# Handwritten Character Recognition using Machine Learning Methods

Sai Swetha Kondubhatla
*University of Florida*
Gainesville, FL
skondubhatla@ufl.edu

Nikhil Reddy Kortha
*University of Florida*
Gainesville, FL
nikhil.kortha@ufl.edu

Keerthana Bhuthala
*University of Florida*
Gainesville, FL
kbhuthala@ufl.edu

***Abstract*—Character recognition has been one of the active and challenging research areas in the field of image processing and pattern recognition due to its diverse applications. Many machine learning techniques have been employed to solve this problem. In this paper, we focus on detecting English alphabets with the use of many machine learning approaches. The most famous machine learning approaches are Convolutional neural networks (CNN), Multi-layer Perceptron (MLP), deep neural network (DNN). In this paper, three approaches – KNN, MLP, and CNN are implemented and compared in terms of factors such as accuracy and performance. User-generated data sets for the characters a, b, c, d, g, h, i, k have been used for conducting the experiments with the pixel values as the features. The results show that among the four approaches, CNN is the most accurate algorithm it has a 94% accuracy rate.**

***Keywords—Handwritten Character Recognition; Pattern recognition; CNN; Multi-Layer Perceptron; KNN; Random Forest; Neural networks***

## I. INTRODUCTION

Character recognition is one of the practically important and challenging issues in the field of pattern recognition. Handwritten character recognition is the ability of a computer system to detect handwritten inputs like digits, characters etc. Several applications of Character recognition include mail sorting, bank processing, document reading. Character recognition helps us access and store information, search information in these images in a very efficient manner. It immensely contributes to the progression of automation process thereby making the interface between man and machine better in many applications.

Handwritten character recognition problem is classified into two types; on-line and off-line handwriting recognition models. [12] In the on-line method, 2D coordinates of successive points and the order of strokes made by the user are represented as a function of time to determine the characters. In offline methods, the writing of the user is physically captured with the help of a scanner and the entire writing is available as an image. The online methods are proved to be superior in recognition of characters due to the availability of previous character information [1]. Many approaches have been provided by the researchers for offline handwritten character recognition [2 – 6]. Some of them successfully achieved high recognition accuracy levels.

In this paper, an offline character recognition system is presented. The image is converted to binary and resized to 50x50 pixels and these pixels are used to train the model which is employed in performing recognition tasks.

The organization of the paper is as follows: Section II describes some theoretical concepts such as KNN, CNN, MLP, which are used in this paper. Section III describes the implementation details of the algorithm. Section IV gives an overview of how the experiment is carried out and the results of the experiment. Paper concludes in Section V.

## II. THEORETICAL BACKGROUND

### A. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) is a neural network-based classifier. Multilayer perceptron consists of an input layer, an output layer and a hidden layer as shown in *Figure 1*. Each layer consists of nodes which are called neurons and each of the neurons are connected to the subsequent layer. [8] The number of nodes in the input layer is determined by the number of features in the data. The number of neurons in the output layer is determined by the number of unique classes in the data. The number of neurons in the hidden layers and the number of hidden layers are the parameters that are to be determined experimentally. The connection between two nodes consists of a weight. [9] These weights are updated are every iteration using a supervised learning technique called backpropagation algorithm.

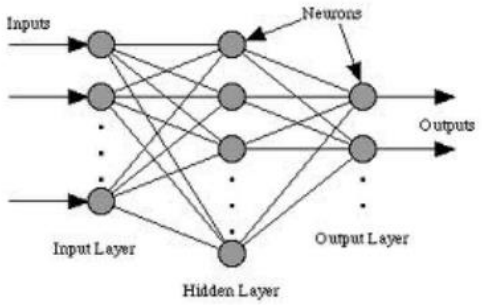


*Figure 1*

### B. K Nearest neighbours (KNN)

K nearest neighbours is a simple classification technique which is used to classify the objects on the basis of most similar or K closest training samples in the feature space. The distance for each testing sample from all the training samples is calculated, and the majority vote of neighbours is used to classify the class of the object. The distance metric used for KNN classifier is Euclidean distance metric which is defined as follows:

$$d(x,y)=\sqrt{(x-y)^T\sum^{-1}(x-y)}$$

Where x and y are two instances and $\Sigma^{-1}$ is the covariance matrix for x and y.

