Handwritten Character Recognition using Neural Networks

Anushka Swarup
Department of ECE
University of Florida
aswarup@ufl.edu

Avinash Ayalasomayajula
Department of ECE
University of Florida
ayalasomayajul.a@ufl.edu

Kshitij Raj
Department of ECE
University of Florida
kshitijraj@ufl.edu

Maneesh Merugu
Department of ECE
University of Florida
maneesh.merugu@ufl.edu

***Abstract*—OCR techniques have been predominant since the advent of computer vision. They have seen notable improvements particularly in the last two decades since the increase in popularity of neural networks, which were widely disregarded in the late 90's. In this project we solve the problem of classifying handwritten alphabets using a multi-layer perceptron classifier. User-generated data sets for the characters {a, b, c, d, h, i, j, k} have been used for conducting the experiments with the raw pixel values as the input features. The project aims at experimentally creating two models one to classify just {a, b} and another to classify all the characters by exploring the effect of varying various parameters like image size, number of hidden layers, number of neurons in each layer, etc on the classification performance of the system. The project also aims to address the issues involved in tackling the problem of multi-class classification with one vs all classifiers.**

***Index Terms*—Character Recognition, Neural Networks, Multi Layer Perceptron**

I. Introduction

OCR stands for Optical Character Recognition, i.e, the process of converting any physical text, either printed or handwritten into electronic data. It has come a long way from only recognizing certain near-perfect text mostly from the English alphabet in a considerable amount of time, due to the lack of access to modern high speed processors or GPU's, limited memory and relatively small databases, to almost instant recognition of random handwritten characters from multiple languages. This has been made possible due to the advancements in the aforementioned fields, access to innumerable data-sets and implementation of Machine learning (deep learning if the number of hidden layers > 2) techniques. Today, this technology is used everywhere from a simple pdf to document converter, preserving books and medical records by digitization and storing on multiple servers, recognizing license plate etc. Though there is a near perfect accuracy for printed scripts, efforts are being made to improve the accuracy of the handwritten text for both online and offline character recognition [1].

Two techniques have been majorly used throughout the research in the field of text recognition. The first involves extracting complex features from the handwritten text and using computer vision techniques for the recognition. This often turned out to be computationally expensive and not easily generalized. With the recent advancements in technology, the use of neural networks for classifying and recognizing handwritten text has become popular. Neural networks have good generalization power which do not require feature extraction [2].

Using Neural Networks offers inherent flexibility unlike other prediction techniques which pose certain restrictions on the distribution of the input variables. As our problem statement involves a processing of a large number of input parameters, Neural Network is a suitable choice to proceed. The following section describes our approach, where in we shall discuss the selection of training and testing data, network architecture, iterating over multiple parameters and selecting suitable parameters which result in high accuracy. Depending upon the type of problem, we define two models, one to classify 'a' and 'b' and the other to classify the eight letters '{a,b,c,d,h,i,j,k}' as specified, which is mentioned in the upcoming sections as follows.

II. Approach & Implementation

A. Image Pre-processing (Normalization & Flattening)

When sampling from train_data.pkl, we realized that about 250 samples were either rotated, distorted or did not serve as good representation of their corresponding labels. We had to remove these samples as they were negatively contributing to our trained model. We had to balance the cleaned data in order to have equal representation of all labels in our set so that the model is trained in a balanced way.

Also, the training data set contains hand-written characters which are of varying sizes. Since we are using a multi-layer perceptron algorithm which needs all inputs to be of a fixed size, each data sample needed to be normalized. We decided to normalize each raw image before feeding it to the network. A size of 100x100 was chosen for normalization as it gave the highest accuracy as compared to other sizes as mentioned in Table V. The method used to achieve this was skt.resize().

B. Input Data

It should be noted that the input images are in the form of a 2-D matrix. But the neural network consists of a single input layer and so, our input should be of the same form. To get the appropriate input dimension, we flattened the 2-D input matrix data to a 1-D matrix by appending every row of the original matrix to the first row in the matrix and obtained our final input vector.

