**PERFORMANCE ANALYSIS OF HANDWRITTEN CHARACTER RECOGNITION**

**USING CONVOLUTIONAL NEURAL NETWORK**

A Thesis submitted to the

Department of Computer Science and Engineering, Jahangirnagar University

in partial fulfillment of the requirements for the degree of

B.Sc. in Computer Science and Engineering

By

Exam Roll: 160020

Exam Roll: 160049



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAHANGIRNAGAR UNIVERSITY

JANUARY 2021

**Abstract**

Traditional systems of handwriting recognition have relied on handcrafted features and a large amount of prior knowledge. Training an Optical character recognition (OCR) system based on these prerequisites is a challenging task. Research in the handwriting recognition field is focused around deep learning techniques and has achieved breakthrough performance in the last few years. Still, the rapid growth in the amount of handwritten data and the availability of massive processing power demands improvement in recognition accuracy and deserves further investigation. Convolutional neural networks (CNNs) are very effective in perceiving the structure of handwritten characters/words in ways that help in automatic extraction of distinct features and make CNN the most suitable approach for solving handwriting recognition problems. Our aim in the proposed work is to explore and evaluate various optimization algorithms in improving the performance of handwritten digit recognition. Thus, a CNN architecture is proposed in order to achieve accuracy even better than that of ensemble architectures. We carried out extensive experiments and achieved a recognition accuracy of 98.4% for a Kaggle dataset.

**Acknowledgement**

In performing our assignment, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much Pleasure. We would like to show our gratitude to our honorable Supervisor who guided and coached us throughout our project.

**Contents**

**Abstract…………………………………………………………………………....2**

**Acknowledgement…………………………………………………………….…..3**

**List of Figures………………………………………………………………….….7**

**1.0 Introduction**.......................................................................................................8

1.1 Background and Motivation.....................................................................9

1.2 Objective...................................................................................................9

1.3 Research Problem ...................................................................................9

1.4 Contribution ...........................................................................................10

**2.0 Literature Review**

2.1 Past work .............................................................................................10

**3.0 Proposed Framework**.....................................................................................11

3.1 Convolutional Neural Network Architecture..................................................................................................11

3.1.1 Input layer................................................................................12

3.1.2 Hidden Layer...........................................................................12

3.1.3 Convolutional Layer................................................................12

3.1.4 Pooling Layer..........................................................................13

3.1.5 Activation Laye.......................................................................13

3.1.6 Classification Layer ...............................................................14

3.1.7 Optimization Algorithm ..........................................................14

**4.0 Project Methodology**......................................................................................14

4.1 Data......................................................................................................14

4.2 Structure of the CNN...........................................................................14

4.3 Optimization Functions.......................................................................16

4.3.1 Adam .......................................................................................16

4.3.2 SGD..........................................................................................16

4.3.3 RMSProp..................................................................................16

4.3.4 Adagrad ...................................................................................16

**5.0 Result**..............................................................................................................17

5.1 Experiments...........................................................................................17

5.1 Testing accuracy and loss using Adam.......................................17

5.1 Testing accuracy and loss using SGD.........................................17

5.1 Testing accuracy and loss using RMSProp.................................18

5.1Testing accuracy and loss using Adagrad…................................19

5.2 Result.....................................................................................................17

**6.0 Conclusion**......................................................................................................22

6.1Summary................................................................................................22

6.2 Future Work..........................................................................................22

**References**………………………………………………………………....…....23

# 

# 

# 

# **List of Figures**

3.1: Typical convolutional neural network

architecture image.………………………………………...………...12

4.1: Flowchart of Proposed Framework.....................................................15

