International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-3S, February 2019

Handwritten Text Recognition: with Deep Learning and Android

Shubham Sanjay Mor, Shivam Solanki, Saransh Gupta, Sayam Dhingra, Monika Jain, Rahul Saxena

Abstract: ***This research paper offers a new solution to traditional handwriting recognition techniques using concepts of Deep learning and computer vision. An extension of MNIST digits dataset called the Emnist dataset has been used. It contains 62 classes with 0-9 digits and A-Z characters in both uppercase and lowercase. An application for Android, to detect handwritten text and convert it into digital form using Convolutional Neural Networks, abbreviated as CNN, for text classification and detection, has been created. Prior to that we pre-processed the dataset and applied various filters over it. We designed an android application using Android Studio and linked our handwriting text recognition program using tensorflow libraries. The layout of the application has been kept simple for demonstration purpose. It uses a protobuf file and tensorflow interface to use the trained keras graph to predict alphanumeric characters drawn using a finger.***

Index Terms—E.G – For Example, NN – Neural Network, RNN – Recurrent Neural Network, CNN – Convolutional Neural Network, EMNIST - Extended NIST dataset.

I. INTRODUCTION

A. Overview

Handwritten Text Recognition is a technology that is much needed in this world as of today. Before proper implementation of this technology we have relied on writing texts with our own hands which can result in errors. It's difficult to store and access physical data with efficiency. Manual labor is required in order to maintain proper organization of the data. Throughout history, there has been severe loss of data because of the traditional method of storing data. Modern day technology is letting people store the data over machines, where the storage, organization and accessing of data is relatively easier. Adopting the use of Handwritten Text Recognition software, it's easier to store and access data that was traditionally stored. Furthermore, it provides more security to the data. One such example of Handwritten text Recognition software is the Google Lens. The aim of our project is to make an application for mobile devices that can recognize the handwriting using concepts of deep learning. We approached our problem using CNN as they provide better accuracy over such tasks.

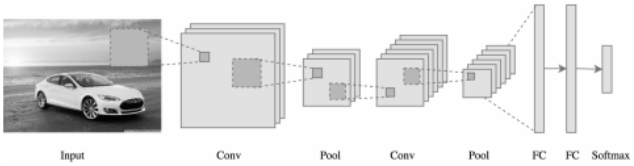


Figure 1.1 Architecture of CNN

B. Deep Learning

Deep learning can be called a sub-part of Machine learning which is mostly used for learning data representations. This recently exploded technology has improved state of the art techniques used in natural language processing, object detection etc. Deep learning finds out complex structure in massive data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters used in each new layer from the representation in the previous layer. A standard neural network (NN) constitutes of many simple inter-connected processors called neurons. Each neuron produces a sequence of activations. The activation of
input neurons takes place through sensors that perceive the environment whereas the others get activated through weighted connections from previously active neurons.

C. Convolutional Neural Networks

CNN image classifications take an input image, process it and classify it under certain categories (E.g., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). E.g., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image where 3 and 1 are the number of color values required to represent an image pixel.

i. **Convolution** – The main building block of CNN is the convolutional layer which is a mathematical operation to merge two sets of information. The convolution is applied on the input data using a convolution filter to produce a feature map. The first layer to extract features from an input image is convolution.

Manuscript published on 28 February 2019.
* Correspondence Author (s)
Shubham Sanjay Mor, Dept. of Computer Science and Engineering, Manipal University Jaipur. (E-mail: shubhammor@outlook.com)
Shivam Solanki, Dept. of Computer Science and Engineering, Manipal University Jaipur. (E-mail: 98shim@gmail.com)
Saransh Gupta, Dept. of Computer Science and Engineering, Manipal University Jaipur. (E-mail: saranshgupta121@gmail.com)
Sayam Dhingra, Dept. of Computer Science and Engineering, Manipal University Jaipur. (E-mail: sayam.dhingra26@gmail.com)
Monika Jain, Dept. of Information Technology, Manipal University Jaipur. (E-mail: monikalnct@gmail.com)
Rahul Saxena, Dept. of Information Technology, Manipal University Jaipur. (E-mail: rahulsaxena0812@gmail.com)

ii. **Striding** – The number of shifts over the input image is called stride. Example- When stride is 1 then we move the filters to 1 pixel at a time.