### C. Convolutional neural networks

The first layer of a convolution neural net is the input layer and the size of the input image is 50 x 50. The second layer is the convolution layer. CNN's use convolutions i.e. filter to detect the edges within an image. The filter moves

through the entire image and performs the convolution operation to detect any edges in the image. The output after performing convolution is called feature maps or convolutions maps. Then we pass these feature maps through a nonlinear activation function to help approximate the features within the underlying data. The famous nonlinear activation functions are ReLu, SoftMax, tanh. After passing through the activation functions, we perform max pooling in which we pass over the sections of the image and pool them into the maximum value of that section. This will help us reduce the features in the feature set that we pass to the neural network and the neural network works just like the multi-layer perceptron. The basic architecture of convolution neural network is shown in We add a dropout p, to reduce overfitting ----------

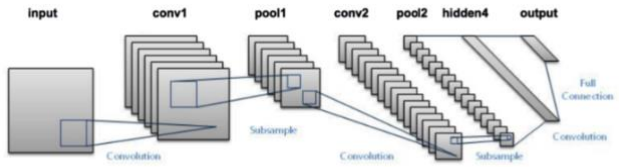


*Figure 2*

## III. Design and Implementation

### A. Data and preprocessing

Character recognition is a very extensive and comprehensive research area in the field of machine learning and computer vision [10]. The feature extraction technique plays an important role in character recognition [11].

In our dataset, Handwritten characters of 'a', 'b', 'c', 'd', 'h', 'i', 'j', 'k' from different people are taken and scanned. Also, another set of these specific characters were extracted from handwritten letters using image processing techniques. These two datasets are combined along with their labels and are used for training. Some of these images are shown in *Figure 3*. We have a total of 7847 images of different characters. These images were converted into a binary feature vector as shown in *Figure 4*. Since these images might be in different sizes, these images were padded with 0's to make them all of the same size i.e. 50 x 50. So, we have an array of size 7847 x 2500; each image has 2500-pixel values and these pixel values are used as features to the model.

We have experimented on the size to which each image must be scaled and after a number of trails, we found that if the image size is too small, the model doesn't have sufficient pixels to learn the shape of character completely thereby resulting in lower accuracy rate. On the other hand, if the size is too big, the model will have too many pixels to learn hence will take more time to learn and will learn much noise, thereby reducing the accuracy rate.

*Figure 3*



*Figure 4*

### B. Models and their implementations

***Multilayer perceptron:*** As discussed in section II, the multilayer perceptron has an input layer, hidden layer and the output layer. After preprocessing of the data, the labels of the train data need to be encoded, i.e. the label 3 should be converted to a vector [0,0,1,0,0,0,0,0] for building the model. Then we need to decide the architecture of the model. The implementation of MLP is as follows:

We have one hidden layer and the number of neurons in the hidden layer are determined experimentally. The input from the input layer is propagated through the network to the output layer and this calculated output is compared with the original output. The error, in our experiment it is mean squared error, is propagated backwards and the weights are updated. The activation function we use here is SoftMax, because, the SoftMax function divides the outputs between 0 and 1 such that the sum of all the outputs is equal to 1. Hence the output of SoftMax function is equivalent to the probability distribution; it tells us the probability of that input belonging to a particular class.

The learning rate of the MLP determines the step width of the gradient descent. If the step width is too small, there might be a possibility that we are stuck in the local minima and hence can never reach the global minima. Momentum term helps us to reach the global minima faster by taking bigger steps if there is a gradient descent. Using a momentum term, the weights are changes proportionally to how much they were updated the last iteration. It helps us roll through the flat parts of the curve without being stuck.