5.1.1:Train vs Test accuracy(Adam)..........………………………………18

5.1.2:Train vs Test loss(Adam)..........…………………………....……....18

5.2.1:Train vs Test accuracy(SGD)..........………………………………..18

5.2.2:Train vs Test loss(SGD)..........…………………………....………..18

5.1.1:Train vs Test accuracy(RMSProp)..........……………..…………....18

5.1.2:Train vs Test loss(RMSProp)..........……………………......……....18

5.1.1:Train vs Test accuracy(Adagrad)..........……………………………19

5.1.2:Train vs Test loss(Adagrad)..........…………………………....……19

5.5.1:Training accuracy using different optimizer…………….…………20

5.5.2:Testing accuracy using different optimizer…..……….....................20

5.6: Performance of the model based on

handwritten character drawn by us…..………...................................21

**Chapter 1**

**Introduction**

In the current age of digitization, handwriting recognition plays an important role in information processing. A lot of information is available on paper, and processing of digital files is cheaper than processing traditional paper files. The aim of a handwriting recognition system is to convert handwritten characters into machine readable formats. The main applications are vehicle license-plate recognition, postal letter-sorting services, Cheque truncation system (CTS) scanning and historical document preservation in archaeology departments, old documents automation in libraries and banks, etc. All these areas deal with large databases and hence demand high recognition accuracy, lesser computational complexity and consistent performance of the recognition system.The deep learning field is ever evolving, and some of its variants are autoencoders, CNNs, recurrent neural networks (RNNs), recursive neural networks, deep belief networks and deep Boltzmann machines. Here, we introduce a convolutional neural network, which is a specific type of deep neural network having wide applications in image classification, object recognition, recommendation systems, signal processing, natural language processing, computer vision, and face recognition. The ability to automatically detect the important features of an object (here an object can be an image, a handwritten character, a face, etc.) without any human supervision or intervention makes them (CNNs) more efficient than their predecessors (Multi layer perceptron (MLP), etc.). The high capability of hierarchical feature learning results in a highly efficient CNN.

A convolutional neural network (CNN) is basically a variation of a multi-layer perceptron (MLP) network and was used for the first time in 1980 [7]. The computing in CNN is inspired by the human brain. Humans perceive or identify objects visually. We (humans) train our children to recognize objects by showing him/her hundreds of pictures of that object. This helps a child identify or make a prediction about objects he/she has never seen before. A CNN works in the same fashion and is popular for analyzing visual imagery. Some of the well-known CNN architectures are GoogLeNet (22 layers), AlexNet (8 layers), VGG (16–19 Ali), and ResNet (152 layers). A CNN integrates the feature extraction and classification steps and requires minimal preprocessing and feature extraction efforts. A CNN can extract affluent and interrelated features automatically from images. Moreover, a CNN can provide considerable recognition accuracy even if there is only a little training data available.

**1.1 Background and Motivation**

Over the last few decades, research on handwriting recognition has made impressive progress. The research and development on handwritten word recognition are to a large degree motivated by many application areas, such as automated postal address and code reading, data acquisition in banks, text-voice conversion, security, etc. As the prices of scanners, computers and handwriting-input devices are falling steadily, we have seen an increased demand for handwriting recognition systems and software packages. Some commercial handwriting recognition systems are now available in the market. Current commercial systems have an impressive performance in recognizing machine printed characters and neatly written texts. For instance, High-Tech Solutions in Israel has developed several products for container ID recognition, car license plate recognition and package label recognition

**1.2 Objective**

The main aim of this project is to design an expert system that can effectively recognize a particular character of type format using the Convolutional Neural Network approach. Neural computing is a comparatively new field, and design components are therefore less well specified than those of other architectures.

This application is useful for recognizing all characters(English) given as in the input image. Once an input image of a character is given to the proposed system, then it will recognize the input character which is given in the image. Recognition and classification of characters are done by Neural Network. The main aim of this project is to effectively recognize a particular character of type format using the Convolutional Neural Network approach.

**1.3 Research Problem**

This application is useful for recognizing all characters(English) given as input images. Once an input image of a character is given to the proposed system, then it will recognize the input character which is given in the image. Recognition and classification of characters are done by Convolutional Neural Network. The main aim of this project is to effectively recognize a particular character of type format using the Convolutional Neural Network approach.

**1.4 Contribution**

After researching the literature review we started this project to contribute something new to this research area. As a group we are able to recognize handwritten characters using CNN with an accuracy higher than 98% after using different optimization algorithms and finding the best one to improve our recognition model. We tried to make this model a better one to recognize more accurately than any other model.

**Chapter 2**

**Literature Review**

**2.1 Past work review**

Handwriting recognition has already achieved impressive results using shallow networks [1–10]. Many papers have been published with research detailing new techniques for the classification of handwritten numerals, characters and words. The deep belief networks (DBN) with three layers along with a greedy algorithm were investigated for the MNIST dataset and reported an accuracy of 98.75% [11]. Pham et al. applied a regularization method of dropout to improve the performance of recurrent neural networks (RNNs) in recognizing unconstrained handwriting [12]. The author reported improvement in RNN performance with significant reduction in the character error rate (CER) and word error rate (WER). The convolutional neural network brings a revolution in the handwriting recognition field and delivers state-of-the-art performance in this domain [13–18]. In 2003, Simard et al. introduced a general convolutional neural network architecture for visual document analysis and weeded out the complex method of neural network training [19]. Wang et al. proposed a novel approach for Sensors 2020, 20, 3344 4 of 18 end-to-end text recognition using multi-layer CNNs and achieved excellent performance on benchmark databases, namely, ICDAR 2003 and Street View Text [20]. Recently, Shi et al. integrated the advantages of both the deep CNN (DCNN) and recurrent neural network (RNN) and named it conventional recurrent neural network (CRNN). They applied CRNN for scene text recognition and found it to be superior to traditional methods of recognition [21]. Badri narayanan et al. proposed a deep convolution network architecture for semantic segmentation. The segmentation architecture is known as SegNet and consists of an encoder network, a decoder network and a pixel-wise classification layer. The proposed method used max-pooling indices of a feature map while decoding and observed good performance. The method is also analyzed and compared with existing techniques for road scene and indoor understanding [22–24]. CNN has shown remarkable abilities in offline handwritten character recognition of Arabic language [25]; handwritten Tamil character recognition [26]; Telugu character recognition [27], handwritten Urdu text recognition [28,29], handwritten character recognition in Indic scripts [30] and Chinese handwritten text recognition [31-33].The performance of CNNs depends mainly on the choice of hyper-parameters [34], which are usually decided on a trial-and-error basis. Some of the hyper-parameters are, namely, activation function, number of epochs, kernel size, learning rate, hidden units, hidden layers, etc. These parameters are very important as they control the way an algorithm learns from data [35]. Hyper-parameters differ from model parameters and must be decided before the training begins.