iii. **Non-Linearity** – ReLU stands for Rectified Linear Unit for a non-linear operation. The output is:

$$\{F(x) = \max(0, x)\}$$

iv. **Pooling Layer** – After a convolution operation we usually perform pooling to reduce the dimensionality. This enables us to reduce the number of parameters, which in turn shortens the training time and reduces chances of overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact. The most common type of pooling is max pooling which just takes the maximum value in the pooling space. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the maximum value in the window. The largest element from the feature map is selected by using this operation.

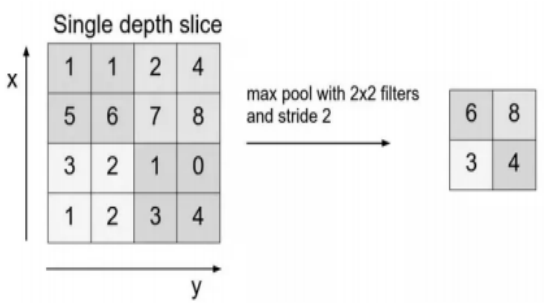


Figure 1.2 Stride of 2 pixels

II. LITERATURE SURVEY

A. *Prior Works*

We have used the dataset of EMNIST [1], there have been several accomplishments that has been achieved using this dataset. Even before using Deep learning, Handwritten text recognition has been made possible, however their accuracies were really low or they had a relatively small dataset as said by Line Eikvil [2]. In this paper, usage of OCR has been discussed such as in Speech Recognition, Radio Frequency, Vision systems, Magnetic Stripes, Bar Code and Optical Mark Reading. A popular machine learning task is classifying the MNIST dataset, which is dataset of numbers. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis by Simard, Steinkraus and Platt is a valuable paper for understanding usage of convoluted neural networks (CNNs). For word recognition, a Paper by Pham et al., used a 2-layer CNN which fed into a bidirectional recurrent neural networks (RNN) with Long Short-Term Memory (LSTM) cells [3]. The best model implemented, according to us, is by Graves and Schmiduber with a multidimensional RNN structure. [4] Another paper on 'Handwritten Text Recognition' by M.J. Castro-Bleda dealt with dataset with slanted words as well and corrected them during pre-processing. [5] Development of English Handwritten Recognition Using Deep Neural Network by Teddy Surya and Ahmad Fakhrur uses a Deep Neural Network model having two Encoding layer and one SoftMax layer on the EMNIST dataset. Their accuracy using DNN was way better than the earlier proposed patternnet and feedforwardnet ANN (Artificial Neural Networks). [6] Handwritten text recognition can also be achieved on basis of Relaxation Convolutional Neural Networks(R-CNN) and alternatively trained relaxation convolutional neural networks (ATRCNN) as done by ChunPeng Wu and Wei Fan. [7] Our model achieved accuracy over 87 percent using Convolutional Neural Networks from Keras[8] library.

B. *Dataset Description*

The EMNIST dataset is a collection of handwritten alphanumeric derived from the NIST Special Database 19. Each image is converted to a 28x28 format and dataset structure that directly matches the dataset is used. The training set has 697932 images and test set has 116323 of uppercase and lowercase alphabets and numerals from 0-9 which are mapped to their corresponding classes. The test set and training set is available in the form of list within list. Each item of outer list represents an image and inner list represents the intensity values of 784 pixels (because size of image is 28 x 28 pixels) ranging from 0-255. The test images as well as train images have a white foreground and black background. Both the test images as well as train images are horizontally flipped and rotated 90 degrees clockwise. Y train and Y test both are arrays which contain number ranging from 0 to 61 as there are 10 numerals from 0-9 and 26 uppercase and lowercase alphabets each which adds up to 62. [1]