C. Neural Network Architecture

Multi-Layer Perceptron Classifier can be considered as a neural network based logistic regression classifier. It consists of an input layer, various hidden layers and an output layers. Each layer consists of a predetermined number of neurons and for each layer, every neuron is connected to neurons in the subsequent layers. The number of neurons in the hidden layers is determined experimentally depending upon the performance of the system. The input to each neuron is weighted and a bias is added during the forward-pass of data through the network. The weights were updated at each iteration using the back-propagation algorithm to minimize the error of the system. Increasing the training efficiency requires determining the training error as soon as possible and avoiding getting stuck in local minima of the cost function, avoiding over-fitting and minimizing the generalization error [3]. Optimization over multiple iterations is the only way to solve the problem of minimization as there is no proposed solution in polynomial time. To achieve high performance of the classifier, hyper-parameter tuning was carried out. Following are the parameters that were varied across various specifications to achieve maximum accuracy:

- Number of hidden layers: These are layers between the input layer and output layers of a neural network. They consists of varying number of neurons. The input is transformed by travelling through the various hidden layers using the weights applied.
- Number of neurons: Each hidden layer consists of several neurons which are the basic perceptron unit of the neural network.
- Learning rate: The learning rate of a neural network defines the step size it takes to converge to the global minima to minimize the error.
- Momentum: This parameter is used to change the learning rates by increasing the step size if more than 2 steps are taken in the same direction.
- Activation function: These are used to make the classifier learn non-linearity present in the feature space. The activation functions decides the output of each neuron based on a set rule after calculating the weighted sum of all inputs and adding a bias to it.
- Solver: This is the technique used to optimize the wights of the network. We use Stochastic Gradient Descent for our purpose.

Early stopping criterion is employed to determine the number of iterations required to train the neural network instead of using pre-determined number of iterations. The early stopping criterion employs an approach wherein the algorithm continues to train the neural network over multiple iterations until the validation error is reduced to a value close to or below a pre-defined threshold. Generally weight decay is the regularization method used to control and prevent over-fitting where in the strong weights are penalized, by the addition of penalty to the cost function. The penalty factor, which minimizes the error is estimated by employing the cross validation technique.

D. OnevsRest Classifier

One of the issues with the single MLP classifier was that it works well for 2 class problem i.e., ‘a’ and ‘b’, but takes a performance hit when used for multi-class problem. We realize that the logistic regression models binary classification and we need to employ other techniques to extend it for multi-class classification. Two of the techniques employed in the literature are One vs Rest and One vs One classification. We use the one vs rest approach wherein one class is taken as positive and rest are considered as negative for each classifier. Essentially, it involves training ‘n’ classifiers for ‘n’ classes [4]. The n-classifier predicts probability of respective class and class with highest probability is selected. Therefore, we use this concept of ensemble learning where we use multiple MLP classifiers to fit one classifier per class for each class.

E. Training and Validation

Since we had only training data available, we resorted to k-fold Cross-Validation to train and validate our neural network. The results from the MLP Classifier was fed into the OneVsRest Classifier, which fits only one classifier per class. In this, each class is fitted against other classes for each classifier. All training parameters are varied inside the MLP Classifier function arguments. The OnevsRest classifier returns the model which is then trained on the data split using k-fold Cross Validation. Our experimental setup included 5-fold, 7-fold and 10-fold Cross Validation and every time we increased the k-value, we observed an increase in accuracy during validation. The accuracy increased by 3% when increasing the k-value from 5 to 10.

