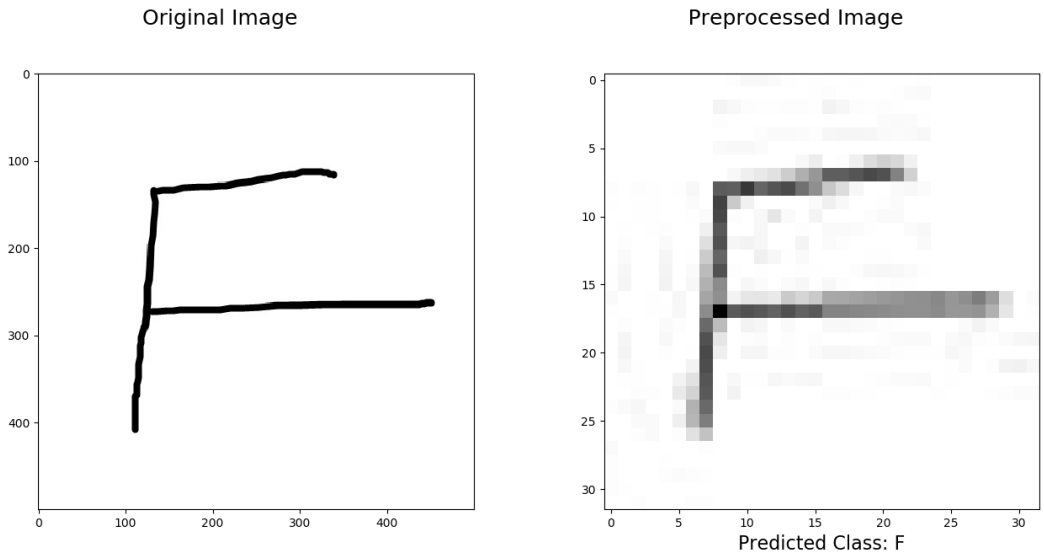

Figure D_5: Sample run on won image data5

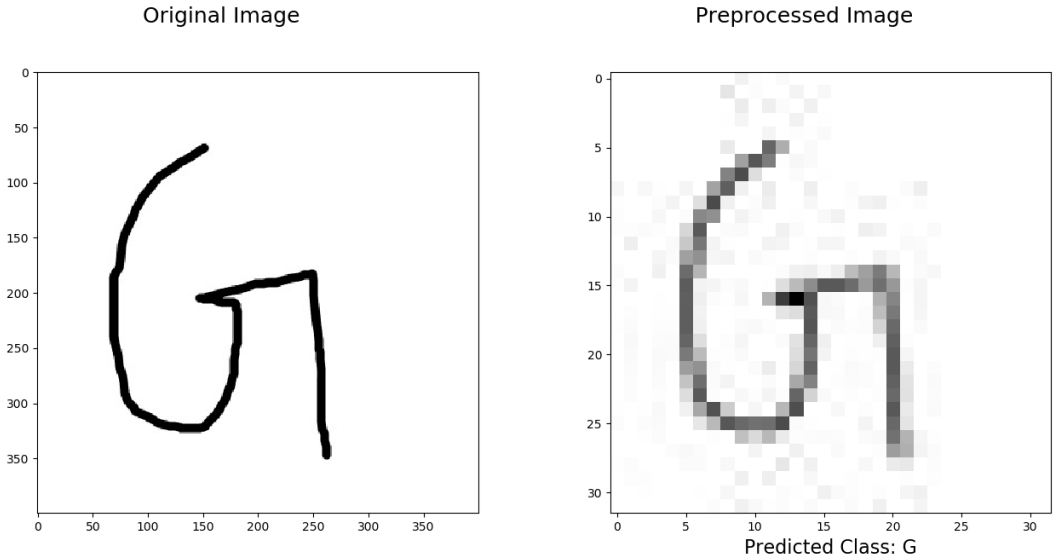

Figure D_6: Sample run on won image data6