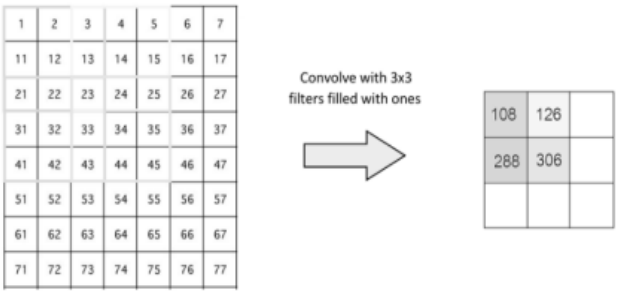


Figure 1.3 Max Pooling

III. METHODOLOGY

The aim of our project is to make an interface that can be used to recognize user handwritten characters. We approached our problem using Convolutional neural Networks in order to get a higher accuracy. Several researches have been undertaken to improve the accuracy of alphanumeric character prediction. Our research will include that to some extent. But our main focus will be providing a GUI that can be used to easily predict characters for further use. We plan to do so using tensorflow [9] and keras. Firstly, we will define a model that will be trained with the Emnist dataset which contains over 690,000 train images and will be validated using the test dataset provided by Emnist again.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

We will then freeze the keras graph and link it to our android application which has a simple layout that takes a user hand drawn input and will predict the alphanumeric character. Later, we will work on python to segment characters from an image of a word and predict each character using our model.

A. *Data Pre-Processing*

Prior to training our sequential model with our train images, we were required to apply some pre-processing techniques to our data in order to make it more fitting for our model and for our app (Our android app provides a 1-D array as an input).

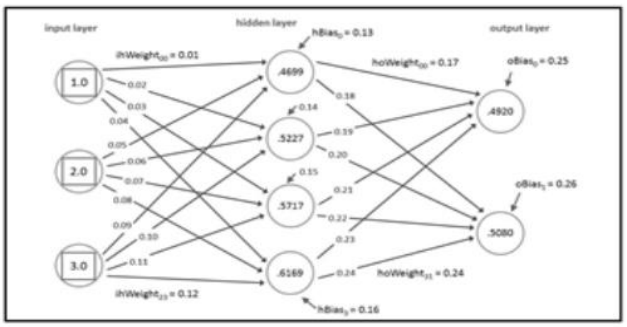
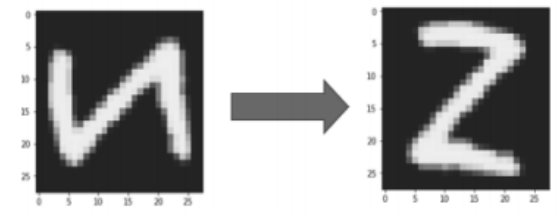


Figure 3.2 Model Visualization

i. Normalization - In image processing, changing the range of pixel intensity values is known as normalization. Some of its applications include correcting photographs with poor contrast due to glare, for example. Normalization is sometimes called contrast stretching or histogram stretching. Here we use the range of float values between 0 and 1.

ii. Rotating and Reversing the data –
As mentioned in the Introduction of the report, the Emnist dataset contains 697932 train images and 116323 test images, all rotated and reversed. In this section, we reverse the images and later rotate them 90 anticlockwise. We achieve both these operations using an inbuilt NumPy transpose function.

iii. Input image filtering – Here, we discuss various pre-processing techniques that we have applied as to get an efficient classification. Firstly, we resize and image in the ratio 4:5(using inbuilt python libraries) and convert it into grayscale. This helps in increasing spaces between the written characters so that we can identify characters that were written extremely close to each other. Next, we perform thresholding on the image. This way, we partition an image into a foreground and background. After this the image is converted to binary format because contours can be found easily in binary images. For this we use the threshold function of cv2. We then dilate the image. This expands its interior objects and increases chances of an efficient character segmentation. Further, we apply a gaussian blur to our image. This results in each pixel in our image being multiplied with a gaussian kernel. This reduces noise and detail. Now our image is ready for segmentation.