**Chapter 3**

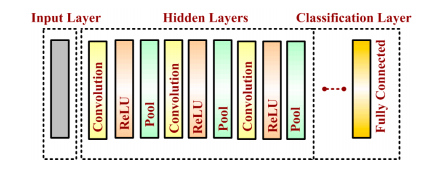
**Proposed Framework**

**3.1 Convolutional Neural Network Architecture**

A basic convolutional neural network comprises three components, namely, the convolutional layer, the pooling layer and the output layer. The pooling layer is optional sometimes. The typical convolutional neural network architecture with three convolutional layers is well adapted for the classification of handwritten images as shown in Figure 1. It consists of the input layer, multiple hidden layers (repetitions of convolutional, normalization, pooling) and a fully connected and an output layer. Neurons in one layer connect with some of the neurons present in the next layer, making the scaling easier for the higher resolution images. The operation of pooling or sub-sampling can be used to reduce the dimensions of the input. In a CNN model, the input image is considered as a collection of small sub-regions called the “receptive fields”. A mathematical operation of the convolution is applied on the input layer, which emulates the response to the next layer. The response is basically a visual stimulus. The detailed description is as follows:

**3.1.1 Input layer**

The input data is loaded and stored in the input layer. This layer describes the height, width and number of channels (RGB information) of the input.

Fig 3.1:Typical convolutional neural network architecture image.

**3.1.2 Hidden Layer**

The hidden layers are the backbone of CNN architecture. They perform a feature extraction process where a series of convolution, pooling and activation functions are used. The distinguishable features of handwritten digits are detected at this stage.

**3.1.3 Convolutional Layer**

The convolutional layer is the first layer placed above the input image. It is used for extracting the features of an image. The n × n input neurons of the input layer are convoluted with an m × m filter and in return deliver (n − m + 1) × (n − m + 1) as output. It introduces non-linearity through a neural activation function. The main contributors of the convolutional layer are receptive field, stride, dilation and padding, as described in the following paragraph.

CNN computation is inspired by the visual cortex in animals [36]. The visual cortex is a part of the brain that processes the information forwarded from the retina. It processes visual information and is subtle to small sub-regions of the input. Similarly, a receptive field is calculated in a CNN, which is a small region of an input image that can affect a specific region of the network. It is also one of the important design parameters of the CNN architecture and helps in setting other CNN parameters [37]. It has the same size as the kernel and works in a similar fashion as the vision of the human eye works for producing sharp central vision. The receptive field is influenced by striding, pooling, kernel size and depth of the CNN [38]. Receptive field (r), effective receptive field (ERF) and projective field (PF) are terminology used in calculating effective sub-regions in a network. The area of the original image influencing the activation of a neuron is described using the ERF, whereas the PF is a count of neurons to which neurons project their outputs, as described in Figure 2. The visualization of the 5 × 5-size filter and its activation map are described in Figure 3. Stride is another parameter used in CNN architecture. It is defined as the step size by which the filter moves every time. A stride value of 1 indicates the filter sliding movement pixel by pixel. A larger stride size shows less overlapping between the cells. The working of the kernel and stride in the convolution layer is presented in Figure 4.

**3.1.4 Pooling Layer**

A pooling layer is added between two convolutional layers to reduce the input dimensionality and hence to reduce the computational complexity. Pooling allows the selected values to be passed to the next layer while leaving the unnecessary values behind. The pooling layer also helps in feature selection and in controlling overfitting. The pooling operation is done independently. It works by extracting only one output value from the tiled non-overlapping sub-regions of the input images. The common types of pooling operations are max-pooling and avg-pooling (where max and avg represent maxima and average, respectively). The max-pooling operation is generally favorable in modern applications, because it takes the maximum values from each sub-region, keeping maximum information.

