# Handwritten Text Recognition

***Abstract*—Our project is based on the idea of recognizing handwritten text. A deep learning architecture model has been essentially used to recognize English language handwritten characters. We used Convolutional Neural Networks (CNNs) to achieve the results of our project. Convolutional Neural Networks are used and trained in a supervised mode. CNN is a suitable and efficient technique for our proposed project which is the recognition of handwritten text. EMNIST Letters dataset has been used in our project. It has 145600 Handwritten English Characters. Several optimization methodologies are imported for improving the performance of CNN. Our proposed CNN is giving an average of 93% accuracy on testing data.**

***Keywords—Convolutional neural network, Handwritten, Character recognition, Deep learning, Handwriting recognition, EMNIST, MATLAB***

## I. Introduction

Recognition is a concept that wraps different areas: image recognition, face recognition, fingerprint recognition, character recognition, etc. Handwritten Character Recognition System is a smart system that has the potential to classify handwritten characters humans see. Character classification is a fundamental chunk of many computer vision and image possessing problems e.g. license plate recognition, optical character recognition, etc. The classification of handwritten characters is a very challenging task due to the fact of different handwriting styles of the writers.

Deep Neural Networks contains the input layer, multiple hidden layers that are non-linear and an output-layer. The quota of parameters that are trainable and connections is large. Having said that, the deep neural network requires a very large amount of set of examples to stop overfitting. Deep Neural Network has a class type that has a reasonably smaller set of parameters, efficient and much easier to train, that is Convolutional Neural Network (CNN). This is a multi-layered neural network based on a feed-forward approach that extricates properties and features from the given data e.g. images, sounds. CNNs are capable of learning from congested and complicated high-dimensional inputs, mappings that are non-linear from a colossal amount of data e.g. images and sounds. One of the positive points of CNNs is that it extracts the main features automatically, the features that are unchangeable and up to some point to shift and shape distortions of the given input characters. Another noticeable advantage of CNNs is the make use of shared wright in convolutional layers. This implies that the same filter is utilized for each given input in the layer. By utilizing the shared weight, the network reduces the number of parameters while performance is improved.

In our project, we used the famous EMNIST dataset. This is not a fresh subject and after a good few years, the EMNIST dataset is still famous and well-liked for validating and evaluating new algorithms. The project's code is written in MATLAB Computation software that comes with a Graphical User Interface with the help of add-ons Toolboxes of Deep Learning and Image Processing.

## II. Methodology

### *A. Dataset*

The EMNIST Letters dataset is used for character recognition that is based on NIST Special Database 19. It has sample forms of handprinted from about 3600 writers 810000 character images are separated from their original forms. For character classification purposes, some datasets are obtained from NIST. The most famous that is well-known to many around the world is MNIST database of handwritten digits that has been extended with letters to create the EMNIST dataset. The EMNIST Letters dataset has relied on images from the NIST database. In this dataset, some of the classes of the lower and upper cases have been merged to avoid errors leading to misclassification from upper and lower case letters. This results in lowering the number of classes to twenty-six from sixty-two for letters.

The dataset has a training set and a test set. The training set has 88,800 images and a test set contains 14,800 images altogether, which have been transformed into a black and white image [high contrast] with 28 *x* 28 pixels size during preprocessing step. Additional flip and rotate counterclockwise by 90 degrees were required for it to be used on the task. Although a preprocessing step can be put forward in order to reduce bias more.

MATLAB software's predefined functions is used to insert images, labels in our model, it is an easy way to parse data for MNIST and EMNIST. This provided us a platform to import images, labels straight into arrays with size ($m$, $n_x$) and ($m$, 1). In this $m$ is the number of examples in our dataset. And $n_x$ is 28 *x* 28 number of pixels in each image. The given images were reshaped into ($m$, 28, 28, 1) for the purpose of CNN model. Followed by the conversion of the list of label numbers into an array of one-hot vectors. It has a shape of ($m$, $n_y$). Here $n_y$ represents number of classes that $n_y$ = 26. The preprocessing final step is involving the division of pixel values by 255. This

is for the purpose of having a distribution of values between 0 and 1.