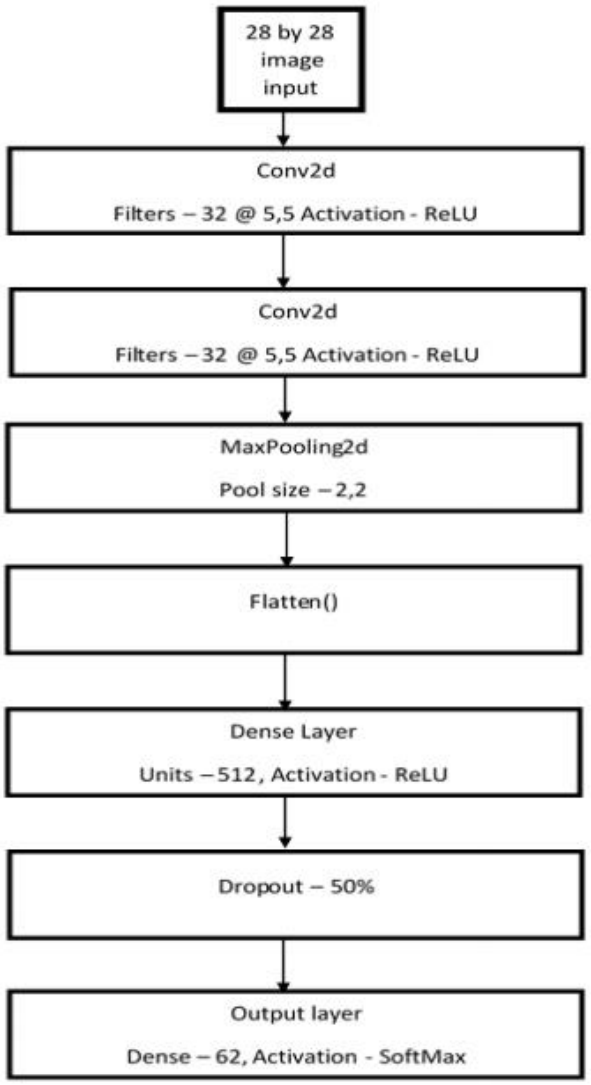


Figure 3.3 Block diagram

B. *Model*

The CNN model for recognizing handwritten characters is constructed using the KERAS library of python and tensorflow [9] backend. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow [9], CNTK, or Theano as backend. The most widely used type of model is SEQUENTIAL. The sequential model consists of linear stack of layers. The same has been used for developing the handwritten text recognition model. Defining a model was our primary task.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Defining a model basically means adding different layers to the stack. Layer 1: The first layer we added is the reshape layer that took an input of (784,1) and reshaped it into (28,28,1). Layer 2 and Layer 3: Convolutional layer - The function of convolutional layer is to find out distinctive features from the given input matrix which is an image of handwritten text which has been already normalized. The convolutional layer uses a filter matrix which is a combination of zeros and ones. The filter matrix slides over the input matrix and keeps the count of elements which matched with the elements of the filter matrix. The matrix so created is called FEATURE MAP. This convolutional layer takes input matrix of shape of (28,28,1). This layer uses a filter of size (5,5) and a stride size of 1. It creates 32 feature maps as output using 32 different filters. It uses ReLU as the activation function. ReLU is preferred over SIGMOID because sigmoid is a slow learning function. The purpose of using the convolution layer twice is to extract more features to improve the model's accuracy. Layer 4: MAX POOLING layer. Maxpooling is done by applying a max filter to (usually) non-overlapping sub regions of the initial representation. When the images are too large, we would need to reduce the number of trainable parameters. Pooling is done for the sole purpose of reducing the spatial size of the image. Here we have used a pool size of (2,2) with stride size 2. Layer 5: FLATTEN layer. The function of the flatten layer is convert all elements of the feature maps matrices to individual neurons which will serve as input to the next layer. Layer 6: DENSE layer or the input layer which accepts the output from the FLATTEN layer as input. The value held by a neuron is called