THESIS

CLASSIFICATION USING OUT OF SAMPLE TESTING OF NEURAL NETWORKS AND SIAMESE-LIKE NEURAL NETWORK FOR HANDWRITTEN CHARACTERS

Submitted by

Sri Sagar Abhishek Yeluri

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2020

Master's Committee:

Advisor: Dr. Charles W. Anderson

Dr. Ross Beveridge
Dr. Ann Hess

ABSTRACT

CLASSIFICATION USING OUT OF SAMPLE TESTING OF NEURAL NETWORKS AND SIAMESE-LIKE NEURAL NETWORK FOR HANDWRITTEN CHARACTERS

In a world where Machine Learning Algorithms in the field of Image Processing is being developed at a rapid pace, a developer needs to have a better insight into all the algorithms to choose one among them for their application. When an algorithm is published, the developers of the algorithm compare their algorithm with already available well-performing algorithms and claim their algorithm outperforms all or the majority of other algorithms in terms of accuracy. However, adaptability is a very important aspect of Machine Learning which is usually not mentioned in their papers. Adaptability is the ability of a Machine Learning algorithm to work reliably in the real world, despite the change in the environmental factors in comparison to the environment in which data used for training is recorded. A machine learning algorithm that can give good results only on the dataset has no practical applications. In real life, the application of the algorithm increases only when it is more adaptable in nature. A few other aspects that are important in choosing the right algorithm for an application are consistency, time and resource utilization and the availability of human intervention. A person choosing amongst a list of algorithms for an application will be able to make a wise decision if given additional information, as each application varies from one another and needs a different set of characteristics of an algorithm for it to be well received. We have implemented and compared three Machine Learning algorithms used in image processing, on two different datasets and compare the results. We observe that certain algorithms, even though better than others in terms of accuracy on paper, fall behind when tested in real-world datasets. We put forward a few suggestions that if followed will simplify the selection of an algorithm for a specific purpose.

ACKNOWLEDGEMENTS

I would like to thank Prof. Chuck Anderson for his constant support and guidance through this project. Dr. Anderson has provided me with valuable insights and resources to work on this project on a topic that has not been explored in-depth, thereby lacking a lot of material. I would like to thank the CS Department of Colorado State University for providing its students with various resources and amenities. I would like to thank my family and friends, especially Gary, for the support they have extended towards me. Finally, I would like to appreciate my parents Diwakar Yeluri and Sarala Yeluri for their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The term Machine Learning was coined by Arthur Samuel, in 1959, while at IBM. Even though initially there was some research interest in the field, it lost its importance as the statistical approach was replaced by expert systems. In the 1980s, with the re-imagination of backpropagation techniques, Machine Learning came back into prominence. And in the 1990s and later, the advent of Machine Learning took place due to the availability of huge amounts of data, the ability to digitize data, the convenience of sharing data over the internet, and the availability of more powerful systems/processors.

Since the 1990s, due to the better computers that were available for the Machine learning algorithms to perform their computations, many developers and researchers released various algorithms that try to provide a slight improvement over other existing algorithms. The newer algorithms are taking advantage of the powerful system resources available to give better results. The developers/researchers then release the algorithm for public use, along with a paper putting forward the results in terms of accuracy in comparison to other algorithms. Some papers also mention other metrics such as training time and/or the time taken for prediction of a certain size of the test portion of a dataset as seen in [1] where they gave training time and classification time for 25% of test data. They give these metrics with respect to their runs over a certain data set. Training and testing are done from data obtained from the same datasets.

With a plethora of algorithms available out in the industry to choose from, a developer or a business choosing an algorithm to implement for their system/application, need to make an informed choice about selecting the algorithm that suits their requirements the most. There are various other factors that need to be considered in order to make a wise decision, such as adaptability, consistency, time and resource utilization and the availability of human intervention. And the impact of the factors varies from field to field and business model to business model. This report puts forward suggestions to provide certain additional information which will help make an informed

decision. The information provided with is respect to the metrics of accuracy, consistency, time and resource utilization, and the availability of human intervention.

We trained and tested Fully Connected, Convolutional, and Siamese-like Convolutional Neural [2] Networks over two datasets: MNIST and EMNIST datasets [3]. We compare the performance of the algorithms and make some key observations in terms of accuracy, consistency, time and resource utilization. We also tested the trained networks on a new dataset prepared by us to compare this out-of-sample performance to the performance in comparison to the performance over the test portion of the dataset used for training.

The type of testing used is out-of-sample testing, where the model trained on a dataset is tested not only on a previously unexposed portion of that dataset but also to data recorded from a different environment than the environment in which the dataset used for training was recorded. This is different from domain adaptation/transfer learning where an algorithm trained on one particular domain is also tested and observed on a different domain [4]. When an algorithm that has been trained to identify a car in an image, is used to identify a truck in an image, it is domain adaptation. The algorithm has not previously seen what a truck is but still tries to identify a truck using the knowledge it acquired about cars. Out-of-sample is different from that in the aspect that an algorithm is trained to identify cars in a parking lot of a building, but is used to identify cars out on the open road. The point is to observe given an algorithm that is trained to identify/classify data in one environment, can it also work in a different environment. If it does what is the observed performance.

The most important observations made were that the performance of Siamese-like Convolutional Neural Network though not as good as Convolutional Neural Network in the test portion of the dataset, it was significantly better than the latter when tested on the dataset prepared by me. It was also observed that not all algorithms have the same consistency in the type of errors commonly generated. Another observation was made in terms of the time and resource utilization of each algorithm as they vary significantly and that Siamese-like Convolutional Neural Network might need human intervention for it to have a good performance.

In the upcoming chapters, we will be discussing the background of Fully Connected, Convolutional and Siamese-like Convolutional Neural Networks, implementations of the various algorithms used by us, the results observed from those implementations and the conclusions that were drawn from those results.

Chapter 2

Background

Fully Connected Neural Networks is a class of feed-forward artificial neural networks. It consists of at least three layers of nodes: an input layer, a hidden layer(s), and an output layer. Except for the input nodes, each node calculates a weighted sum of its inputs followed by a nonlinear activation function. In a Fully connected Neural Network each node in one layer is connected to all nodes in the next layer. Training utilizes a supervised gradient-descent technique called backpropagation.

Convolutional Neural Networks are regularized versions of Fully Connected Neural Networks, as the "fully-connectedness" of the regular Fully Connected Neural Networks makes them prone to overfitting data, and the Convolutional Neural Network reduces this. Usual ways of regularization include adding some form of weights to the loss function. However, Convolutional Neural Networks take a different approach towards regularization, where they take into consideration the hierarchical pattern in data and create more complex patterns from smaller and simpler patterns. Therefore, on the scale of connectivity and complexity, Convolutional Neural Networks are on the lower extremity.

Convolutional Neural Networks were inspired by biological processes that is the connectivity pattern between nodes resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

Convolutional Neural Network uses Kernel Convolution. It is a process where we take a small matrix of numbers (called kernel or filter), and pass it over an image and transform it based on the values from filter. Subsequent feature map values are calculated using (2.1).

$$\mathbf{G}[a,b] = (f * h)[a,b] = \sum_i \sum_j h[i,j] f[a-i, b-j] \tag{2.1}$$

where a and b indicate the rows and columns of the result matrix, h denotes the kernel, and f denotes the input image, similar to what is observed in [5].

After placing the filter over a selected pixel, it takes each value from the kernel and multiplies them in pairs with corresponding values from the image. Finally, it sums up everything and puts the result in the appropriate position in the output feature map.

For an input image of size $m \times m$ pixels, if a kernel of size $k \times k$ is used, the size of the feature map would be $(m-k+1) \cdot (m-k+1)$. As the image shrinks every time convolution is performed, convolution can only be performed a limited number of times, before the image disappears completely. Also, the impact of the pixels located on the outskirts is much smaller than those in the center of the image. To solve both these problems, padding can be used to add an additional border to the input image. Let us assume a padding of p pixel(s) is used on an input image of size m*m pixels, the input image size will become $(m+2p) \times (m+2p)$ and if the kernel size remains at $k \times k$, then the feature map size will be $(m+2p-k+1) \times (m+2p-k+1)$. If padding is used, then the equation is as given in (2.2)

$$p = \frac{k-1}{2} \tag{2.2}$$

where p is the padding and k is the filter dimensions, should always be satisfied. Padding should be chosen keeping in mind this equation and the filter size.

Stride is the number of pixels the filter shifts at a time. It is usually 1 but can be made 2 or more if we want less overlap in the receptive fields, or if we want smaller dimensions for the feature map. The dimensions of the output matrix considering an input image of size $m \times m$ pixels, kernel/filter of size $k \times k$, padding of p, and stride of s is given by (2.3).

$$n = \left\lfloor \frac{m+2p-f}{s} + 1 \right\rfloor \tag{2.3}$$

where n is the size of the output matrix in one dimension, i.e. output matrix is of the size $n \times n$.

If we want to use Convolutional Neural Networks with color images, or if we want to use multiple filters in a single layer, the concept of convolution over a volume needs to be used. An important thing to note is that when using convolution over a volume, the number of channels in the filter and the input image on which the filter is applied should be the same. The dimensions of the input tensor should follow (2.4)

$$[m, m, n_c] \times [k, k, n_c] = \left[\left\lfloor \frac{m + 2p - k}{s} + 1 \right\rfloor, \left\lfloor \frac{m + 2p - k}{s} + 1 \right\rfloor, n_f \right] \tag{2.4}$$

where m is the image size, i.e input image is of the size $m \times m$, k is the filter size i.e filter is of the size $k \times k$, n_c is the number of channels in the image, c is the padding used, s is the stride used, n_f is the number of filters.

Convolutional Neural Networks have a convolution layer that is responsible for forward propagation which consists of two steps. The first step is to calculate the intermediate value $\mathbf{Z}$, which is obtained as a result of the convolution of the input data from the previous layer with $\mathbf{W}$ tensor which contains the filters and then adding bias b. This is done using the equation (2.5). The second step is the application of a non-linear activation function, g, to the intermediate value. This is done using (2.6)

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} * \mathbf{A}^{[l-1]} + b^{[l]} \tag{2.5}$$

$$\mathbf{A}^{[l]} = g^{[l]}(Z^{[l]}) \tag{2.6}$$

where $\mathbf{Z}^{[l]}$ denotes the intermediate matrix at layer l, $\mathbf{W}^{[l]}$ denotes the weight matrix at layer l, $\mathbf{A}^{[l]}$ denotes the output data/matrix at layer l, $b^{[l]}$ denotes the bias at layer l, and $g^{[l]}$ denotes the activation layer at layer l.

A loss function is used to evaluate the performance of a Machine Learning model. For an input it gives out an output whose value if high (in case of maximize loss function) or low (in case of

minimize loss function) and will give an idea if the model is performing well. The loss function used will denoted as L.

For back-propagation, just like in any densely connected neural networks, derivatives are calculated using equations (2.7), (2.8) and (2.9). These derivatives are later used to update the values of the parameters in a process called gradient descent.

$$\mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} \tag{2.7}$$

$$db^{[l]} = \frac{\partial L}{\partial b^{[l]}} \tag{2.8}$$

$$\mathbf{dA}^{[l-1]} = \frac{\partial L}{\partial \mathbf{A}^{[l-1]}} \tag{2.9}$$

$\mathbf{dW}^{[l]}$ and $db^{[l]}$ are derivatives of the parameters of the current layer, and $\mathbf{dA}^{[l-1]}$ is the derivative of the input data of the previous layer. The derivative of the intermediate value $\mathbf{dZ}^{[l]}$ is obtained by applying a derivative of the activation function to the input tensor. Using chain rule we have (2.10)

$$\mathbf{dZ}^{[l]} = \mathbf{dA}^{[l]} g'(Z^{[l]}) \tag{2.10}$$

Backpropagation of convolution is obtained by a matrix operation called full convolution. This operation can be described by (2.11).

$$\mathbf{dA} = \mathbf{dA} + \sum_{m=0}^{n_{height}} \sum_{n=0}^{n_{width}} W * \mathbf{dZ}[m, n] \tag{2.11}$$

where the filter is denoted by $\mathbf{W}$, and $\mathbf{dZ}[m, n]$ is a scalar that belongs to a partial derivative obtained from the previous layer.