III. Experimental Results

To evaluate the neural network, different hyper-parameters were varied through several experiments and a thorough analysis was performed by varying a single parameter and keeping all the other test parameters constant to achieve the ideal configuration for classification of the 'AB' i.e., {'a','b'} and 'All' i.e., {'a','b','c','d','h','i','j','k'}classes based on the accuracy achieved. The experimental procedure is described in the following section -

Learning Rate	Momentum	Validation Accuracy (AB)	Validation Accuracy (All)
0.01	0.025	94.291	74.5101
0.025	0.025	92.609	80.103
0.025	0.05	93.801	80.474
0.05	0.025	93.993	79.818
0.05	0.05	93.099	80.598
0.1	0.05	95.569	81.9101

TABLE I: Varying Learning Rate and Momentum without Cross Validation

Learning rate and momentum are two key parameters that effect the performance of the neural network. Hence observing the effect of the variation of these parameters on the performance of the neural network was a crucial task. The following parameters were kept constant for the following analysis:

Solver function: 'sgd', Activation function: 'tanh', Hidden layers=3, No. of neurons per layer=100, Image resolution=70x70. We found that the combination of learning rate=0.1 and momentum=0.05 gives the best validation accuracy.

Learning Rate	Momentum	Validation Accuracy (AB)	Validation Accuracy (All)
0.01	0.025	93.4611	71.4954
0.025	0.025	93.397	73.0620
0.025	0.05	94.355	72.59366
0.05	0.025	94.0575	70.3811
0.05	0.05	94.291	68.66925
0.1	0.05	94.7603	73.3688

TABLE II: Varying Learning rate and Momentum with cross validation

Also, we wanted to test how variation of the learning rate and momentum effected the validation accuracy when we employ 5-fold cross-validation. The parameters were kept constant as before. We observed that learning rate=0.1 and momentum=0.05 gives best validation accuracy but these are less compared to without cross validation. Although, we observed that on increasing the number of folds of cross-validation, this accuracy increases.

No of Neurons per layer (N)	Validation Accuracy (AB)	Validation Accuracy (All)
20	94.824	81.901
50	94.845	80.717
70	94.781	81.293
100	94.973	80.7547
200	95.151	73.7349
300	94.952	72.2706

TABLE III: Varying No of Neurons in all hidden layers without cross validation

As neural network accuracy depend on the number of perceptrons in an hidden layer, we tested to check how the change in number of neurons can effect the performance of the neural network.For this we had kept the following parameters as constant.Solver Function: sgd, Activation function : tanh, momentum=0.05,learning rate=0.1 and 3 hidden layers.We observe that we get the best validation accuracy when each layer has 100 neurons.

No of Neurons per layer (N)	Validation Accuracy (AB)	Validation Accuracy (All)
20	94.8242	74.5801
50	93.887	72.9328
70	94.9733	70.5426
100	95.1011	74.2248
200	94.6964	70.8494
300	94.0362	72.4483

TABLE IV: Varying No of Neurons in hidden layers with cross validation

We varied the number of neurons with 5-fold cross validation to check for variation in accuracy. We kept the same parameters as above variations in number of neurons without cross validation. We found the best performance in terms of validation accuracy to be for 100 neurons per hidden layer. The validation for AB dataset is similar for with and without cross-validation experiments. Though, for the ALL dataset we see an initial decrease in accuracy when using cross-validation. This accuracy increases on increasing the number of folds as discussed below.

Image Resolution	Validation Accuracy (AB)	Validation Accuracy (All)
20x20	93.5037	71.5277
50x50	95.1863	72.4483
70x70	94.5473	71.9799
100x100	94.9094	74.0633

TABLE V: Varying Image resolution with cross validation

We varied the image resolution to check how image resolution effects the neural network. For this we kept the following parameters constant: Solver Function: sgd, Activation Function: tanh, hidden layers=(100,3), alpha=1e-5, momentum=0.05, learning rate=0.1. Image resolution of 100x100 gave the best validation accuracy.