**Convolutional Neural Networks:** The working of the Convolutional neural networks is discussed in Section II. We defined the architecture for our project as follows. We have an input layer with 2500 pixels, and we defined 3 layers each of size 128, 64, 32, 10. So first, the 2500 pixels will be mapped to 128 neurons, 128 neurons will be mapped to 64 neurons and finally, these 64 neurons will be mapped to the output layer. At each of these layers, we have a nonlinear activation function ReLu. The reason for using ReLu is that ReLu is a nonlinear activation function with values between [0, inf] and its gradient is never saturated thereby converging faster when compared to other nonlinear activation functions like sigmoid/tanh. This can be seen in Figure 5. Also, there wouldn't be any expensive operations; ReLu can be implemented by thresholding a matrix of activations at Zero. The dropout value used to reduce overfitting will be decided experimentally. The initial value of dropout is set to 1 and gradually decreased until a point where there was a significant decrease in the prediction performance.

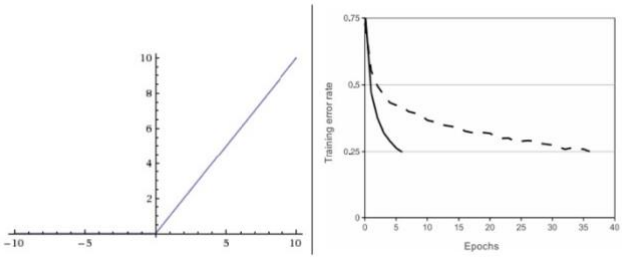


*Figure 5: Rectified Linear Unit (ReLU) activation function and A plot from [7] paper indicating the 6x improvement in convergence with the ReLU unit compared to the tanh unit.*

## IV. EXPERIMENTATION

After the above implementation of different neural networks, we performed various experiments on various parameters for different models.

### A. *K nearest neighbours*

The varying parameters in k nearest neighbours algorithm are the k value and distance metrics. The value of K is determined by using a technique called cross-validation, wherein we divide the data into an n number of folds and train the model by varying the k value. We need to find a k that fits the data appropriately to give the best accuracy rate during cross-validation. The accuracy vs k value graph for 5-fold cross-validation is shown in *Figure* ***6***. From the figure, we can say that for a k value between $10 \leq k \leq 15$ we get the highest accuracy rate of 94.64%. If the k value is too small, the error is high as we will be considering very few neighbours and the noise in the data isn't minimized. The data will have high variance and low bias. If the K value is more, since it will consider many neighbours, there is a possibility that some of the neighbours might belong to other class thereby increasing the error and reducing accuracy. Hence, it is important to choose the right value of k such that the noise in the data is minimized and it is small enough that the sample from other classes isn't included.

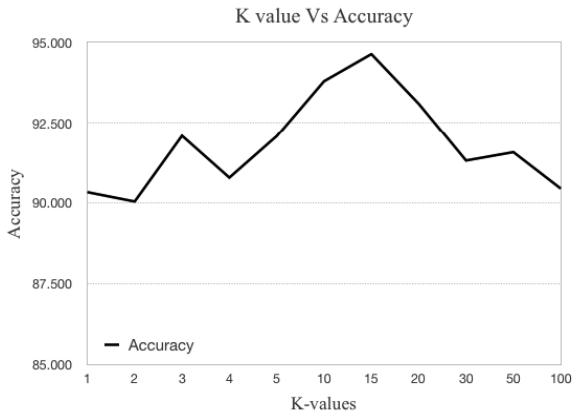


*Figure 6*

For the distance metric, we tried Manhattan, Mahabolis, Canberra distance metrics apart from Euclidean. There wasn't any noticeable difference except for Canberra. The accuracy decreased when compared to other 3.

### B. *Multi-Layer Perceptron:*

The accuracy of the model depends upon a number of parameters like the learning rate, number of iterations, initial weights, momentum. These parameters are determined experimentally.