Apart from Convolution layers, Convolutional Neural Networks also use pooling layers to reduce the size of the tensor and speed up calculations. Pooling layers also need to have backpropagation.

In Figure 2.1, we can look at the typical structure of a Convolutional Neural Network.

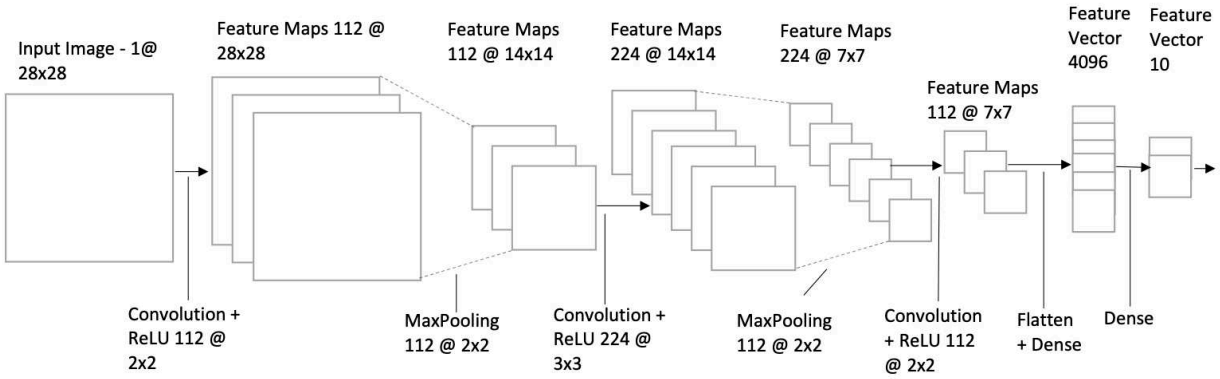


Figure 2.1: Typical structure of a Convolutional Neural Network model

In 1999, Yann LeCun, et al., [6] presented the idea of Convolutional Neural Networks and trained and tested it on various image-based datasets. They compare their algorithms with other algorithms in terms of error rates only.

In [7] published by Siddique, et al., in 2019 the authors compare different network models of Convolutional Neural Network with each other. The build models with different number of layers and hidden units, and compare with each other in terms of accuracy. This paper does help in understanding the performance of Convolutional Neural Network, and how network structure might have an impact on the performance, which we will use as an inspiration for my work to build and test multiple models before choosing the best, however they could have additionally provided other useful metrics like training time, consistency of the results.

A Siamese-like Neural Network is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. Often one of the output vectors are pre-computed, thus forming a baseline against which the other output vector is compared. Siamese-like Neural Network consists of two Neural Network models that take in two inputs and compares those two inputs with each other and tries to identify the similarity between the two. Siamese-like Neural Networks was first introduced in 1993 by Bromley, et al., [2] to solve signature verification as an image matching problem. In this paper they provided results

in terms of how many signatures were correctly identified as genuine or fake, and played around with the way comparisons are made by eliminating certain data points However, no comparison was made with the alternatives to observe the performance. Siamese-like Neural Network might be used in fields where similarity is used a measure, i.e. fields such as recognizing handwritten text, automatic detection of faces in pictures, and matching queries with indexed documents. Perhaps the most well-known application of Siamese-like Neural Networks is face recognition, where known images of people are pre-computed and compared to an image from public spaces such as parks, and stations.

Siamese-like Neural Networks is considered a one-shot learning algorithm [8] and they vary from the regular classification algorithms in the fact that the latter calculates probability distribution over all the classes that it was trained for. Therefore, if the model was not trained for a certain class, the model then needs to be retrained for that class, for it to be considered. The name one-shot learning algorithm comes from the nature of the algorithm where a new class can be added for prediction by just taking one shot of it and using that one shot for comparison for subsequent predictions.

The other advantage of one-shot learning algorithms such as the Siamese-like Neural Network is that they need relatively less data to learn [8]. It needs data just to calculate the degree of similarity between two inputs by finding features rather than trying to identify which class the input falls under.

In Figure 2.2 below, we can look at the typical structure of a Siamese-like Convolutional Neural Network.

The contrastive loss function mentioned in the figure is a typical loss function used in Siamese-like Convolutional Neural Network. It is a distance based loss function which tries to put semantically similar values closer together and different values are pushed apart. This helps in establishing the degree of similarity between two images. The formula typically used for contrastive loss function as mentioned in [9] is

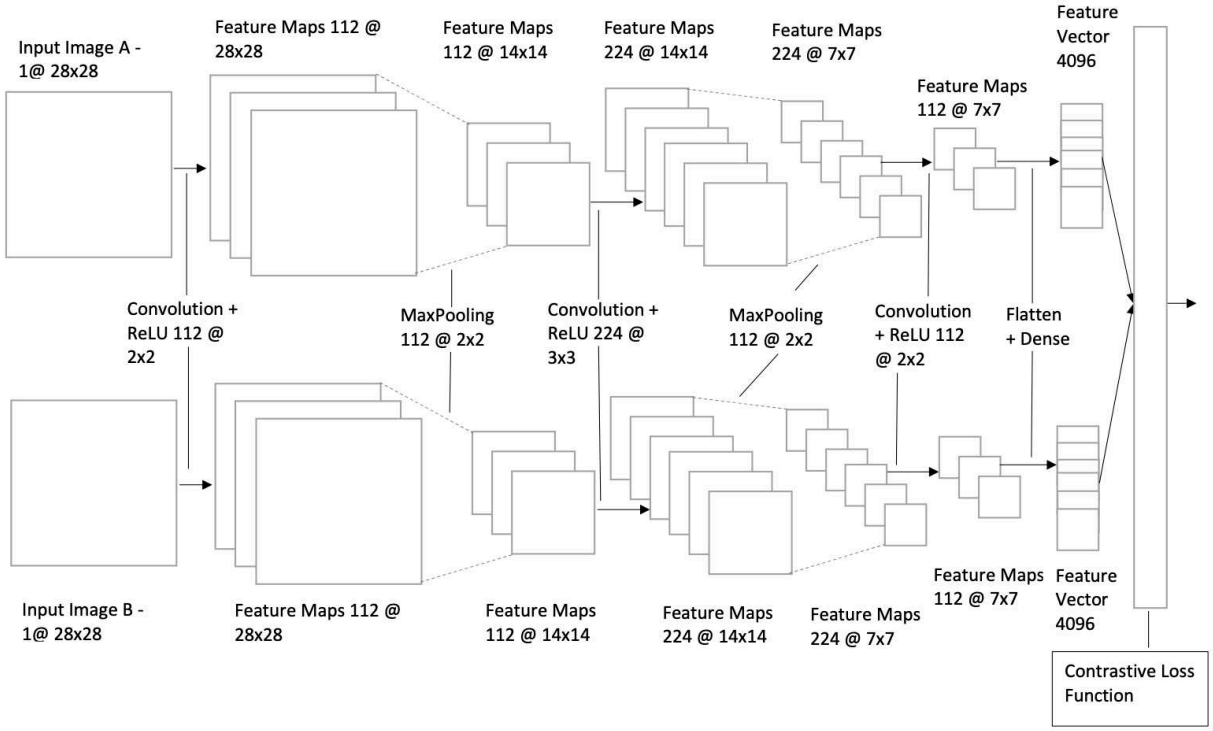


Figure 2.2: Typical structure of a Siamese-like CNN model

$$C = \frac{1}{2}((1-Y)(d_W)^2 + Y(max(0, m - d_W))^2) \tag{2.12}$$

where the Y is the binary label assigned to the pair of images, d_W is the distance between the outputs of the sister networks, usually a euclidean distance. m is margin which is a value greater than 0. Dissimilar pairs beyond the margin will not contribute to the loss.

In 2015, Gregory Koch, et al [8], tested Siamese-like Convolutional Neural Network over multiple datasets against multiple algorithms. The results were however compared in terms of accuracy between the algorithms in the test portion of the dataset and no other metrics were provided to support their claim that the Siamese-like Convolutional Neural Network was better than most of the other algorithms.

Very few papers compare Fully Connected and Convolutional Neural Networks. Syrine Ben Driss, et al., in 2017, in [1] compare Fully Connected and Convolutional Neural Network models for character recognition, on optical character recognition problem. The noticeable aspect is that the comparison was made in terms of accuracy and time. The time metric is not usually specified

in the papers, but this paper does and provides more information. However, two aspects where this paper falls short is that testing is done on the test portion of the dataset from which the training portion was obtained and the consistency of errors was not touched upon. This means that the standard deviation is low, the accuracy is high, the adaptable nature of the algorithm was never tested, and the tendency of the algorithms to generate False-Positives (Type I Error) or False-Negatives (Type II Error) was not tested.

Luca Di Persio, et al., in 2016 [10] compared Fully Connected and Convolutional Neural Network, in a stock price prediction problem. As is common in papers talking about comparison of two or more algorithms, this paper just talks about accuracy in terms of comparing the algorithms. No information about the time and resource utilization or the consistency was provided. The adaptable nature of the algorithms was also not tested as the test portion was again taken from the same dataset from which the training portion was taken, and the standard deviation remained low, keeping the accuracy high.

Chapter 3

Implementation

We have used the MNIST dataset(Modified National Institute of Standards and Technology dataset) which is a subset of a larger dataset made by NIST (National Institute of Standards and Technology) [11]. It is a supervised learning dataset of images of handwritten digits in 28×28 pixels resolution. It is a good dataset to test pattern learning and matching techniques as it represents real-world data and requires minimal efforts on pre-processing and formatting. It is commonly used by researchers to test the effectiveness of their algorithms or even see if their algorithm actually works. The other dataset that was used is EMNIST dataset (Extended Modified National Institute of Standards and Technology database) which is a supervised dataset of handwritten digits and characters in 28×28 pixels resolution [3]. In the paper [12] published by Baldominos, et al., in 2019 the authors stated that the latest algorithms generated are doing fairly well on MNIST dataset that it cannot be the only dataset to judge a performance of an algorithm. They tested certain algorithms on the EMNIST dataset, and observed that it is better are establishing the differences between in algorithms in terms of error rate which reflects on the accuracy of the algorithms. This is the reason for EMNIST dataset to be considered for this experiment. When the authors of that paper compared the performance of various algorithms, they only used accuracy as a metric. There are multiple sub datasets within EMNIST dataset. The sub datasets are:

- **Letters dataset** - It only has balanced sets of letters in it, both Upper and Lower case classified into 26 classes. We were using this dataset for our Convolutional Neural Network model.

- **Digits and MNIST dataset** - They are balanced handwritten digit datasets.

- **ByClass and ByMerge datasets** - They have an uneven number of images per class and there are more digits than letters. The number of letters roughly equate to the frequency of use in the English language.

- **Balanced dataset** - It tackles the unbalanced nature of ByClass and ByMerge datasets by reducing the misclassification errors due to capital and lower case letters by having a balanced number of upper and lower case letters along with digits in 47 classes. We were using this upper case images from this dataset for our experiment.

Three algorithms were used on these datasets. Fully Connected Neural Network, a rudimentary algorithm in image processing. Convolutional Neural Network, a well-established algorithm in the field of image processing with good results. Siamese-like Convolutional Neural Network that is a one-shot learning algorithm, which uses Convolutional Neural Networks but in a probabilistic manner.

In MNIST, we used 1000 entries of each class for training of Fully Connected and Convolutional Neural Network, and 999 true and false pairs of images were used for training of Siamese-like Convolutional Neural Network. 200 entries of each class is set aside for testing. In EMNIST, 2000 entries of each class were used for training of Fully Connected and Convolutional Neural Network and 1999 true and false pairs of images for training of Siamese-like Convolutional Neural Network. 500 entries of each class is set aside for testing.