**3.1.5 Activation Layer**

Just like regular neural network architecture, CNN architecture also contains the activation function to introduce the non-linearity in the system. The sigmoid function, rectified linear unit (ReLu) and Softmax are some famous choices among various activation functions exploited extensively in deep learning models. It has been observed that the sigmoid activation function might weaken the CNN model because of the loss of information present in the input data. The activation function used in the present work is the non-linear rectified linear unit (ReLu) function, which has output 0 for input less than 0 and raw output otherwise. Some advantages of the ReLu activation function are its similarity with the human nerve system, simplicity in use and ability to perform faster training for larger networks

**3.1.6 Classification Layer**

The classification layer is the last layer in CNN architecture. It is a fully connected feed forward network, mainly adopted as a classifier. The neurons in the fully connected layers are connected to all the neurons of the previous layer. This layer calculates predicted classes by identifying the input image, which is done by combining all the features learned by previous layers. The number of output classes depends on the number of classes present in the target dataset. In the present work, the classification layer uses the ‘softmax’ activation function for classifying the generated features of the input image received from the previous layer into various classes based on the training data.

**3.1.7 Optimization Algorithm**

Optimization algorithms are used to optimize neural networks and to generate better performance and faster results. The algorithm helps in minimizing or maximizing a cost function by updating the weight/bias values, which are known as learning parameters of a network, and the algorithm updating these values is termed as the adaptive learning algorithm. These learning parameters directly influence the learning process of a network and have an important role in producing an efficient network model. The aim of all the optimization algorithms is to find the optimum values of these learning parameters.

**Chapter 4**

**Project Methodology**

**4.1 Data**

The data we collected is a dataset called “A-Z Handwritten Data.csv” from kaggle for our project. The dataset for this project contains 372450 images of alphabets all present in the form of a CSV file. After reading the data from the dataset, the data was preprocessed and split into training and testing dataset and was also reshape into 28x28 pixels. We converted the floating point values of the label into integer and created a dictionary to map the integer values with the characters. After that the thresholded gray scale image was depicted from the dataset, and an opencv thresholding library called thresh binary was used for that.

**4.2 Structure of the CNN**

To build the Convolutional Neural Network, we used the “Sequential” model from Keras. We had 3 convolutional layers followed by 3 pooling layers. We used 32,64 and 128 filters for the 1st,2nd and 3rd convolutional layer respectively and the kernel size was 3x3. For the activation function “ReLU” was used for each convolutional layer.

ReLU(x) = ⎨ x if x > 0

0 if x ≤ 0

The choice of the activation function significantly affects the speed of convergence. Compared with traditional sigmoid function selection, the ReLU function has been proven to speed up the training process multiple times. It is a piecewise linear function, which turns all negative values into 0, while the positive value does not change. This operation is unilaterally suppressed. Because of this unilateral inhibition, the neurons in the neural network also have sparse activating, and the negative neurons are set to 0 during the training process, which reduces the amount of calculation and is equivalent to reducing the network volume. That’s

why ReLU was chosen as our activation function.

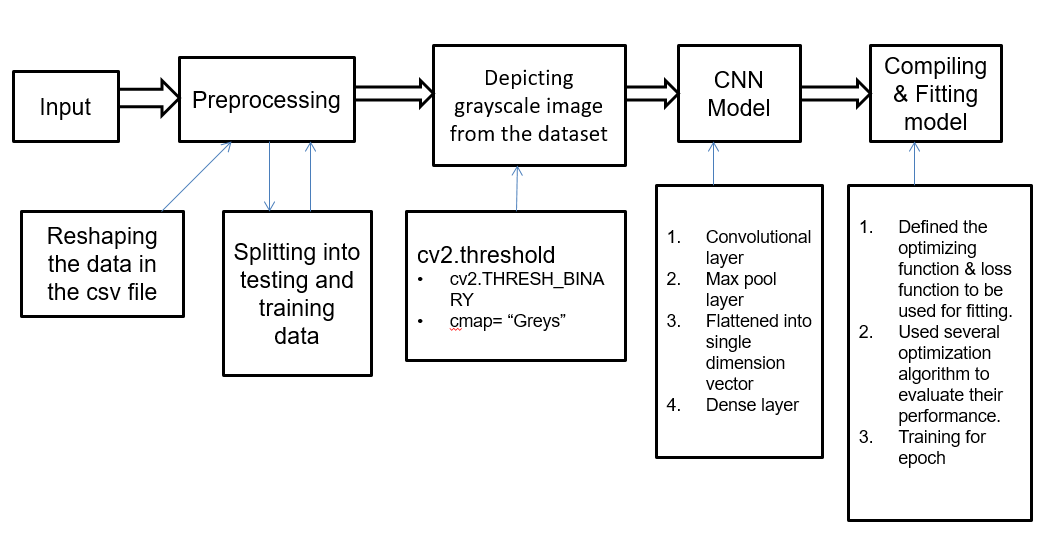
After each convolution layer we used a maxpool layer to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The output of the maxpool layers and convolution layers are then flattened into a vector of single dimension and given as an input to the fully connected layer or Dense layer.