*a) Learning rate:* Learning rate tells us how fast our weights change. If the learning rate is large, there will be larger weight changes on each epoch. This will make the network learn quickly. But, if the learning rate is too large, then, weight changes no longer give us any insights on how to approximate the gradient descent. The weights start taking large steps, resulting in never reaching the global optima. There will be huge oscillations which might make it even worse. On the other hand, if the learning rate is too small, the weights take smaller steps. This will lead to taking a larger number of iterations to converge. Ideally, we would like to choose the largest learning rate, since, the randomly chosen weight values would be far from optimal, so taking larger steps, in the beginning, helps us reach the area of global optima faster, but this might make things worse too. As we can see from *Figure* **7** that the learning rate is set to 0.001, the number of iterations to converge is 300. But when set to 0.1, the number of iterations is 3. If we further increase the learning rate to 40, the number iterations are around 70. To overcome these difficulties, we use the concept of momentum. Using momentum, we take into account, all our past steps to guide our next step. This technique helps us reach the local minima faster.

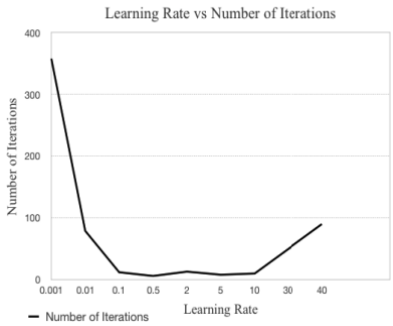


*Figure 7*

*b) Number of nodes in the hidden layer:* Selecting the number of nodes in the hidden layer is a very important sept for any neural network. If there are very few neurons in the hidden layer, then the neural net might not be able to classify the values correctly. On the other hand, if there are a large number of neurons in the hidden layer then the computation time increases, and this can also lead to overfitting. The strategy we used to find the optimal number of neurons in the hidden layer is, first assume a higher number of neurons and keep decreasing the number of neurons until the model is able to classify correctly. We started with 100 and after many experiments, we found out that for a number of neurons = 100 the accuracy is high.

*c) Learning rate vs Accuracy:* If the learning rate is too high, the accuracy is low. This is because the algorithm might keep oscillating between two maxima and might not reach the minima. On the other hand, if the learning rate is too low the accuracy will be high but, the number of iterations required to converge will also be high. The values can be seen in *Figure* **8**. After numerous experiments, we achieved an accuracy of 96.8 with 5 hidden layer neurons, the learning rate of 0.01 with SoftMax activation function.

Learning rate VS Accuracy

| LEARNING RATE | ACCURACY |
|---|---|
| 0.01 | 96.690 |
| 2 | 96.600 |
| 10 | 96.160 |
| 100 | 90.000 |
| 300 | 53.450 |
| 500 | 46.000 |
| | |

*Figure 8*

## C. *Convolutional Neural Networks:*

The accuracy of the model depends on various factors like the learning rate, drop out, momentum, number of iterations, activation function.

*a) Cost Function:* In previous neural network we used least squared error as the cost function to minimize the error, but here, we used the cross Entropy function as stated below where t is the target vector and y is the predicted output vector.

$$E = -\sum_{i=1}^{nout} (t_i \log(y_i) + (1 - t_i) \log(1 - y_i))$$

This is because, initially the output value calculated from feedforward network is very low and hence, in Mean squared error, then the weight will be changed by very small amounts and hence takes a little while to take bigger steps towards minima. But, in the case of cross entropy, the log makes it better and the magnitude by which the weight changes depends completely on the magnitude of the error incurred. Therefore, for larger derivates, the weights are adjusted by a higher value and for smaller derivates, the weights are adjusted by a smaller value.