Two datasets were prepared by me, first having data similar to MNIST dataset, where we have images of digits, and the second dataset having data similar to EMNIST dataset with images of alphabets in upper case. All these images have a 1:1 aspect ratio, but not necessarily a 28×28 pixels resolution. When using the image for prediction purposes, the images need to processed to bring it to a 28×28 pixels resolution. Also the images may not necessarily have the digit or the alphabet in the center but could be towards one side of the image. This is also taken care of in pre-processing done before prediction. The reason for not sticking to 28×28 pixels resolution and centering of the digit or the alphabet is to observe the impact of change in the environment and pre-processing of images on the accuracy of the algorithms. The data was collected by writing these digits and alphabets with different styles on a paper and then scanning the images using a phone camera and then cropping them with a 1:1 aspect ratio. There are 10 images for each class in both the prepared datasets.

Sample images from the MNIST and the EMNIST dataset can be seen in Figure 3.1a and Figure 3.1b respectively. Sample images to be predicted can be seen in Figure 3.2a and Figure 3.3a before pre-processing and in Figure 3.2b and Figure 3.3b after pre-processing for MNIST and EMNIST respectively.

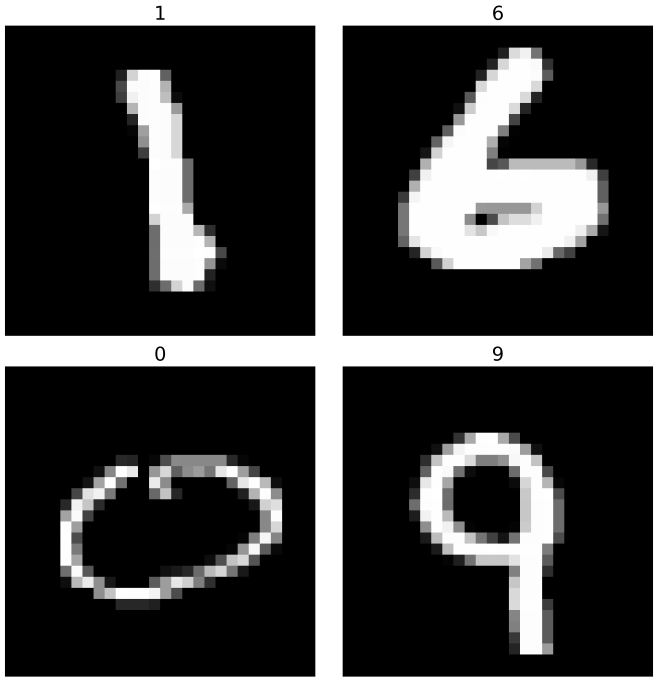
Fully Connected Neural Network used for training and testing in the MNIST dataset was built using the sklearn package. It was trained using 3 hidden layers with the first layer having **1000** hidden units, second layer having **500** hidden units, and the third layer having **250** hidden units, with the alpha value/learning rate being set to **0.00001**. Optimizer used was Adam [13] and the loss function used was a log loss function. This model took **98 seconds** for training and provided an accuracy of **97.05%** on average. This model was chosen among the many models that were tested and compared, as the time taken for training was not too high, and a good accuracy was achieved. Some of the models that were tested are as seen in Table 3.1.

Table 3.1: Other Fully Connected Neural Network models trained and tested on MNIST dataset.

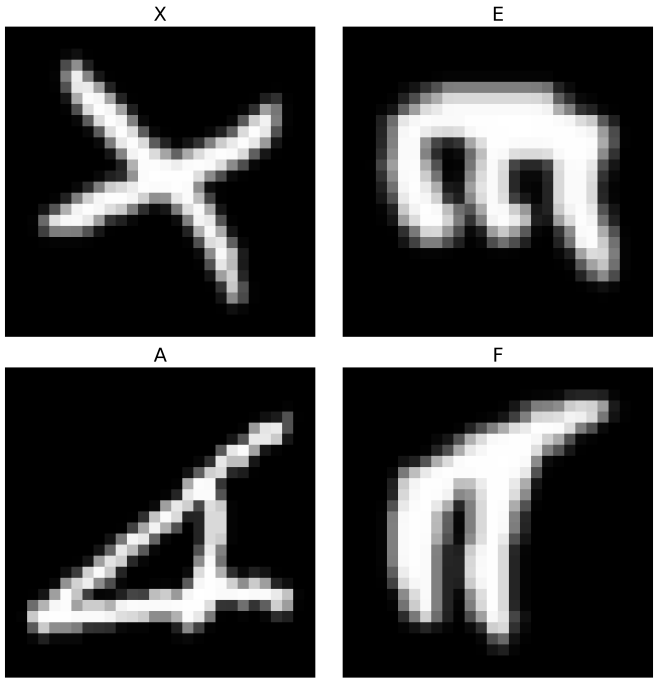
	Parameters for the model				Metrics for the model	
	Layer 1	Layer 2	Layer 3	Learning Rate	Accuracy(%)	Time for Training (seconds)
1	500	250	125	0.00001	96.89	64
2	1500	1000	500	0.00001	96.03	151
3	1000	500	250	0.0001	96.5	64
4	1000	500	250	0.000001	95.95	83

The values listed in each of the layers columns is the number of hidden units in that layer.

Convolutional Neural Network used for training and testing in the MNIST dataset was built using Keras [14] package. The Convolutional Neural Network structure used consisted of 5 layers. The first layer had **112** Convolutional 2D hidden units with LeakyReLU(Leaky REctified Linear Units), kernel size of **(2,2)** and a Maxpooling2D of (2,2), the second layer had **224** Convolutional 2D hidden units with LeakyReLU, kernel size of (3,3) and a Maxpooling2D of (2,2), and the third layer has **112** Convolutional 2D hidden units with LeakyReLU, kernel size of (2,2) and a

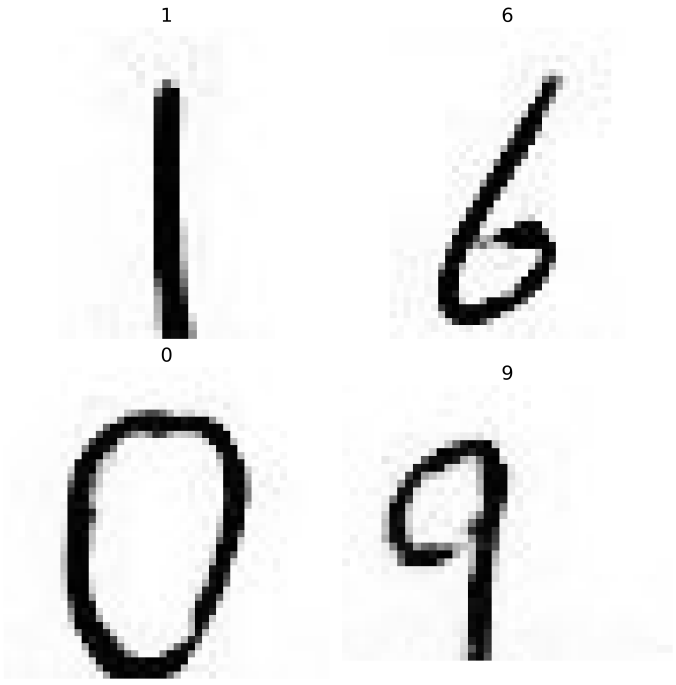


(a) Sample images from MNIST dataset.

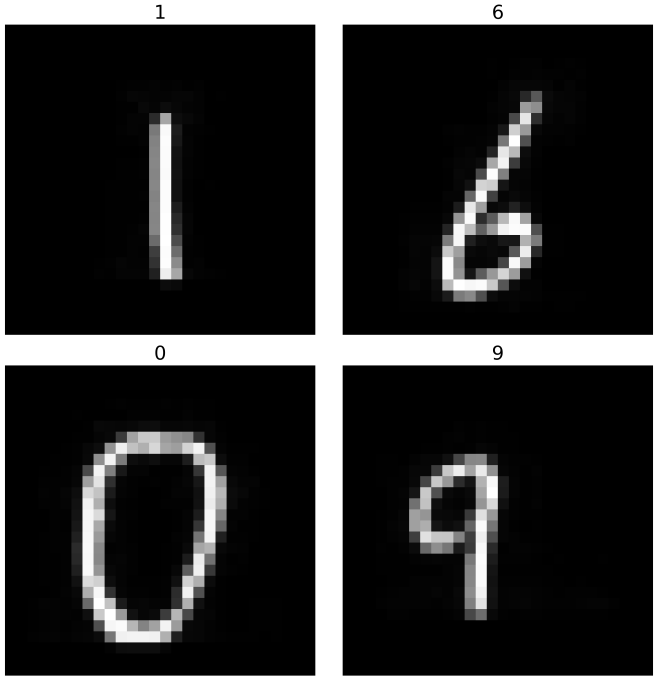


(b) Sample images from EMNIST dataset.

Figure 3.1: Sample images sent for prediction from prepared dataset before and after pre-processing



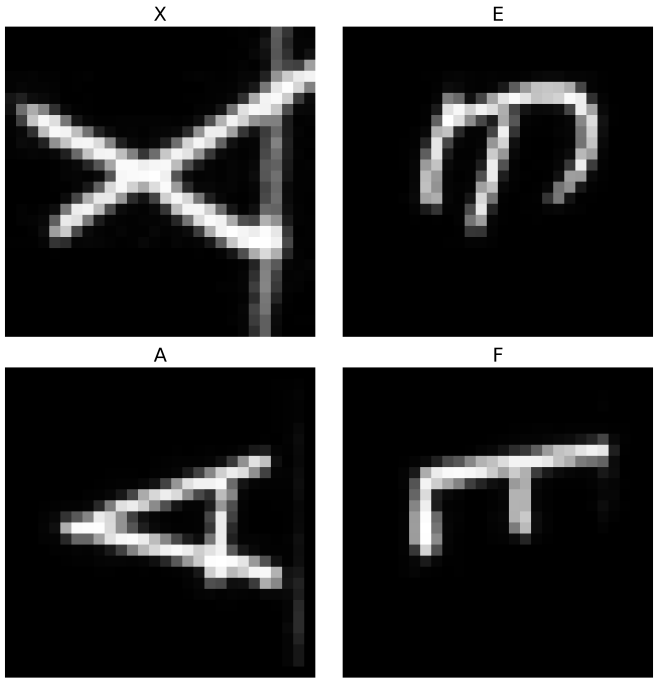
(a) Sample images from prepared dataset to be predicted before pre-processing.



(b) Sample images from prepared dataset to be predicted after pre-processing.

Figure 3.2: Sample images sent for prediction from prepared dataset before and after pre-processing for MNIST

(a) Sample images from prepared dataset to be predicted before pre-processing.



(b) Sample images from prepared dataset to be predicted after pre-processing.

Figure 3.3: Sample images sent for prediction from prepared dataset before and after pre-processing for EMNIST

Maxpooling2D of (2,2). These units were Convolutional 2D units. The layers of Convolutional 2D units were followed by a flatten function and then two layers of Dense units, with the first layer having **4096** units with sigmoid activation function and the second layer having 10 units with softmax activation function. The learning rate used was **0.0001** and the optimizer used is Adam. Adam optimizer was chosen over Stochastic Gradient Descent(SGD) due to the better accuracy provided. The loss function used was Categorical Cross Entropy function. The model was trained for 25 epochs with a default batch size of 32. This model took **2182 seconds** for training(rounded off) and provided an accuracy of **98.85%** on average. This specific model was chosen among the many models that were tested and compared, as the time taken for training was not too high, with good accuracy being achieved. Some of the other models that were tested are as seen in Table 3.2.

Table 3.2: Other Convolutional Neural Network models trained and tested on MNIST dataset.