Fig 4.1:Flowchart of Proposed Framework

To complete the model it was then compiled with an optimizer. For this part we tested out several optimizing algorithms that were found to be used for this purpose and fitted the data in those compiled models to evaluate the performance of each model with different optimization algorithms.

For loss function, “categorical\_crossentropy” from keras was used. Cross-entropy loss function characterizes the distance between the actual output (probability) and the expected output (probability). Thus, the smaller the value of the cross entropy is, the closer the two probability distributions are. Learning rate of 0.001 was used which is an important hyperparameter that controls the speed at which we adjust neural network weights based on the loss gradients.To fit the data in our model since we are using a large dataset, we used 10 epochs.

**4.3 Optimization Functions**

We have tried out several optimizer to compile our model to find the best one for our project. Following are the optimizers that were used in our project:

**4.3.1 Adam**

Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. Adam is relatively easy to configure where the default configuration parameters do well on most problems.

**4.3.2 SGD**

Stochastic gradient descent (SGD) is an [iterative method](https://en.wikipedia.org/wiki/Iterative_method) for [optimizing](https://en.wikipedia.org/wiki/Mathematical_optimization) an [objective function](https://en.wikipedia.org/wiki/Objective_function) with suitable [smoothness](https://en.wikipedia.org/wiki/Smoothness) properties . It can be regarded as a [stochastic approximation](https://en.wikipedia.org/wiki/Stochastic_approximation) of [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent) optimization, since it replaces the actual gradient by an estimate thereof . Especially in [high-dimensional](https://en.wikipedia.org/wiki/High-dimensional) optimization problems this reduces the [computational burden](https://en.wikipedia.org/wiki/Computational_complexity), achieving faster iterations in trade for a lower convergence rate.

**4.3.3 RMSprop**

RmsProp optimizer. RmsProp is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients. We always keep a moving average over the root mean squared (hence Rms) gradients, by which we divide the current gradient.

**4.3.4 Adagrad**

AdaGrad is a stochastic optimization method that adapts the learning rate to the parameters. It performs smaller updates for parameters associated with frequently occurring features, and larger updates for parameters associated with infrequently occurring features.

**Chapter 5**

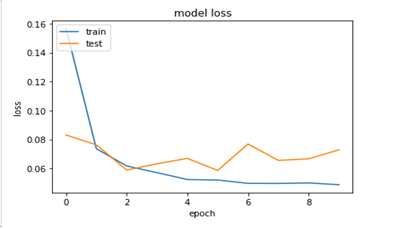
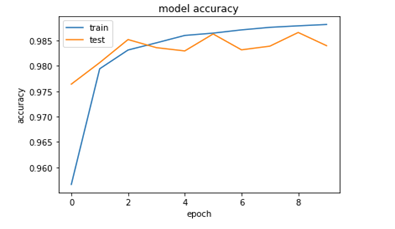
**Result**

**5.1 Experiments**

Here are the test results and analysis of the different optimization functions we used to understand the best one for our project.

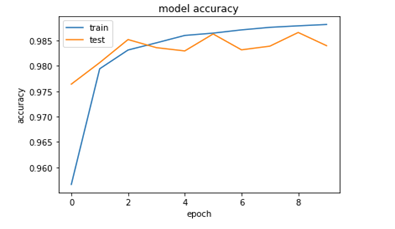
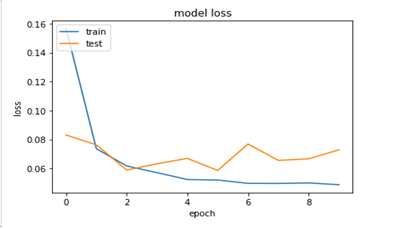
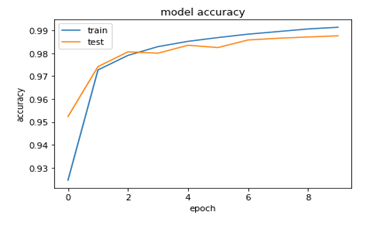
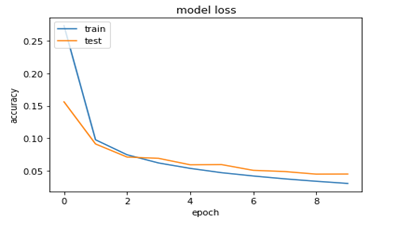
**5.1.1 Testing accuracy and loss using Adam**

First we used the adam optimizer, the left graph shows the train vs test accuracy, the right graph shows the train vs test loss. As we can see from the accuracy graph, even from the first epoch, the training accuracy is over 95% which at the end of running 10 epochs increases over 98%. The testing accuracy was also pretty good staying over 97% pretty consistently over all 10 epoch. The model loss graph also shows pretty consistently low result for both training and testing loss.