### B. Architecture

CNNs specialty is to process the two-dimensional (2-D) image. It has three important types of layers: Convolution layers, Sub-Sampling layers, and Output layer. Network layers have been given a structure of feed-forward. The process is based on such a style that each single convolution layer is following the sub-sampling layer. The very last convolution layer is followed by an output layer. Both convolution and sub-sampling layers could be considered two dimensional or 2-D layers. While the output layer is considered as a one dimensional or 1-D layer. A feature map is the output of a plane.

In the convolutional layer, every single plane is fastened to one or more feature maps of the previous layer. A connection is correlated with a convolution mask, which is a two-dimensional matrix of flexible entries which is called weights. Every single plane computes the convolution first in-between the two-dimensional inputs and the convolution masks. The outputs that we have from convolutions are added together, which is then being attached with an adjustable scalar that is called a bias term. To obtain the plane's result, we apply the activation function then. The output that we have from the plane is a two-dimensional matrix called a feature map. We call it a feature map since each single convolution output implies the existence of a visual feature that is at a given pixel location.

Usually, one or more than one feature maps are produced by the convolution layer. Each single feature map is fastened to precisely one plane in the up-coming sub-sampling layer. The number of planes we have in the sub-sampling plane is the same as we have in the earlier convolution layer. A sub-sampling plane cuts its two-dimensional plane into chunks of size 2 *x* 2 pixels that are non-overlapping. For every single chunk, the summation of four pixels is being calculated. Followed by before adding it to a bias term, this sum is being multiplied by weight that is adjustable. The output that we receive from this is gone through the activation function to create an output for a two times two block. Precisely, a sub-sampling plane minimizes its input given size by half with respect to each dimension.

In the very last convolution layer, the planes are well connected to one before the feature map. This layer makes use of convolution masks, which has the precise identical size as its input feature maps.

### C. CNN Layers

Convolution layers. The Convolution layer performs as a feature extractor that withdraws major features of the inputs e.g. endpoints, edges, corners. Rectified Linear Units (ReLU) is the activation function in our given approach, which applies the non-saturating. We implement the ReLU activation function to every single output of each convolutional layer and fully connected (FC) layer. The ReLU intensifies the non-linear properties of our decision function while not affecting the receptive areas of the convolution layer.

Pooling layers. We could have a pooling layer, after the effect of each convolutional layer. This layer takes a tiny block in a rectangular shape from convolution layer while sub-sampling it in order to get a single result from that block. The pooling layer minimizes the resolution of an image which minimizes the correctness of the translation of the shift and distortion effect. We have different ways to do this process of pooling, e.g. having an average or maximum, or a linear combination of neurons that have been learned in the block. The pooling layers we have will always be considered max-pooling layers which takes the highest of the block which is pooling. This process of max-pooling carrying out is on two times two-pixel window.

FC or Fully Connected layers. Ultimately, after the effect of several convolutional and pooling layers. The high-level scenario is achieved through FC layers in the neural network. An FC layer would attract all neurons in the preceding layer (in order to be fully connected) while connecting to each neuron it has. FC layers could not necessarily be resided spatially anymore. And we can visualize them in a one-dimensional way. So, we are not left with any CNNs layers after an FC layer. Ultimately, the FC layer has the composition of the SoftMax classifier that produces 26 output classes.

### D. CNN Optimization

Learning Rate. Learning rate α usage is in the during of updating the weight of architecture. It is an important parameter in realizing the success of generalization and convergence of neural networks. If our learning rate is too low that leads to very slow convergence and the opposite this approach would lead divergence.

Activation Function. We have used ReLU (Rectified Linear Units) activation in our project. ReLU implements the non-saturating. We implemented the ReLU activation function each convolutional layer and the FC layer's output. The Rectified Linear Units improves the properties that are non-linear of a function that makes the decision and of our whole network while not affecting the fields that are receptive of the convolution layer. ReLUs make the training of deep neural networks considerably quicker and increases and improves its ability in generalizing.

Adaptive Moment Estimation (Adam). This Adam algorithm is about updating weights and biases parameters in the way to minimize the error function while moving forward slowly in the way of a negative gradient of the loss function.