	Parameters for the model						Metrics for the model	
	Conv. Layer 1	Conv. Layer 2	Conv. Layer 3	Dense Layer 1	Dense Layer 2	Learning Rate	Accuracy(%)	Time for training (seconds)
1	56 (2,2)	112 (3,3)	X	4096 sigmoid	10 softmax	0.0001	97.80	6891
2	224 (2,2)	448 (3,3)	224 (2,2)	4096 sigmoid	10 softmax	0.0001	98.9	4981
3	224 (3,3)	448 (3,3)	224 (2,2)	4096 sigmoid	10 softmax	0.0001	98.35	4959
4	224 (3,3)	224 (3,3)	224 (2,2)	4096 sigmoid	10 softmax	0.0001	98.75	4939
5	112 (2,2)	224 (3,3)	112 (2,2)	4096 sigmoid	10 softmax	0.00001	96.25	1981
6	112 (2,2)	224 (3,3)	112 (2,2)	1024 sigmoid	10 softmax	0.0001	98.9	1459

In the above table each convolutional layer is listed using the format:

Conv2D units

Kernel Size

Each convolutional layer uses a Maxpooling2D of (2,2),and uses LeakyReLU as the activation function. After all the convolutional layers we have a flatten function which is followed by dense layers. Each dense layer is listed using the format:

Dense units

Activation function

In the table above, the second model despite providing a higher accuracy than the chosen model, was not selected as the time taken for training in comparison to the chosen model was way higher and the gain in accuracy was minimal.

Siamese-like Convolutional Neural Network used for training and testing in the MNIST dataset was also built using Keras package. For this purpose, two networks working hand in hand are needed to identify the amount of similarity between two images, each supplied to one of the networks. The structure of both networks is the same. The structure of an individual neural network in the Siamese-like Convolutional Neural Network consists of 4 layers. The first layer had **112** of Convolutional 2D hidden units with ReLU(REctified Linear Units), kernel size of **(2,2)** and a Maxpooling2D of (2,2), the second layer had **224** Convolutional2D hidden units with ReLU(REctified Linear Units), kernel size of (3,3) and a Maxpooling2D of (2,2), and the third layer has **112** Convolutional2D hidden units with ReLU(REctified Linear Units), kernel size of (2,2). These layers of Convolutional 2D units were followed by a flatten function and then one layer of **1024** dense units with sigmoid activation function. Unlike the structure used in Convolutional Neural Network used above, the structure in the Siamese-like Convolutional Neural Network does not have the last layer of 10 dense units. The learning rate used was **0.0001** and the optimizer used is Adam. Adam optimizer was chosen over Stochastic Gradient Descent(SGD) due to the better accuracy provided. The loss function used was contrastive loss function. The model was trained for 25 epochs with a batch size of 128. This model took **2467 seconds** for training(rounded off) and provided an accuracy of **98.01%** on average. Some of the other models that were tested are as seen in Table 3.3.

Fully Connected Neural Network used for training and testing in the EMNIST dataset was built using the sklearn package. It was trained using 3 hidden layers with the first layer having **500**

Table 3.3: Other Siamese-like Convolutional Neural Network models trained and tested on MNIST dataset.

	Parameters for the model					Metrics for the model	
	Conv. Layer 1	Conv. Layer 2	Conv. Layer 3	Dense Layer	Learning Rate	Accuracy(%)	Time for training (seconds)
1	112 (2,2)	224 (3,3)	X	4096 sigmoid	0.0001	97.21	4096
2	224 (2,2)	448 (3,3)	224 (2,2)	4096 sigmoid	0.0001	97.66	8201
3	112 (2,2)	224 (3,3)	112 (2,2)	4096 sigmoid	0.0001	97.61	2826
4	224 (3,3)	448 (3,3)	224 (2,2)	4096 sigmoid	0.0001	97.93	8229
5	224 (2,2)	224 (3,3)	224 (2,2)	4096 sigmoid	0.00001	97.88	5934
6	112 (2,2)	224 (3,3)	112 (2,2)	4096 sigmoid	0.00001	95.64	2939
7	28 (3,3)	56 (3,3)	28 (2,2)	1024 sigmoid	0.0001	97.01	446

hidden units, the second layer having **250** hidden units, and the third layer having **125** hidden units, with the alpha value/learning rate being set to **0.00001**. Optimizer used was Adam and the loss function used was a log loss function. This model took **224 seconds** for training and provided an accuracy of **89.84%** on average. This model was chosen among the many models that were tested and compared, as the time taken for training was not too high, with good accuracy being achieved. Some of the models that were tested are as seen in Table 3.4.

Convolutional Neural Network used for training and testing in the EMNIST dataset was built using Keras package. The Convolutional Neural Network structure used consisted of 4 layers. The first layer had **112** Convolutional 2D hidden units with LeakyReLU(Leaky REctified Linear Units), kernel size of **(3,3)** and a Maxpooling2D of (2,2), the second layer had **224** Convolutional 2D hidden units with LeakyReLU, kernel size of (3,3) and a Maxpooling2D of (2,2), and the third layer has **112** Convolutional 2D hidden units with LeakyReLU, kernel size of (3,3) and a

Table 3.4: Other Fully Connected Neural Network models trained and tested on EMNIST dataset.

	Parameters for the model					Metrics for the model	
	Layer 1	Layer 2	Layer 3	Layer 4	Learning Rate	Accuracy(%)	Time for Training (seconds)
1	1500	1000	500	X	0.00001	88.33	761
2	1000	500	250	X	0.00001	88.87	328
3	1000	500	250	X	0.0001	89.31	401
4	1000	500	250	X	0.000001	89.84	288
5	1000	500	250	125	0.000001	89.84	342
6	1000	500	X	X	0.000001	87.37	173

Maxpooling2D of (2,2). These units were Convolutional 2D units. The layers of Convolutional 2D units were followed by a flatten function and then two layers of Dense units, with the first layer having **1024** units with a linear activation function and the second layer having 27 units with softmax activation function. The learning rate used was **0.0001** and the optimizer used is Adam. Adam optimizer was chosen over Stochastic Gradient Descent(SGD) due to the better accuracy provided. The loss function used was Categorical Cross Entropy function. The model was trained for 25 epochs with a default batch size of 32. This model took **5900 seconds** for training(rounded off) and provided an accuracy of **94.90%** on average. Some of the other models that were tested are as seen in Table 3.5

Siamese-like Convolutional Neural Network used for training and testing in the EMNIST dataset was also built using Keras package. For this purpose, two networks working hand in hand are needed to identify the amount of similarity between two images, each supplied to one of the networks. The structure of both networks is the same. The structure of an individual neural network in the Siamese-like Convolutional Neural Network consists of 4 layers. As the images of letters at a low resolution of 28×28 pixels can lead to the wrong classification of images of similar-looking alphabets such as 'D' and 'O', or 'A' and 'H', we are measuring by considering two different accuracies. The first accuracy is when the algorithm only considers one best match. If the prediction for an is the expected class, we increment the count of correctly predicted images

Table 3.5: Other Convolutional Neural Network models trained and tested on EMNIST dataset.

	Parameters for the model						Metrics for the model	
	Conv. Layer 1	Conv. Layer 2	Conv. Layer 3	Dense Layer 1	Dense Layer 2	Learning Rate	Accuracy(%)	Time for training (seconds)
1	32 (3,3)	64 (3,3)	128 (3,3)	128 linear	27 softmax	0.0001	94.36	1274
2	112 (3,3)	224 (3,3)	112 (3,3)	128 linear	27 softmax	0.0001	94.08	5175
3	112 (3,3)	224 (3,3)	X	224 linear	27 softmax	0.0001	93.87	4802
4	224 (3,3)	448 (3,3)	224 (3,3)	128 linear	27 softmax	0.0001	94.53	16834
5	112 (3,3)	112 (3,3)	112 (3,3)	128 linear	27 softmax	0.0001	94.42	3914
6	112 (2,2)	224 (3,3)	112 (3,3)	128 linear	27 softmax	0.0001	94.78	5217
7	112 (3,3)	224 (3,3)	112 (3,3)	128 linear	27 softmax	0.00001	93.37	7585

by one, else, we do not increment the count. The second accuracy is when the algorithm gives out the best 3 predictions for an image, and if the actual class of the image is one among the 3, then the count of correctly predicted images is incremented by one, else, we do not increment the count. We will call them 'Single Prediction Accuracy' and 'Top Five Prediction Accuracy' from now on.

The first layer had **112** of Convolutional 2D hidden units with ReLU(REctified Linear Units), kernel size of **(3,3)** and a Maxpooling2D of (2,2), the second layer had **224** Convolutional2D hidden units with ReLU(REctified Linear Units), kernel size of (3,3) and a Maxpooling2D of (2,2), and the third layer has **112** Convolutional2D hidden units with ReLU(REctified Linear Units), kernel size of (3,3). These layers of Convolutional 2D units were followed by a flatten function and then one layer of **4096** dense units with sigmoid activation function. Unlike the structure used in Convolutional Neural Network used above, the structure in the Siamese-like Convolutional Neural Network does not have the last layer of 27 dense units. The learning rate used was **0.0001** and the

optimizer used is Adam. Adam optimizer was chosen over Stochastic Gradient Descent(SGD) due to the better accuracy provided. The loss function used was contrastive loss function. The model was trained for 25 epochs with a batch size of 128. This model took **16990 seconds** for training(rounded off) and provided a Single Prediction Accuracy of **62.62%** and Top Five Prediction Accuracy of **98.78%** on average. This specific model was chosen among the many models that were tested and compared, as the time taken for training was not too high, with a good accuracy being achieved. Some of the other models that were tested are as seen in Table 3.6

Table 3.6: Other Siamese-like Convolutional Neural Network models trained and tested on EMNIST dataset.

	Parameters for the model					Metrics for the model		
	Conv. Layer 1	Conv. Layer 2	Conv. Layer 3	Dense Layer	Learning Rate	Single Prediction Accuracy (%)	Top Five Prediction Accuracy (%)	Time for training (seconds)
1	56 (2,2)	112 (3,3)	56 (2,2)	1024 sigmoid	0.0001	44.18	96.78	6024
2	112 (2,2)	224 (3,3)	X	1024 sigmoid	0.0001	40.96	94.90	17361
3	224 (2,2)	448 (3,3)	224 (2,2)	1024 sigmoid	0.0001	62.85	93.83	46463
4	112 (2,2)	224 (3,3)	112 (2,2)	4096 sigmoid	0.0001	59.13	97.75	23063
5	112 (2,2)	112 (3,3)	112 (2,2)	1024 sigmoid	0.0001	55.00	97.51	12259
6	112 (2,2)	224 (3,3)	112 (2,2)	1024 sigmoid	0.00001	36.00	82.46	13628
7	112 (3,3)	224 (3,3)	112 (2,2)	1024 sigmoid	0.0001	51.54	97.53	11745
8	56 (2,2)	112 (3,3)	224 (2,2)	1024 sigmoid	0.0001	55.31	97.77	5975

There are three versions of the Siamese-like Convolutional Neural Network. The first version of the Siamese-like Convolutional Neural Network uses only one control image from each class to compare the image to be predicted with. The second version that we refer to as Averaged Siamese-

like Convolutional Neural Network uses three control images from each class to compare the image to be predicted with. The final version of the Siamese-like Convolutional Neural Network uses the same concept of the Averaged Siamese-like Convolutional Neural Network of using three control images but uses three threads to parallelize the comparison and prediction with each control image. This version is referred to as the Multi-threaded Siamese-like Convolutional Neural Network.

Subsequent measures of accuracy for all versions of Siamese-like Convolutional Neural Network are only Top Five Prediction Accuracies. All the comparisons made between all the algorithms are using the Top Five Prediction Accuracies.

The testing is done in three phases. In the first phase, we compare the results of the three algorithms over the training portion of the data. The second phase of testing involves testing over the test portion, the portion from the dataset on which the three algorithms were not trained on. The final phase of testing is done over a dataset prepared by us, which has similar data as from the dataset, but the environmental factors might change. The last phase of testing is done to see the adaptability of the three algorithms.