Fig 5.1.1:Train vs Test accuracy(Adam) Fig 5.1.2:Train vs Test loss(Adam)

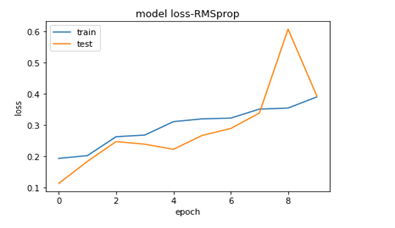
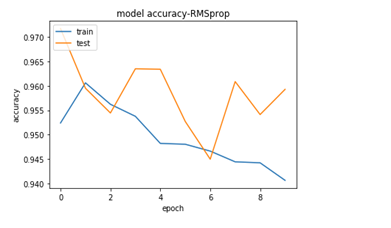
**5.1.2 Testing accuracy and loss using SGD**

Next we used the Stochastic Gradient Descent or SGD optimizer, as we can see from the accuracy graph, even though training accuracy started from around 92%, it gradually increased as we went over more epochs to go over 99% accuracy at the end. The testing result was also consistently similar to that of training result.

Fig 5.2.1:Train vs Test accuracy(SGD) Fig 5.2.2:Train vs Test loss(SGD)

**5.1.3 Testing accuracy and loss using RMSprop**

Then we used RMSprop optimizer, as we can see from the graph, compared to the 2 optimizer we used before where as we ran the model over and over again the accuracy for both training and testing increased, however the rmsprop using model showed different nature, it started going up for first 2 epoch but then gradually started decreasing in terms of accuracy, as we ran only 10 epoch we cant be sure just how far it will go down or if it will start increasing again at some point, but as of our observation it showed poor performance, the testing result was also similar to the training result and very inconsistent over all the run.

Fig5.3.1:Train vs Test accuracy(RMSProp) Fig5.3.2:Train vs Testloss(RMSProp)

**5.1.4 Testing accuracy and loss using Adagrad**

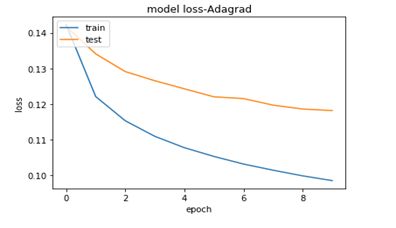
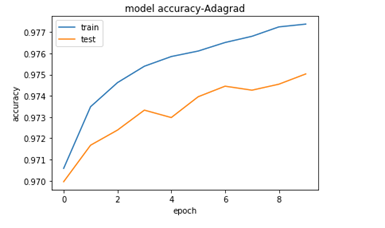
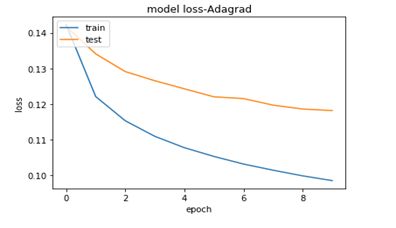
Lastly we used adagrad, which also showed good consistent result for both the training and testing accuracy.

Fig5.4.1:Train vs Test accuracy(Adagrad) Fig5.4.2:Train vs Test loss(Adagrad)

**5.2 Result**

Here we compared all the model performance using different optimization algorithm to reach a decision to choose the best one for our project, as we can see, SGD and Adam optimizer based models showed the best results and SGD even going over the accuracy of Adam optimizer based model slightly in the end, but the adam optimizer showed more consistency over all the run in both training and testing accuracy and achieved a pretty high accuracy over 98% consistently. So we choose adam as our optimizer for our model which gave us an accuracy of 98.4% and built the project to recognize handwritten character given to it.

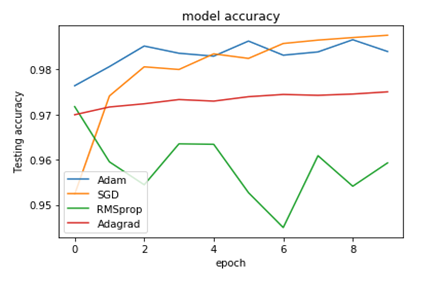
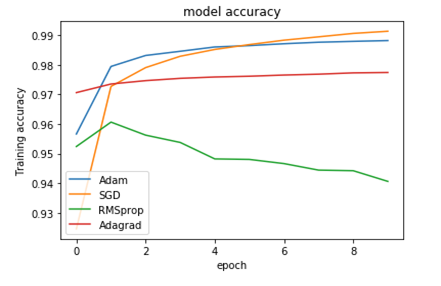
Fig 5.5.1:Training accuracy using different optimizer

Fig 5.5.2:Testing accuracy using different optimizer

Now here are some characters drawn by us to test the model with real data and see if its able to predict the words correctly. Even though it was able to predict most of the characters correctly sometime it blunders and makes the wrong prediction too, like it did in the last image. In near future we are looking to improve this part of our project more to make it predict real life data more accurately.