*b) Dropout:* Dropout is a technique which is used in deep learning to reduce overfitting. It takes advantage of Ensemble learning methods without increasing the computation time. Every node in the neural net is either retained with a probability p or dropped with a probability of 1-p for each iteration thereby creating a new model for every iteration. If the value of p is low the model will vary a lot and if the value of p is high the model will be relatively stable. The effect of dropout on accuracy is shown in *Figure* **9**

Dropout vs Accuracy

| P | ACCURACY |
|---|---|
| 0.2 | 94 |
| 0.1 | 90 |
| 0 | 50 |
| 0.8 | 99 |
| 0.7 | 98 |

*Figure 9*

*c) Number of Iterations:* The more the number of iterations, higher the number of weight updates thereby reducing the error. After various experiments, we set the number of iterations in the range of 5-10 to allow the training loss to reduce enough to provide the best accuracy. This can be depicted in *Figure* **10.**

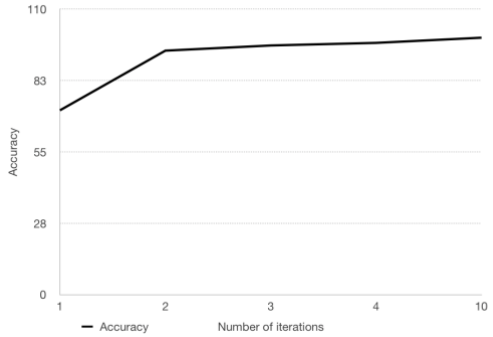


*Figure 10*

*d) Learning Rate:* Since Convolution neural network also uses backpropagation algorithm to update the weights, the effect of learning rate is same as the effect of learning rate on Multilayer Perceptron. After various experiments, the best learning rate found to obtain the highest accuracy is 0.01.

## REFERENCES


[1] G. R. Plamondon and S. N. Srihari, "On-line and offline handwritten character recognition: A comprehensive survey,"IEEE. Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, no. 1, pp. 63-84, 2000

[2] CL Liu, K Nakashima, H Sako and H. Fujisawa,"Benchmarking of state-of-the-art techniques," PatternRecognition, vol. 36, no 10, pp. 2271– 2285, Oct. 2003.

[3] M. Shi, Y. Fujisawa, T. Wakabayashi and F. Kimura,"Handwritten numeral recognition using gradient and curvature of a grayscale image," Pattern Recognition, vol.35, no. 10, pp. 2051–2059, Oct 2002.

[4] LN. Teow and KF. Loe, "Robust vision-based features and classification schemes for off-line handwritten digit recognition," Pattern Recognition, vol. 35, no. 11, pp.2355–2364, Nov. 2002.

[5] K. Cheung, D. Yeung and RT. Chin, "A Bayesian framework for deformable pattern recognition with application to handwritten character recognition," IEEETrans PatternAnalMach Intell, vol. 20, no. 12, pp. 382–1388, Dec. 1998.
[12] I.J. Tsang, IR. Tsang and DV Dyck, "Handwritten character recognition based on moment feature derived from image partition," in Int. Conf. image processing 1998, vol. 2, pp 939–942.

[6] U. Bhattacharya, and B. B. Chaudhuri, "Handwritten numeral databases of Indian scripts and multistage recognition of mixed numerals," IEEE Transaction on Pattern analysis and machine intelligence, Vol.31, No.3, pp.444-457, 2009.

[7] Srivastava, Nitish et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

[8] Pereira, F., Mitchell, T., & Botvinick, M. Machine learning classifiers and fMRI: a tutorial overview Neuroimage, 45(1), S199-S209, 2009.

[9] Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Steinbrecher, M., & Held, P. Multi-Layer Perceptrons. In Computational Intelligence (pp. 47- 81)

[10] Trier, Ø. D., Jain, A. K., & Text, T. Feature extraction methods for character recognition-a survey. Pattern Recognition, 29(4), 641-662.

[11] Koerich, A. L., Sabourin, R., & Suen, C. Y. Large vocabulary off-line handwriting recognition: A survey. Pattern Analysis & Applications, 6(2), 97-121.

[12] Liu, C. L., Yin, F., Wang, D. H., & Wang, Q. F. (2013). Online and offline handwritten Chinese character recognition: benchmarking on new databases. Pattern Recognition, 46(1), 155-162.