Chapter 4

Results

The testing phase provided some insightful results. The comparisons we discuss below are based on the parameters of accuracy, consistency, time and resource utilization, and availability of human intervention.

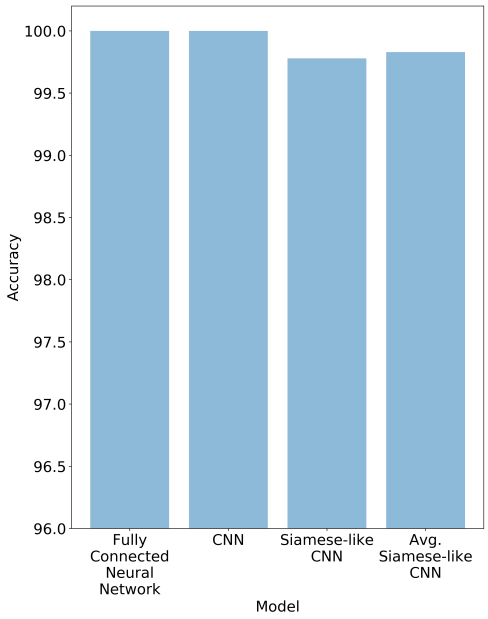
The first observation is the drop in the accuracy of Fully Connected Neural Network when compared in terms of accuracy it delivers on the training portion of both MNIST and EMNIST dataset, the testing portion of the MNIST and EMNIST dataset and the prepared dataset, as seen in Table 4.1 and Table 4.2 and in Figure 4.1, Figure 4.2, Figure 4.3, and Figure 4.4 . Despite the dataset we prepared having been normalized before prediction, the change in environmental factors in which the image was recorded, such as the lighting conditions, the angle of the camera with respect to the paper, the thickness of the ink, and more importantly the slight deviations in the position of the digit or the character within the image, affected the accuracy of the Fully Connected Neural Network on the prepared dataset. This does not reflect well upon the adaptable nature of the Fully Connected Neural Network algorithm. This points to the fact that in the case of image processing the Fully Connected Neural Network algorithm would not be a good choice if the environmental factors change between training and prediction time. However, there are applications to it in the field of image processing. If used in a photocopier machine to scan a paper and convert the handwritten text to a text file, or if used to convert a hardcopy of a printed paper, to a digital copy of the document, as the variations in such scenarios are limited, as the amount of light, angle of the camera, the pattern of the printed font, remains the same, the variance is reduced. As the training time and the resource utilization of Fully Connected Neural Network is low, it could be a good choice in such a field, or in field where accuracy is not the most important parameter.

Table 4.1: Average of observed accuracies of various algortihms over MNIST dataset

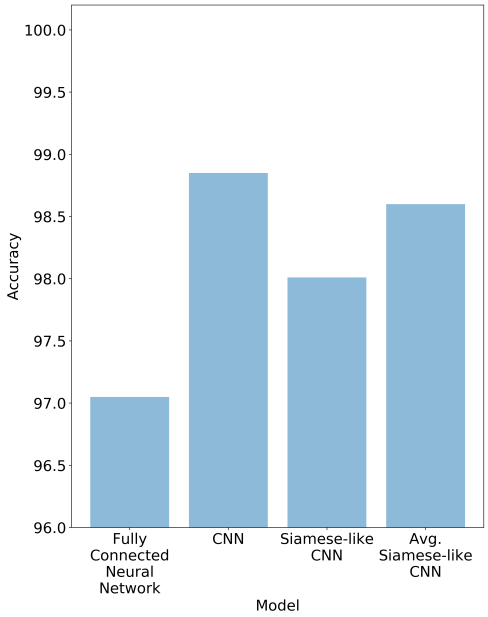
	Fully Connected Neural Network	Conv. Neural Network	Siamese -like Conv. Neural Network	Averaged Siamese -like CNN	Averaged Siamese -like CNN with base images from prepared dataset
Accuracy on Training Portion	100	100	99.78	99.83	X
Accuracy on Testing Portion	97.05	98.85	98.01	98.60	X
Accuracy on prepared dataset	59.00	88.00	80.00	82.00	91.00

Table 4.2: Average of observed accuracies of various algortihms over EMNIST dataset

	Fully Connected Neural Network	Conv. Neural Network	Siamese -like Conv. Neural Network	Averaged Siamese -like CNN	Averaged Siamese -like CNN with base images from prepared dataset
Accuracy on Training Portion	93.63	98.88	99.47	99.75	X
Accuracy on Testing Portion	87.82	94.80	97.82	98.42	X
Accuracy on prepared dataset	64.23	87.69	89.16	90.00	95.38

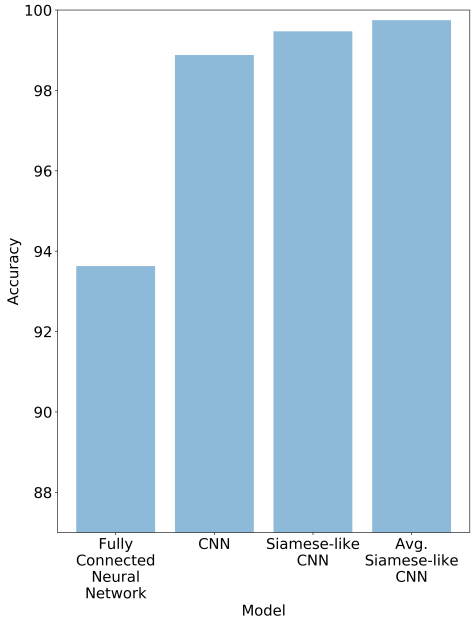


(a) Accuracy on train portion.

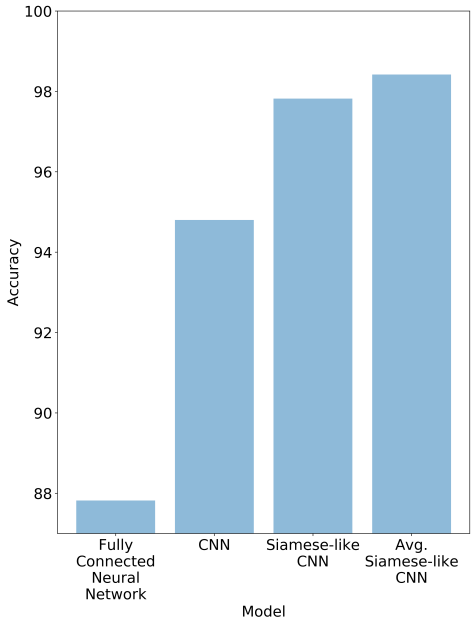


(b) Accuracy on test portion.

Figure 4.1: Comparison of the accuracy of various models.(a). Accuracy of Fully Connected Neural Network, Convolutional Neural Network, Siamese-like CNN models and Siamese-like CNN models with three control images on the train portion of the MNIST dataset. (b) Accuracy of Fully Connected Neural Network, Convolutional Neural Network, Siamese-like CNN models and Siamese-like CNN models with three control images on the test portion of the MNIST dataset.



(a) Accuracy on train portion Letters.



(b) Accuracy on test portion.

Figure 4.2: Comparison of the accuracy of various models.(a). Accuracy of Fully Connected Neural Network, Convolutional Neural Network, Siamese-like CNN models and Siamese-like CNN models with three control images on the train portion of the EMNIST dataset. (b) Accuracy of Fully Connected Neural Network, CNN, Siamese-like CNN models and Siamese-like CNN models with three control images on the test portion of the EMNIST dataset.

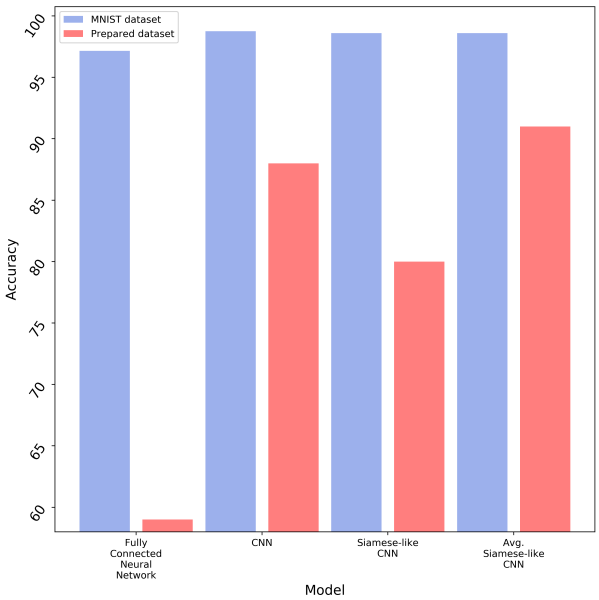


Figure 4.3: Accuracy of Fully Connected Neural Network, CNN, Siamese-like CNN models and Siamese-like CNN models with three control images on the test portion of MNIST dataset compared to prepared dataset, where the base images used are from the prepared dataset.

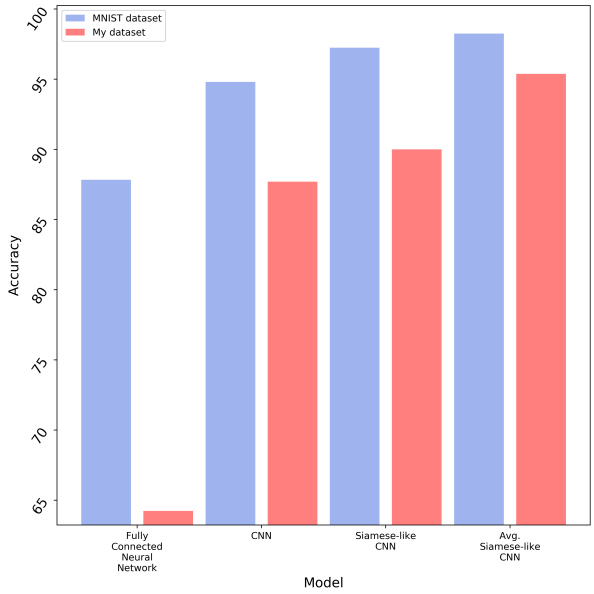


Figure 4.4: Accuracy of Fully Connected Neural Network, Convolutional Neural Network, Siamese-like CNN models and Siamese-like CNN models with three control images on the test portion of EMNIST dataset compared to prepared dataset, where the base images used are from the prepared dataset.

The second and very important observation was made when comparing the accuracy of Convolutional Neural Network and Averaged Siamese-like Convolutional Neural Network on the MNIST data in Table 4.1 and Figure 4.1. The accuracy of the Convolutional Neural Network was higher compared to Averaged Siamese-like Convolutional Neural Network when compared over the training portion of the MNIST dataset and the testing portion of the MNIST dataset. So, on paper, Convolutional Neural Network would seem like a good choice in comparison to Averaged Siamese-like Convolutional Neural Network. However, if we observe the accuracy of the Convolutional Neural Network and Averaged Siamese-like Convolutional Neural Network when tested on the dataset prepared by us, we notice that the drop in the accuracy of the Convolutional Neural Network is way higher than the drop of the Siamese-like Convolutional Neural Network. The Averaged Siamese-like Convolutional Neural Network performs better than the Convolutional Neural Network in the test dataset prepared by us. This is due to the probabilistic nature of the Averaged Siamese-like Convolutional Neural Network where it compares the image to be predicted with a control image and gives the amount of similarity/dissimilarity. Now if we observe the accuracy of the Siamese-like Convolutional Neural Network against Averaged Siamese-like Convolutional Neural Network and Convolutional Neural Network, the drop in Siamese-like Convolutional Neural Network is high enough to still keep it under-performing in comparison to Convolutional Neural Network. The Averaged Siamese-like Convolutional Neural Network is performing better due to the multiple comparison, which allows for more deviation in the control data. The data in MNIST has less standard deviation due to little or no change in the environmental factors and lack of very varied handwriting. As the Averaged Siamese-like Convolutional Neural Network does not just try to give a definite answer in terms of what digit/character is present in the picture, and it gives the degree of similarity/dissimilarity, it tends to be more reliable. We can see that the adaptable nature of the Averaged Siamese-like Convolutional Neural Network is much better than that of regular Convolutional Neural Network. Due to this, Averaged Siamese-like Convolutional Neural Network make more sense in applications where the environment in which the data will be recorded,

keeps changing by a significant amount, that is standard deviation is high. Convolutional Neural Network can keep up with the changes only if the changes are smaller.