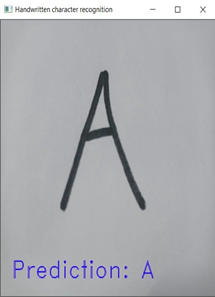


Fig 5.6:Performance of the model based on handwritten character drawn by us

**Chapter 6**

**Conclusion**

**6.1 Summary**

We have successfully developed Character recognition with Python, Tensorflow, and Machine Learning libraries.

Characters have been recognized with high accuracy using the same model with different optimizers to further find out the best approach . This can be also further extended to identifying the handwritten characters of other languages too. Also we are thinking using different CNN architecture to make our work even better in near future.

**6.2 Future Work**

1.For the future work, we are thinking of changing the CNN architecture for our model and try out the alexnet and googlenet CNN architecture to see the respective outcome and compare it with our current output to further improve this project.

2.Research in OCR domain is usually done on some of the most widely spoken languages. This is partially due to non-availability of datasets on other languages. One of the future research direction is to conduct research on languages other than widely spoken languages i.e. regional languages and endangered languages. This can help preserve cultural heritage of vulnerable communities and will also create positive impact on strengthening global synergy.

3. Another research problem that needs attention of research community is to built systems that can recognize on screen characters and text in different conditions in daily life scenarios e.g. text in captions or news tickers, text on sign boards, text on billboards etc.

4. To build robust system for “text in the wild”, researchers needs to come up with challenging datasets that is comprehensive enough to incorporate all possible variations in characters. Aim of this challenge is invite research studies that proposes robust system for multilingual text recognition in daily life or “in the wild” scenario.