Activation Function	Validation Accuracy (AB)	Validation Accuracy (All)
relu	94.3769	73.1589
tanh	94.3343	75.5813
identity	94.2279	71.7702
logistic	93.8658	33.4463

TABLE VI: Varying Activation Function with cross validation

For varying the Activation function we had fixed the following parameters as Solver Function=sgd, alpha=10e-5, learning rate = 0.1, momentum = 0.05, hidden layers = (100,3), resolution = 70x70, With 5-fold Cross Validation. From the results we observed that 'Relu' and 'tanh' were giving equally good performances. We observe that tanh has an edge over Relu and hence we chose tanh as out activation function.

K-fold	Validation Accuracy (AB)	Validation Accuracy (All)
5	94.8668	73.4173
7	94.2917	74.7739
10	94.5269	77.1693
15	98.0392	80.1356
20	99.0295	82.6711

TABLE VII: K-fold cross validation vs Normalized accuracy (AB,All)

K-fold cross-validation technique was used to better train the model and to check for any overfitting issues. We saw that for the AB dataset the classifier started overfitting the data after 10 folds. This can be due to the small size of the AB dataset. For the ALL dataset, we found that the accuracy of the classifier increased steadily as we increased the number of folds with the best accuracy of 82% at 20 folds.It should be noted that using Cross-Validation does not yield a significant improvement when validating for the 'AB' model,when k varies from 5 to 10 but as k approaches 15, over-fitting is

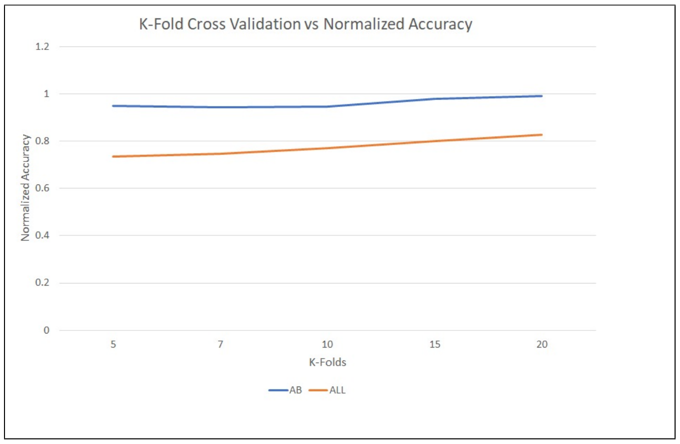

Fig. 1: K-fold cross validation vs Normalized accuracy (AB,All)

observed because the data-set is not expansive enough to accommodate 15-fold Cross Validation. But when the same is performed for 'All' model, we observe that the validation accuracy increases and can accommodate higher k values without over-fitting because now the data-set is expansive enough.

Conclusion

In view of the multiple experiments carried out for this research, we can conclude that the Multi Layer Perceptron classifier is an efficient classifier for provided classes of hand-written alphabets. From the results, we conclude that for the classification of the $\{a,b\}$ letters, with the settings consisting of "the image size being 100x100 with kfold value as 5 using the sgd solver, alpha value of 1e-5, 3 hidden layers consisting of 100 neurons each, early stopping set to true with momentum of 0.05 and the 'tanh' activation function, image size = 100x100" gives an accuracy of 94.86% and nearly 82% for the 'All' dataset, with a change in the k value from 5 to 20. This is true because we were able to obtain high accuracy without performing any major changes to the dataset or extracting any image features. We believe that steps such as advanced preprocessing techniques involving feature extraction,deeper and more complex network architectures could be employed to improve the accuracy for the given problem.

References

[1] C.-L. Liu, "Normalization-Cooperated Gradient Feature Extraction for Handwritten Character Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1465–1469, Aug. 2007.

[2] T. Hazra, R. Sarkar, and A. Kumar, "Handwritten English Character Recognition Using Logistic Regression and Neural Network," vol. 5, pp. 6–391, Jul. 2015.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overtting," p. 30.

[4] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, Aug. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320311000458