The third observation was made when comparing the accuracy of Convolutional Neural Network and Averaged Siamese-like Convolutional Neural Network on the EMNIST data in Table 4.2 and Figure 4.2. We see that the Convolutional Neural Network performs better than Siamese-like Convolutional Neural Network when we only consider the Single Prediction Version of the Siamese-like Convolutional Neural Network. However when we compare it to the Top Five Prediction version of the Siamese-like Convolutional Neural Network, the latter provides better results. The reason the Single Prediction performs so low when we have established that Siamese-like Convolutional Neural Network provides good results in the MNIST dataset is that unlike the data in the MNIST dataset, data in the EMNIST dataset has more similarities. 'A' looks similar to 'R' and 'H', 'K' looks similar to 'X'. And there are even some images which might not seem like obvious matches but tend to be. We can see one such example between 'N' and 'W'. These similarities lead to a higher wrong top Single Prediction Accuracy. When we consider the top five predictions provided by the Top Five Prediction version of the Siamese-like Convolutional Neural Network, we get a better result. This is possible due to the way Siamese-like Convolutional Neural Network works by providing a degree of similarity between two images. Convolutional Neural Network only provides the best match as the result not a list of possible matches. This plays in advantage of Siamese-like Convolutional Neural Network. However, this advantage is only useful in fields where a possible human intervention can be made should it be needed.

The fourth observation was made in terms of the flexibility of the Siamese-like Convolutional Neural Network. When we compare Figure 4.3 and Figure 4.4 with Figure 4.5 and Figure 4.6 respectively, we notice that when we use the images from the MNIST and EMNIST dataset as the base images to make comparisons with, the accuracy obtained from both Siamese-like Convolutional Neural Network and Averaged Siamese-like Convolutional Neural Network versions were lesser than when the images from the prepared dataset were used as the base images. This is because with images being from the prepared environment the difference in the environment

when compared to the images to be predicted is less. Siamese-like Convolutional Neural Network provides the ability to adapt the algorithm to the target environment it will be deployed. So if a small sample can be collected from the target environment and can be stored as the base samples to be compared, Siamese-like Convolutional Neural Network or similar one-shot learning algorithms have the ability to provide better results.

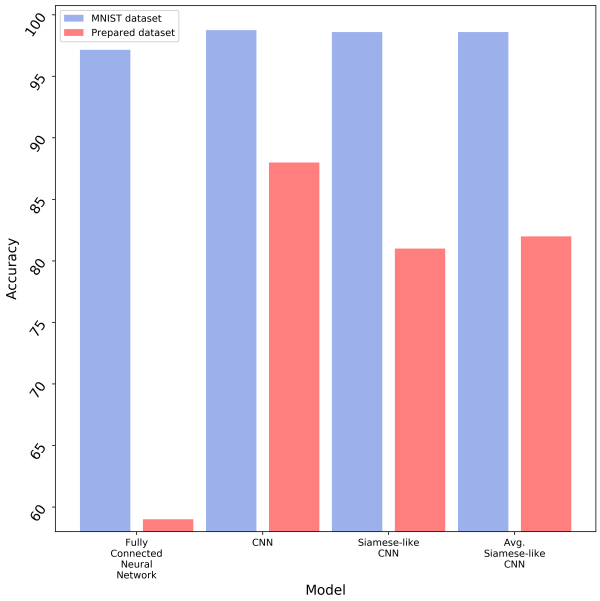


Figure 4.5: Accuracy of Fully Connected Neural Network, CNN, Siamese-like CNN models and Siamese-like CNN models with three control images on the test portion of MNIST dataset compared to prepared dataset, where the base images used are from the MNIST dataset.

The fifth observation was made in terms of consistency. There are four possible outcomes from an algorithm working on a dataset with two classes. True-Positives, True-Negatives, False-Positives (Type I error) and False-Negatives (Type II error). Ideally, we expect the algorithm to give only True-Positives and True-Negatives as the outcome for every prediction. But in reality, that cannot be achieved yet. Some algorithms give a balanced number of False-Positives (Type I error) and False-Negatives (Type II error), while some give a greater number of False-Positives (Type I error) over False-Negatives (Type II error) and the rest give a greater number of False-Negatives (Type II error) over False-Positives (Type I error). Depending on the business model and the application in which Machine Learning will be used, the ratio of False-Positives (Type

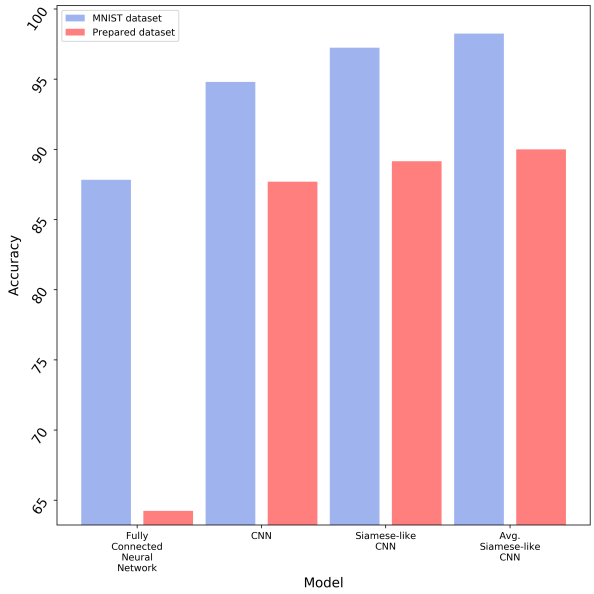


Figure 4.6: Accuracy of Fully Connected Neural Network, Convolutional Neural Network, Siamese-like CNN models and Siamese-like CNN models with three control images on the test portion of EMNIST dataset compared to prepared dataset, where the base images used are from the EMNIST dataset.

I error) to False-Negatives (Type II error) can have a huge impact. Let's look at the following examples to see the impact of consistency of False-Positives (Type I error) and False-Negatives (Type II error).

1. **Use of Machine Learning in medical diagnosis** - When using Machine Learning in medical diagnosis, both False-Positives (Type I error) and False-Negatives (Type II error) are undesirable outcomes. As one could lead to diagnosing a present medical condition as not present and not be treated while the other will lead to treatment even when the medical condition is not present. Sometimes treating a patient for a non-existing medical condition can be fatal. In 2017 [15] by Parampreet, et al., showed the importance of Type-I and Type-II errors in the field of medical diagnosis and they stated that both types of errors can be dangerous in the field of medical science.

2. **Use of Machine Learning for Face Recognition Locking mechanism** - The latest technological advancement in securing a device using someone's identity has been to use their face

as a password. This involves using an RGB picture along with a 3D Depth map to identify the person. Machine Learning is being implemented to identify subtle changes in facial appearance such as the growth of hair, beard, change in hairstyle, etc. In such an application, False-Positives (Type I error) is way more dangerous than False-Negatives (Type II error), as providing access to the device by wrongly predicting a non-owner as an owner could lead to loss of device or the information within the device. However, failing to unlock the device for the rightful owner leads to repeated attempts to unlock the device or use of secondary authentication. This is a slight inconvenience but not a risky outcome.

3. **Use of Machine Learning in threat detection in a security system** - Machine Learning is being used to identify any security holes in a system where security is of utmost importance. In such a case False-Negatives (Type II error) is way more dangerous than False-Positives (Type I error) as not identifying a security risk due to a False-Negative (Type II error) means that the required measures needed to rectify that risk are never taken. In 2014, Junoven, et al., in their paper [16] showed what a positive case is and a what a negative case is and the type of errors were discussed. From that paper we can understand that Type-I error is a false alarm, but Type-II error is a missed threat. Sensitivity needs to be high, i.e. real threats are detected and Type-II errors are maintained as low as possible. If a security risk is identified despite it not being present, it would only lead to a false alarm which is only a matter of inconvenience.

In the case of an algorithm working with more than two classes as in the datasets used by us, we do not of the type of errors as shown above. We understand the consistency by looking into the confusion matrix. The observation is made in terms of what were the predictions for a given image of a class. We can identify the accuracy of an algorithm by considering this matrix and three variables, precision, recall, f-1 score. Precision gives us an idea about how many of the positive predictions were actually positive. Recall gives us an idea about how many positive predictions were made out of all the actual positive cases provided. f-1 score is used we need a balance of both

precision and recall in a single equation, where the concern lies with both - if all positives are truly positive and all true positives are identified.

The formulae for precision, recall and f-1 score are as given in (4.1), (4.2), and (4.3) respectively.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{4.1}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{4.2}$$

$$f - 1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.3}$$

We compute the precision and recall for each class individually for an algorithm. As in the case of two class predictions we discussed above, the desired value of precision and recall depends on the specific application, however a higher F-1 score is always the desired quality. For facial recognition like software we need higher precision and for threat detection, we need higher recall. Confusion matrices for Fully Connected Neural Network, Convolutional Neural Network, Siamese-like Convolutional Neural Network and Averaged Siamese-like Convolutional Neural Network with base images from the prepared dataset are shown in Table 4.3, Table 4.4, Table 4.5, and Table 4.6 respectively for MNIST and Table 4.11, Table 4.12, Table 4.13, and Table 4.14 respectively for EMNIST datasets. Precision, Recall, and F-1 score for Fully Connected Neural Network, Convolutional Neural Network, Siamese-like Convolutional Neural Network and Averaged Siamese-like Convolutional Neural Network with base images from the prepared dataset are shown in Table 4.7, Table 4.8, Table 4.9, and Table 4.10 for MNIST, and in Table 4.15, Table 4.16, Table 4.17 and Table 4.18 for the EMNIST dataset.

Table 4.3: Confusion matrix for the Fully Connected Neural Network model over prepared dataset with MNIST data.

		Actual									
		0	1	2	3	4	5	6	7	8	9
Prediction	0	9	0	0	0	0	4	0	0	0	0
	1	0	7	0	0	0	0	0	0	0	0
	2	0	0	3	0	0	0	0	1	0	0
	3	0	0	0	7	0	0	0	0	0	0
	4	0	0	1	0	3	0	1	1	1	3
	5	0	0	2	3	1	5	1	0	3	0
	6	0	3	0	0	3	1	6	0	0	1
	7	0	0	0	0	0	0	0	8	0	0
	8	0	0	4	0	1	0	2	0	6	1
	9	1	0	0	0	2	0	0	0	0	5

Table 4.4: Confusion matrix for the Convolutional Neural Network model over prepared dataset with MNIST data.

		Actual									
		0	1	2	3	4	5	6	7	8	9
Prediction	0	10	0	0	0	0	0	0	0	0	0
	1	0	10	0	0	0	0	0	0	0	0
	2	0	0	9	0	0	0	0	1	0	0
	3	0	0	0	10	0	0	0	0	0	0
	4	0	0	0	0	9	0	0	0	1	1
	5	0	0	0	0	0	6	0	0	0	0
	6	0	0	0	0	1	0	9	0	0	0
	7	0	0	0	0	0	0	0	9	0	1
	8	0	0	0	0	0	0	1	0	9	1
	9	0	0	1	0	0	4	0	0	0	7

Table 4.5: Confusion matrix for the Siamese-like CNN model over prepared dataset with MNIST data.