**References**

1. Choudhary, A.; Ahlawat, S.; Rishi, R. A neural approach to cursive handwritten character recognition using features extracted from binarization technique. Complex Syst. Model. Control Intel. Soft Comput. 2015, 319, 745–771.
2. Choudhary, A.; Rishi, R.; Ahlawat, S. Handwritten numeral recognition using modified BP ANN structure. In Proceedings of the Communication in Computer and Information Sciences (CCIS-133), Advanced Computing, CCSIT 2011, Royal Orchid Central, Bangalore, India, 2–4 January 2011; Springer: Berling/Heildelberg, Germany, 2011; pp. 56–65.
3. Cai, Z.W.; Li-Hong, H. Finite-time synchronization by switching state-feedback control for discontinuous Cohen–Grossberg neural networks with mixed delays. Int. J. Mach. Learn. Cybern. 2018, 9, 1683–1695. [CrossRef]
4. Zeng, D.; Dai, Y.; Li, F.; Sherratt, R.S.; Wang, J. Adversarial learning for distant supervised relation extraction. Comput. Mater. Contin. 2018, 55, 121–136.
5. Long, M.; Yan, Z. Detecting iris liveness with batch normalized convolutional neural network. Comput. Mater. Contin. 2019, 58, 493–504. [CrossRef]
6. Chuangxia, H.; Liu, B. New studies on dynamic analysis of inertial neural networks involving non-reduced order method. Neurocomputing 2019, 325, 283–287.
7. Xiang, L.; Li, Y.; Hao, W.; Yang, P.; Shen, X. Reversible natural language watermarking using synonym substitution and arithmetic coding. Comput. Mater. Contin. 2018, 55, 541–559.
8. Huang, Y.S.; Wang, Z.Y. Decentralized adaptive fuzzy control for a class of large-scale MIMO nonlinear systems with strong interconnection and its application to automated highway systems. Inf. Sci. 2014, 274, 210–224. [CrossRef]
9. Choudhary, A.; Rishi, R. Improving the character recognition efficiency of feed forward bp neural network. Int. J. Comput. Sci. Inf. Technol. 2011, 3, 85–96. [CrossRef]
10. Ahlawat, S.; Rishi, R. A genetic algorithm based feature selection for handwritten digit recognition. Recent Pat. Comput. Sci. 2019, 12, 304–316. [CrossRef]
11. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. Neural Comput. 2006, 18, 1527–1554. [CrossRef]
12. Pham, V.; Bluche, T.; Kermorvant, C.; Louradour, J. Dropout improves recurrent neural networks for handwriting recognition. In Proceedings of the 14th Int. Conf. on Frontiers in Handwriting Recognition, Heraklion, Greece, 1–4 September 2014.
13. Tabik, S.; Alvear-Sandoval, R.F.; Ruiz, M.M.; Sancho-Gómez, J.L.; Figueiras-Vidal, A.R.; Herrera, F. MNIST-NET10: A heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. Ensembles Overv. Proposal Inf. Fusion 2020, 62, 73–80. [CrossRef]
14. Lang, G.; Qingguo, L.; Mingjie, C.; Tian, Y.; Qimei, X. Incremental approaches to knowledge reduction based on characteristic matrices. Int. J. Mach. Learn. Cybern. 2017, 8, 203–222. [CrossRef]
15. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 2481–2495. [CrossRef]
16. He, S.; Zeng, W.; Xie, K.; Yang, H.; Lai, M.; Su, X. PPNC: Privacy preserving scheme for random linear network coding in smart grid. KSII Trans. Internet Inf. Syst. 2017, 11, 1510–1532. Sensors 2020, 20, 3344 16 of 18
17. Sueiras, J.; Ruiz, V.; Sanchez, A.; Velez, J.F. Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing. 2018, 289, 119–128. [CrossRef]
18. Liang, T.; Xu, X.; Xiao, P. A new image classification method based on modified condensed nearest neighbor and convolutional neural networks. Pattern Recognit. Lett. 2017, 94, 105–111. [CrossRef]
19. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practice for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, UK, 3–6 August 2003.
20. Wang, T.; Wu, D.J.; Coates, A.; Ng, A.Y. End-to-end text recognition with convolutional neural networks. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012
21. Shi, B.; Bai, X.; Yao, C. An End-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 2298–2304. [CrossRef]
22. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
23. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. 2018, 40, 834–848. [CrossRef]
24. Noh, H.; Hong, S.; Han, B. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, Araucano Park, Las Condes, Chille, 11–18 December 2015.
25. Boufenar, C.; Kerboua, A.; Batouche, M. Investigation on deep learning for off-line handwritten Arabic character recognition. Cogn. Syst. Res. 2018, 50, 180–195. [CrossRef]
26. Kavitha, B.; Srimathi, C. Benchmarking on offline Handwritten Tamil Character Recognition using convolutional neural networks. J. King Saud Univ. Comput. Inf. Sci. 2019. [CrossRef]
27. Dewan, S.; Chakravarthy, S. A system for offline character recognition using auto-encoder networks. In Proceedings of the International Conference on Neural Information Processing, Doha, Qatar, 12–15 November 2012.
28. Ahmed, S.; Naz, S.; Swati, S.; Razzak, M.I. Handwritten Urdu character recognition using one-dimensional BLSTM classifier. Neural Comput. Appl. 2019, 31, 1143–1151. [CrossRef]
29. Husnain, M.; Saad Missen, M.; Mumtaz, S.; Jhanidr, M.Z.; Coustaty, M.; Luqman, M.M.; Ogier, J.-M.; Choi, G.S. Recognition of urdu handwritten characters using convolutional neural network. Appl. Sci. 2019, 9, 2758. [CrossRef]
30. Sarkhel, R.; Das, N.; Das, A.; Kundu, M.; Nasipuri, M. A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts. Pattern Recognit. 2017, 71, 78–93. [CrossRef]
31. Xie, Z.; Sun, Z.; Jin, L.; Feng, Z.; Zhang, S. Fully convolutional recurrent network for handwritten chinese text recognition. In Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016), Cancun, Mexico, 4–8 December 2016.
32. Liu, C.; Yin, F.; Wang, D.; Wang, Q.-F. Online and offline handwritten Chinese character recognition: Benchmarking on new databases. Pattern Recognit. 2013, 46, 155–162. [CrossRef]
33. Wu, Y.-C.; Yin, F.; Liu, C.-L. Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models. Pattern Recognit. 2017, 65, 251–264. [CrossRef]
34. Cui, H.; Bai, J. A new hyperparameters optimization method for convolutional neural networks. PatternRecognit. Lett. 2019, 125, 828–834. [CrossRef]Sensors 2020, 20, 3344 17 of 18
35. Tso, W.W.; Burnak, B.; Pistikopoulos, E.N. HY-POP: Hyperparameter optimization of machine learning models through parametric programming. Comput. Chem. Eng. 2020, 139, 106902. [CrossRef]
36. Eickenberg, M.; Gramfort, A.; Varoquaux, G.; Thirion, B. Seeing it all: Convolutional network layers map the function of the human visual system. NeuroImage 2017, 152, 184–194. [CrossRef] [PubMed]
37. Le, H.; Borji, A. What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks? arXiv 2018, arXiv:1705.07049.
38. Luo, W.; Li, Y.; Urtasun, R.; Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. arXiv 2017, arXiv:1701.04128.