Mini batch. The Adam algorithm is about the evaluation of gradient. As a matter of fact, updates the available parameters while utilizing a subset of the given training set. This notion of a subset is called a mini batch. The way mini-batch works is dividing the dataset into small batches while computing the available gradient using a single batch. Afterward, go for an update while moving to the next batch then. Every single evaluation of a given gradient through mini batch is supposed an iteration. Going through an iteration, the present algorithm goes in mini steps in the direction of reducing the loss function. The notion of the epoch is about the fully passing of training algorithms over the whole training set utilizing mini batches.

The mini-batch specification is 200. The highest number of epochs is 10.

SoftMax classifier. This classifier can be used in different probabilistic many-classes classification methods. The SoftMax function is implemented on final output units of the network in order to achieve a probability value for each class.

### *E. Training*

Our neural network is being trained on the EMNIST Letters training dataset. The process that is used itself is called a supervised learning process. The training dataset has about 88,800 images. The dataset has uppercase and lowercase letters. This dataset has been well ready and prepared. The images that we have in the dataset have already been centered in a 28 *x* 28 image fashion. It is done through computing the center of mass of given pixels. While defining the image in order to position this point at the center of 28 times 28 field.

After having some research, we realized that LeNet is the convenient model when it comes to interacting with the convolutional network that involves the methodology of pooling layer, batch size and some tips for feature visualization.

Afterward, we started designing a neural network. We need one that could give us optimum outputs. Although, there is not a precise way that of searching the firm and correct model of Neural Network. In order to have a good one, we could use a trial and error methodology. Selecting different Neural Network models and train these models and noticing the accuracy that we receive from outputs. Twisting the kernel sizes that include the numbers of convolutional layers specifies that one needs to have more convolutional layers to have better accuracy. Having said that, changing the kernel size has little effect, also, sometimes, the accuracy decreases. Predicting the reason behind that is the hands of that convolutional layer out to feature extraction and feature transformation. We also invest some time by implementing another approach e.g. Adam in order to have a better result. The objective was to analyze the arrangement of the network, dataset, and parameter twisting it in our model. We also tried several optimization methods e.g. SGD and RMSProp. We have noticed that in real-world scenarios, the accomplishment of neural network classifier relies on the number of training iterations needed to train the network. If the number of training epochs is too small, that will lead to a poorly trained network due to the fact of underfitting of the network. Otherwise, so much training epochs outcome in poor generalization because of overfitting of the network. The selection of network learning iteration should be in a format that the network could be converged appropriately with less generalization error.

Having been finished with the training process, the neural network prepares in order to go through the testing process. The testing group gets from the EMNIST Letters dataset just like the training group. Altogether, it has 14,800 images with 26 classes of letters. The whole achievement of the processing unit is chosen by comparison of testing results with provided labels for the testing images. Each prediction of the label gets compared with the prechosen testing label. This situation stays like this for each instance in the training set to choose a whole percentage to explain the accurateness of the processing unit.

## III. Result

In this segment, the result of suggestive handwritten character recognition technique is estimated experimentally through arbitrarily test samples. The experiments are applied in MATLAB 2019b in a macOS environment having an Intel Core i5 2.6 GHz processor with 8 GB of Random-Access Memory.

Microsoft Paint application is being used for drawing handwritten character images. The character recognition could also be facilitated by scanning it through a scanner. This methodology itself is known as Image Acquisition. In order to make ready the images for the upcoming processing step, the given the format of the image is being changed to .PNG. A background that is full white or has some noise in it could be used to draw these handwritten images. Different colored ink could be used to draw samples.

The image that we drew could be imported as an input image. Followed by this image going through the preprocessing step. Firstly, the image needs to be changed into a greyscale image. By converting the RGB image into a greyscale image, it minimizes the saturation and hue information and keeps the required luminance. After finishing with this, it goes through the binarization process. The binarization step changes the grayscale image into a binary image. This step is an important one in the process because in it the pixel values are untied into two teams. We are only left with black and white colors which could be settled in a binary image. The objective behind the binarization process to decreases the unneeded info present in the image and at the same time securing the important information. It essentially conserves the effective information existed in an image. It should also essentially minimize the noise in the background affiliated with the image in a structured manner.