		Actual									
		0	1	2	3	4	5	6	7	8	9
Prediction	0	10	0	0	0	0	0	0	0	0	0
	1	0	8	0	0	0	0	0	0	0	0
	2	0	0	8	0	0	0	0	0	0	0
	3	0	0	0	10	0	5	0	1	0	0
	4	0	2	0	0	9	1	3	0	0	0
	5	0	0	0	0	0	4	3	0	0	0
	6	0	0	1	0	1	0	3	0	0	0
	7	0	0	0	0	0	0	0	9	0	0
	8	0	0	1	0	0	0	1	0	9	0
	9	0	0	0	0	0	0	0	0	1	10

Table 4.6: Confusion matrix for the Siamese-like CNN model with multiple control images over prepared dataset with MNIST data.

		Actual									
		0	1	2	3	4	5	6	7	8	9
Prediction	0	10	0	0	0	0	0	0	0	0	0
	1	0	8	0	0	0	0	0	0	0	0
	2	0	0	8	0	0	0	0	0	0	0
	3	0	0	0	10	0	0	0	0	0	0
	4	0	0	0	0	9	1	0	0	0	0
	5	0	0	0	0	0	7	0	0	0	0
	6	0	2	0	0	1	2	9	0	0	0
	7	0	0	0	0	0	0	0	10	0	0
	8	0	0	2	0	0	0	1	0	10	0
	9	0	0	0	0	0	0	0	0	0	10

Table 4.7: Precision, Recall, and F-1 score of Fully Connected Neural Network over MNIST dataset

	Precision	Recall	F-1 Score
0	0.90	0.69	0.78
1	0.70	1.00	0.82
2	0.30	0.75	0.43
3	0.70	1.00	0.82
4	0.30	0.30	0.30
5	0.50	0.33	0.40
6	0.60	0.43	0.50
7	0.80	1.00	0.89
8	0.60	0.43	0.50
9	0.50	0.62	0.56

Table 4.8: Precision, Recall, and F-1 score of Convolutional Neural Network over MNIST dataset

	Precision	Recall	F-1 Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	0.90	0.90	0.90
3	1.00	1.00	1.00
4	0.90	0.82	0.86
5	0.60	1.00	0.75
6	0.90	0.90	0.90
7	0.90	0.90	0.90
8	0.90	0.82	0.86
9	0.70	0.58	0.64

Table 4.9: Precision, Recall, and F-1 score of Siamese-like Convolutional Neural Network over MNIST dataset

	Precision	Recall	F-1 Score
0	1.00	1.00	1.00
1	0.80	1.00	0.89
2	0.80	1.00	0.89
3	1.00	0.62	0.77
4	0.90	0.60	0.72
5	0.40	0.57	0.47
6	0.30	0.60	0.40
7	0.90	1.00	0.95
8	0.90	0.82	0.86
9	1.00	0.91	0.95

Table 4.10: Precision, Recall, and F-1 score of Averaged Siamese-like Convolutional Neural Network over MNIST dataset

	Precision	Recall	F-1 Score
0	1.00	1.00	1.00
1	0.80	1.00	0.89
2	0.80	1.00	0.89
3	1.00	1.00	1.00
4	0.90	0.90	0.90
5	0.70	1.00	0.82
6	0.90	0.64	0.75
7	1.00	1.00	1.00
8	1.00	0.77	0.87
9	1.00	1.00	1.00

Table 4.11: Confusion matrix for the Fully Connected Neural Network model over prepared dataset with EMNIST data.

		Actual																								
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Prediction	A	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	B	0	4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	1	0	8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0
	E	0	1	0	0	9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	F	2	0	0	0	1	9	0	0	0	0	0	1	0	0	0	5	0	1	1	0	0	0	0	0	0	0
	G	0	1	0	0	0	0	9	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0
	H	1	0	0	0	0	0	0	8	0	0	0	0	6	0	0	0	0	0	0	0	1	0	1	0	0	0
	I	0	0	0	0	0	0	0	0	6	1	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	2
	J	1	1	0	0	0	0	0	0	0	9	0	0	0	1	0	0	0	0	2	0	4	2	3	0	0	0
	K	0	0	0	0	0	0	0	1	0	0	8	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
	L	1	0	0	0	0	0	0	0	4	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	M	0	0	0	0	0	0	0	1	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	1	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	1	0	0	0	0	0	0	0	0	0
	P	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	Q	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
	S	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0
	T	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
	U	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	2	1	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	6	1	0	1	0
	W	0	0	0	0	0	0	0	0	0	0	1	0	0	3	0	0	1	0	0	0	0	0	5	0	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	9	3	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	6	0
	Z	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	8

Table 4.12: Confusion matrix for the Convolutional Neural Network model over prepared dataset with EMNIST data.

		Actual																								
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Prediction	A	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	2	0	9	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
	E	0	0	0	0	10	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	I	1	0	0	0	0	0	0	0	9	2	0	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0
	J	0	0	0	0	0	0	0	0	0	8	0	0	0	1	0	0	0	0	0	0	2	3	0	0	0	0
	K	1	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	L	0	0	0	0	0	0	0	0	1	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	M	0	0	0	0	0	0	0	1	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	1	0	0	0	0	0	0	0	0	0	0
	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0
	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0
	Q	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	10	2	0	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
	U	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	1	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	10	0	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	10

Table 4.13: Confusion matrix for the Siamese-like CNN model over prepared dataset with MNIST data.

		Actual																								
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Prediction	A	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	I	0	0	0	0	0	0	0	0	10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	J	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	K	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	L	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	M	1	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	1	0	0	0	0	0
	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	1	0	0	0	0	0
	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	2	1	0	0	0	0
	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1	0	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8	0	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
	Y	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	10	0
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

Table 4.14: Confusion matrix for the Siamese-like CNN model with multiple control images over prepared dataset with EMNIST data.

		Actual																								
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Prediction	A	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	B	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	I	1	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	J	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	K	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	L	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	M	1	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	1	0	0	0	0	0
	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	1	0	1	0	0	0
	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	2	0	0	0	0	0
	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	10	0
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

The sixth observation is about time and resource utilization. The time taken for training and prediction by Fully Connected, Convolutional, Siamese-like and Averaged Siamese-like Convolutional Neural Network is shown in Table 4.19 and Table 4.20, and Figure 4.7 and Figure 4.8 for MNIST and EMNIST datasets respectively. With the technological advancements that we have been making, applications of Machine Learning are spreading across fields and platforms. Machine learning is being implemented in mobile devices, which have a limited amount of resources available. Due to the limited availability of resources, a mobile device such as a phone, tablet computer, smartwatch, etc., cannot afford to have some or most of its resources run an algorithm for a long time. It can be seen that the training time and prediction time of algorithms tested by us, increases in the order of Fully Connected Neural Network, Convolutional Neural Network and all versions of Siamese-like Convolutional Neural Network. Running a complex algorithm in a mobile device where accuracy is not of utmost importance, such as object/face detection in the photo gallery is an unnecessary and unwise move. Mentioning the time and resource utilization of an algorithm in the paper will allow the business or the developer from falling into such pitfalls if they know the requirements for the proper functioning of the algorithm before they start designing their product around the algorithm.

Table 4.15: Precision, Recall, and F-1 score of Fully Connected Neural Network over EMNIST dataset

	Precision	Recall	F-1 Score
A	0.40	0.67	0.50
B	0.40	0.80	0.53
C	1.00	1.00	1.00
D	0.80	0.67	0.73
E	0.90	0.82	0.86
F	0.90	0.45	0.60
G	0.90	0.69	0.78
H	0.80	0.47	0.59
I	0.60	0.46	0.52
J	0.90	0.39	0.55
K	0.80	0.73	0.76
L	0.70	0.58	0.64
M	0.40	0.80	0.53
N	0.60	0.86	0.71
O	0.80	0.89	0.84
P	0.10	0.50	0.17
Q	0.30	0.75	0.43
R	0.30	1.00	0.46
S	0.50	0.83	0.62
T	1.00	0.91	0.95
U	0.20	0.33	0.25
V	0.60	0.60	0.60
W	0.50	0.50	0.50
X	0.90	0.64	0.75
Y	0.60	0.75	0.67
Z	0.80	0.67	0.73

Table 4.16: Precision, Recall, and F-1 score of Convolutional Neural Network over EMNIST dataset

	Precision	Recall	F-1 Score
A	0.80	1.00	0.89
B	0.80	1.00	0.89
C	1.00	1.00	1.00
D	0.90	0.64	0.75
E	1.00	0.91	0.95
F	1.00	1.00	1.00
G	0.90	1.00	0.95
H	0.80	1.00	0.89
I	0.90	0.56	0.69
J	0.80	0.57	0.67
K	1.00	0.91	0.95
L	0.90	0.90	0.90
M	1.00	0.91	0.95
N	0.80	0.89	0.84
O	0.90	1.00	0.95
P	0.70	1.00	0.82
Q	1.00	0.71	0.83
R	0.80	1.00	0.89
S	0.90	1.00	0.95
T	1.00	1.00	1.00
U	0.40	0.80	0.53
V	0.60	0.86	0.71
W	1.00	0.83	0.91
X	1.00	1.00	1.00
Y	0.90	1.00	0.95
Z	1.00	0.91	0.95

Table 4.17: Precision, Recall, and F-1 score of Siamese-like Convolutional Neural Network over EMNIST dataset

	Precision	Recall	F-1 Score
A	0.80	0.80	0.80
B	1.00	1.00	1.00
C	1.00	1.00	1.00
D	1.00	0.91	0.95
E	1.00	1.00	1.00
F	1.00	1.00	1.00
G	1.00	1.00	1.00
H	1.00	1.00	1.00
I	1.00	0.91	0.95
J	0.90	0.75	0.82
K	1.00	1.00	1.00
L	1.00	1.00	1.00
M	1.00	0.91	0.95
N	1.00	1.00	1.00
O	0.90	1.00	0.95
P	0.90	0.90	0.90
Q	1.00	1.00	1.00
R	1.00	1.00	1.00
S	1.00	0.91	0.95
T	1.00	1.00	1.00
U	0.60	0.75	0.67
V	0.50	0.71	0.59
W	0.80	1.00	0.89
X	1.00	1.00	1.00
Y	1.00	0.83	0.91
Z	1.00	1.00	1.00

Table 4.18: Precision, Recall, and F-1 score of Averaged Siamese-like Convolutional Neural Network over EMNIST dataset

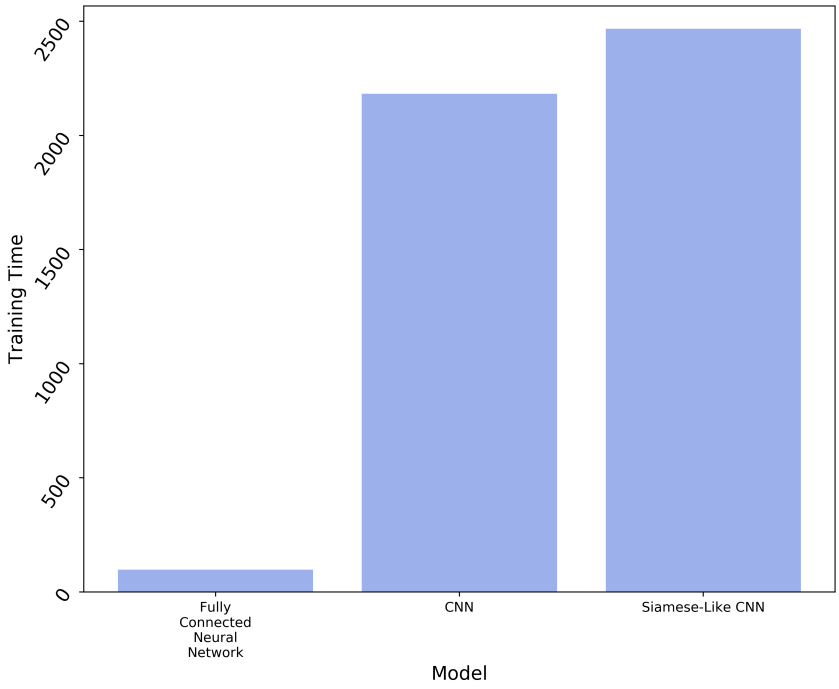
	Precision	Recall	F-1 Score
A	0.80	1.00	0.89
B	1.00	0.91	0.95
C	1.00	1.00	1.00
D	1.00	1.00	1.00
E	1.00	1.00	1.00
F	1.00	1.00	1.00
G	1.00	1.00	1.00
H	1.00	1.00	1.00
I	1.00	0.91	0.95
J	0.90	1.00	0.95
K	1.00	1.00	1.00
L	1.00	1.00	1.00
M	1.00	0.91	0.95
N	1.00	1.00	1.00
O	1.00	1.00	1.00
P	0.90	0.90	0.90
Q	1.00	1.00	1.00
R	1.00	1.00	1.00
S	1.00	0.91	0.95
T	1.00	0.77	0.87
U	0.50	1.00	0.67
V	0.90	0.90	0.90
W	0.80	0.89	0.84
X	1.00	1.00	1.00
Y	1.00	0.83	0.91
Z	1.00	1.00	1.00

Table 4.19: Average of time utilization of various algortihms over MNIST dataset

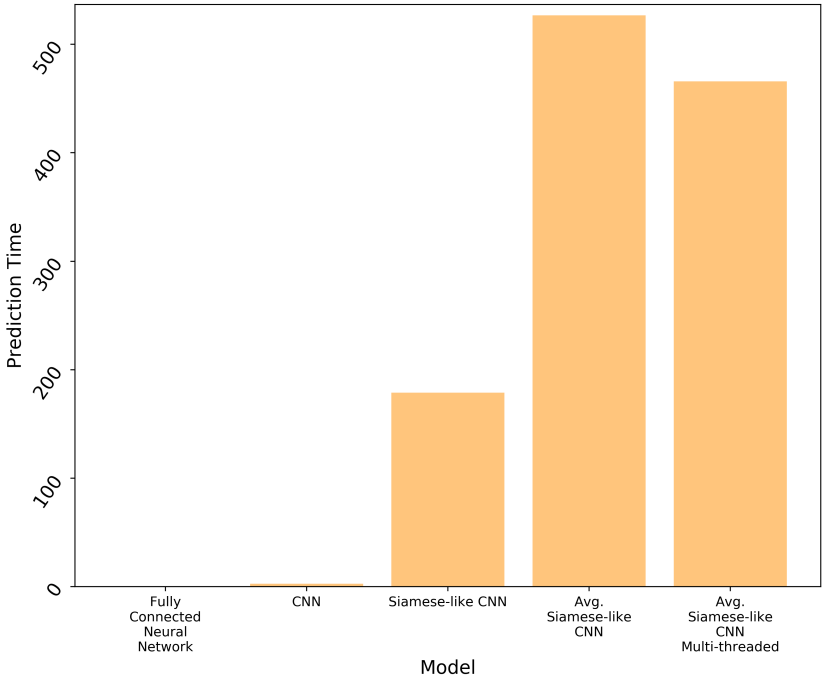
	Model observed				
	Fully Connected Neural Network	Conv. Neural Network	Siamese -like Conv. Neural Network	Averaged Siamese -like CNN	Multi-threaded Averaged Siamese -like CNN
Training Time	98	2182	2467	X	X
Prediction Time 200 samples	0.08	2.59	178.89	526.73	465.89

Table 4.20: Average of time utilization of various algortihms over EMNIST dataset

	Model observed				
	Fully Connected Neural Network	Conv. Neural Network	Siamese -like Conv. Neural Network	Averaged Siamese -like CNN	Multi-threaded Averaged Siamese -like CNN
Training Time	98	2182	2467	X	X
Prediction Time 200 samples	0.15	7.75	501.392	1376.11	1032.42

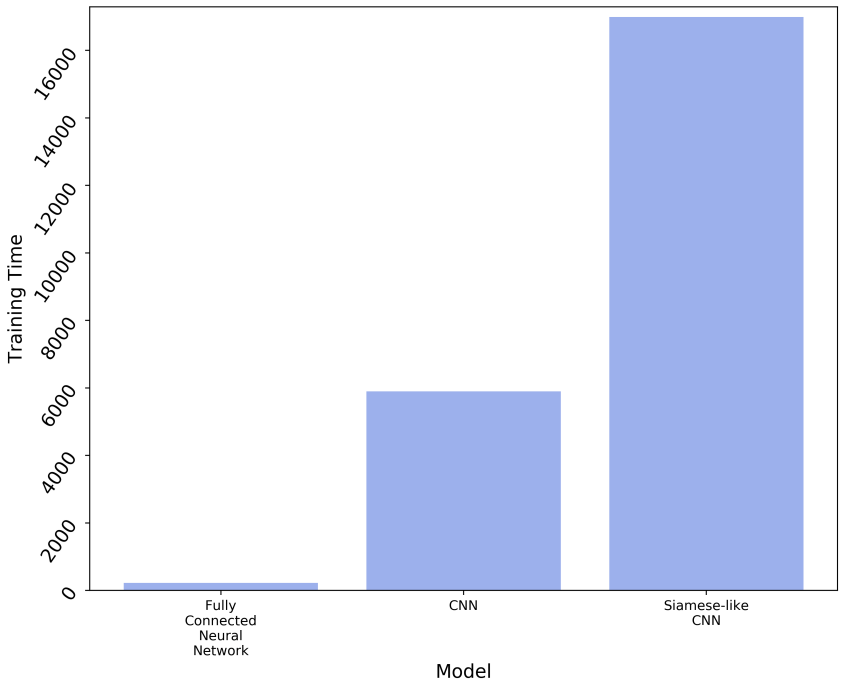


(a) Training time.

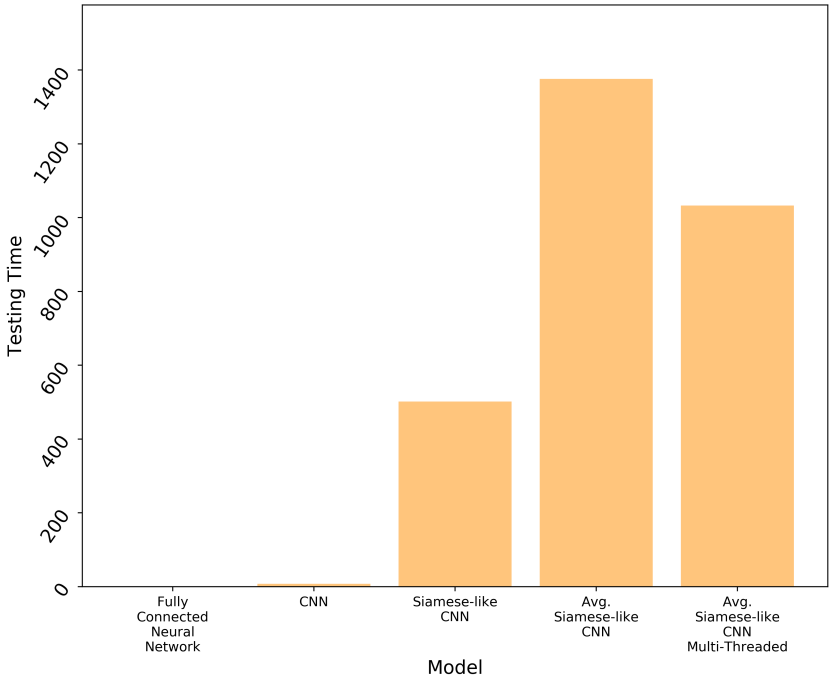


(b) Prediction time.

Figure 4.7: Comparison of the time metric of various models.(a). Time taken for model training of Fully Connected Neural Network, Convolutional Neural Network, and Siamese-like CNN models on MNIST dataset. (b) Time taken to predict 200 samples of each of the 10 classes of MNIST dataset by Fully Connected Neural Network, CNN, Siamese-like CNN, Siamese-like CNN with 3 control images and Siamese-like CNN with 3 control images and Multi-threading.



(a) Training time.



(b) Prediction time.

Figure 4.8: Comparison of the time metric of various models.(a). Time taken for model training of Fully Connected Neural Network, Convolutional Neural Network, and Siamese-like CNN models on EMNIST dataset. (b) Time taken to predict 200 samples of each of the 26 classes of EMNIST dataset by Fully Connected Neural Network, CNN, Siamese-like CNN, Siamese-like CNN with 3 control images and Siamese-like CNN with 3 control images and Multi-threading.

Chapter 5

Conclusion

We have observed that among the Fully Connected, Convolutional and Siamese-like Convolutional Neural Network when implemented over MNIST and EMNIST datasets, there are multiple other metrics that can be used to measure their performance. All algorithms give a lower performance on the prepared dataset in comparison to their performance in terms of accuracy on train and test portions of the datasets. Convolutional Neural Network provides better accuracy than Siamese-like Convolutional Neural Network, when tested on the train and test portions of the datasets, however, falls behind the latter when verified on the prepared dataset. This gives the idea that verification cannot be done entirely on the basis of test portion performance. Consistency in terms of precision or recall and the f-1 score is important depending on the type of application over which the algorithm might be deployed. Other metrics like time and resource consumption play a vital role with ML algorithms being deployed over smaller devices. Unavailability of human intervention can sometimes eliminate some algorithms from the list of suitable algorithms as they may not be up to par without human intervention in certain edge cases.

Further work in this field involves implementing a version of Siamese-like Neural Network where the base image can be saved as a feature vector, and the image that needs to be predicted can be directly compared with that by calculating the cosine distance. This methodology will make the comparison process a lot faster and will bring down the prediction time. More algorithms can be implemented apart from this and the co-relation of the results can be observed for possibly more metrics that can be used for algorithm selection. Due to limitation of time, one more aspect was not studied where some algorithms have two peak performance accuracies with two different sets of parameters, i.e. the total error plot does not look like a 'U' shaped graph but a 'W' shaped graph. This could also be explored further, as this provides more information about selecting parameters for training an algorithm. Another topic that can be explored is the mislabeled data in the datasets. Siamese-like Convolutional Neural Network is more resilient to mislabeled data as it works on

similarities between images rather working on probability distribution. Further study about this can provide more details about how resilient it is with respect to others.

Bibliography

[1] Syrine Ben Driss, Mahmoud Soua, Rostom Kachouri, and Mohamed Akil. A comparison study between mlp and convolutional neural network models for character recognition. In *SPIE Conference on Real-Time Image and Video Processing*, volume 10223, 2017.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, pages 737–744. Morgan Kaufmann Publishers Inc., 1993.

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017.

[4] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *CoRR*, abs/1802.03601, 2018.

[5] Gentle dive into math behind convolutional neural networks. https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9. Last Accessed: 2020-06-30.

[6] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Feature Grouping*. Springer, 1999.

[7] Fathma Siddique, Shadman Sakib, and Md. Abu Bakr Siddique. Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers. *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, 2019.

[8] Gregory R. Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, volume 37, 2015.

[9] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR 06*, pages 1735–1742. IEEE Computer Society, 2006.

[10] Luca Di Persio and Oleksandr Honchar. Artificial neural networks approach to the forecast of stock market price movements. *International journal of circuits, systems and signal processing*, 1:158–162, 2016.

[11] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[12] Alejandro Baldominos, Yago Sáez, and Pedro Isasi. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences*, 2019:3169, 2019.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[14] Keras.io. https://keras.io/. Last Accessed: 2020-06-30.

[15] Parampreet. Kaur and Jill. Stoltzfus. Type I, II, and III statistical errors: A brief overview. *International Journal of Academic Medicine*, 3(2):268–270, 2017.

[16] Antti Juvonen and Tuomo Sipola. Anomaly detection framework using rule extraction for efficient intrusion detection. *CoRR*, abs/1410.7709, 2014.