The same image goes through a test, supposedly called a connectivity test. It is for the purpose of testifies the maximum connected parts and the properties of each part, that is in the shape of the box. Figuring out the box, then each separate character is cropped into several sub-images that is data that is in a raw form for the upcoming character extraction procedure. The following step is handwritten character should essentially be cropped separately and resizing it into 28 *x* 28 images. In order to search for words, we would count pixels in between bounding boxes. After finding space that is more than a hundred fifty pixels that would imply that the word has reached a finishing point and should continue with a new word.

In order to recognize the character, The Graphical User Interface (GUI) is set already by ourselves. The initial step is to choose the drawn image. Followed by loading the image. It goes through the preprocessing steps, following by visual activations. This whole scenario is about how to put in an image into a convolutional neural network and visualizing the activations of several layers of the network. The mediums in initial layers comprehend straightforward features e.g. edges and color. The mediums in deeper layers grasp complex features. When we identified features in this way, we could study what the network has learned.

The final phase of the given recognition system, it imprints the correlated recognized characters in the systematic text form by calculating correspondent ASCII value though the recognition index of text samples.

One of the screenshots of our output which is shot by phone camera.

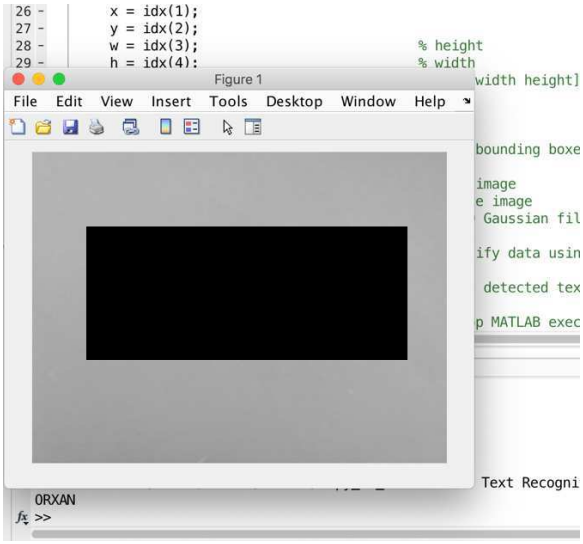


Fig. 1. Screenshot

## IV. Conclusion

The question of Handwritten Character Recognition for an English character is opened and an active research area that needs attention. This ideology of recognizing the English language characters through convolutional neural networks. If done comparison in-between deep learning architectures, Convolutional Neural Network is in a leading position in both images and big data. The objective of using deep learning was to take benefit from the potential of Convolutional Neural Network, which takes control of inputs with large dimension and utilizes and share their weights. After experimenting, we noticed a decent 93% classification accuracy rate on our testing images.

For further upcoming scenarios, we intend to manage and save the handwritten characters from people while preprocessing it for training. Following making our handwritten character dataset. And in order to train our dataset, we want to use the Capsule Neural Network model. Our intention also includes more optimization of our model. This will benefit us in efficiently running the model on our setups.

## Contributions

Our team members contribute equally and were involved in making the proposed model. The proposal and report were written together.

## References

[1] Cohen, Gregory et al. "EMNIST: An Extension Of MNIST To Handwritten Letters". Arxiv.Org, 2020, https://arxiv.org/abs/1702.05373.

[2] https://nl.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html

[3] https://mccormickml.com/2015/01/10/understanding-the-deeplearntoolbox-cnn-example/

[4] https://medium.com/tebs-lab/how-to-classify-mnist-digits-with-different-neural-network-architectures-39c75a0f03e3

[5] https://nl.mathworks.com/help/deeplearning/ref/trainnetwork.html

[6] https://nl.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[7] https://nl.mathworks.com/help/deeplearning/ref/trainingoptions.html

[8] Jayasundara, Vinoj et al. "Textcaps: Handwritten Character Recognition With Very Small Datasets". 2019 IEEE Winter Conference On Applications Of Computer Vision (WACV), 2019. IEEE, doi:10.1109/wacv.2019.00033.