activation of that neuron. Every unit of input(neuron) has activation corresponding to intensity of pixel. The output of this layer is determined using the activation function which is ReLU (Rectified Linear Unit) in this case. The function of activation function is to activate the neurons of the DENSE layer. This layer has 512 neurons. Every neuron has a bias associated with it. More over every neuron has weights corresponding to each and every input. This can be visualized with the help of picture illustrated above. In the picture above each neuron of hidden layer has 1 bias and 3 weights associated with it corresponding to each input neuron. The neurons are activated based on the outcome from the activation function, if its activation is greater than 0.5 the neuron is activated else the neuron remains inactive. Layer 7: is the dropout layer. The function of the dropout layer is to remove some of the neurons or the unwanted features that can make the model bulky and increase the training time. It is also helping in avoiding overfitting. Here we have a used a dropout layer which removes 50 percent of neurons. Layer 8: The last layer of the model is the dense layer which is also called the output layer. The last layer has 62 neurons corresponding to 0-9 numbers and A-Z alphabets in uppercase as well as lower case. The neurons are activated in the similar fashion, but this time SOFTMAX is used as the activation function. SOFTMAX is a logistic classification function which is similar to the SIGMOID function. Like SIGMOID function it also used to calculate probabilities of different classes. The only difference is SOFTMAX is used for multiclass classification whereas SIGMOID is used for binary classification. The equation for the SoftMax function is given below.

$$\left\{ \sigma(z)_{\{j\}} = \frac{\left\{e^{\{z_{\{j\}}\}}\right\}}{\left\{\sum_{\{K=1\}}^{\{k\}} e^{\{z_{\{k\}}\}}\right\}} for\ j = 1, \ldots, k \right\}$$

SOFTMAX will calculate the probabilities of each target class over all possible target classes. The target class can be later easily identified using the calculated probabilities for the inputs. SOFTMAX provides us with a range of 0 to 1 with the sum of all probabilities equal to 1. The class with the highest probability is the target class. The block diagram of the model is illustrated as following:

IV. RESULTS

A. Training

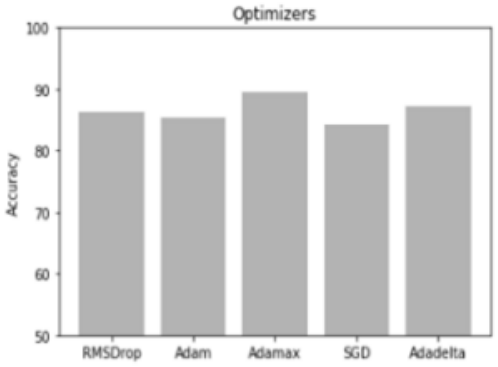


Figure 4.1 Comparison of various keras optimizers

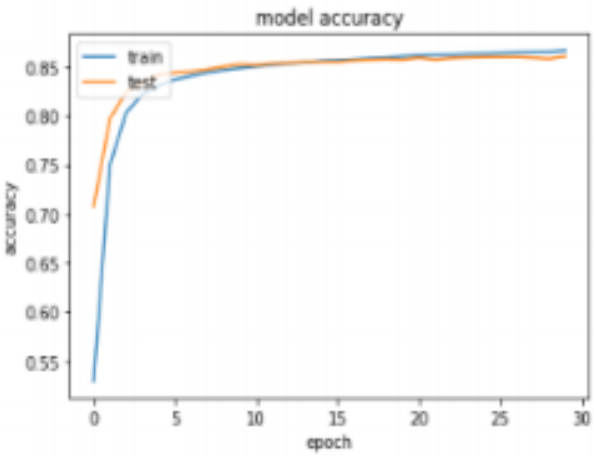


Figure 4.1 1 Conv2d and 2 Dense Layers

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

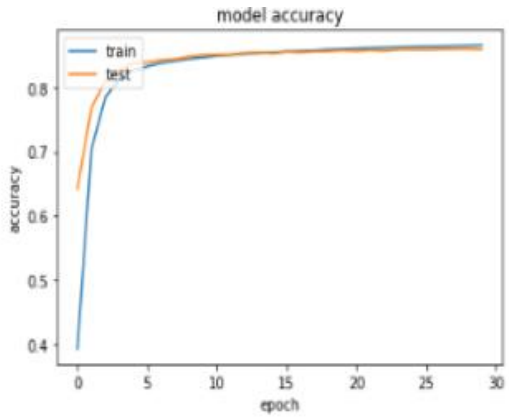


Figure 4.3 Model Accuracy

We trained our models with different optimizers available for keras [8]. As the bar chart illustrates, the models we used included RmsProp, Adam, Adadelta, Adamax, SGD. The highest accuracy obtained was when we used Adamax for our experiment. So, we decided to use it for the purpose of our research on the Emnist dataset. Other optimizers gave accuracies somewhat very close to Adamax, but Adamax substantially reduced our training time as well. The training on a personal computer (RAM - 16 GB) took about 20 hours with 512 units in our dense layer. This training time can be reduced in a manifold way by adopting the usage of GPU. The model trained with our Emnist dataset which had been pre-processed and optimized for training beforehand. We have categorically distributed the train and test labels and flattened out train and test arrays for easy input into our model and android application. The Adamax optimizer is extremely popular for training large models and has provided us with robust results. We tried out different hyperparameters for our keras model:

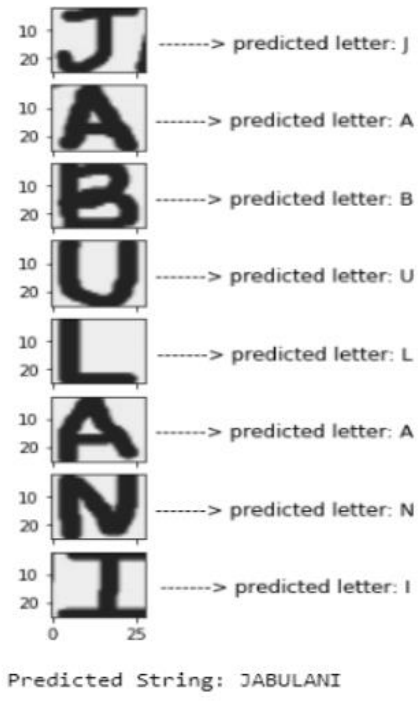


Figure 4.5 Sample prediction using our model

Figure 4.5 Original Image

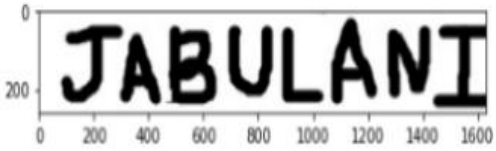


Figure 4.6 Resized Image with extra space

Figure 4.7 Image with thresholding, dilation and Gaussian blur filter applied

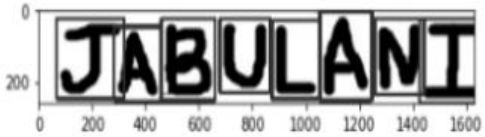


Figure 4.8 Segmented Image

Figure 4.7 Prediction using application

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

We then stuck with a model with 2 Convolution layers as it gave us the highest accuracy. Model accuracy increased with each epoch. At first, it increased exponentially, and later had a minute and steady growth. At the end of our training, we obtained 89.53 percent accuracy which is a little shy of the maximum accuracy obtained using the Emnist dataset.

The evaluation of the model was done with the test data Emnist provided. Our model was able to predict with an accuracy of 87.1 percent. This accuracy can be increased further in future research work by pre-processing the dataset even more and by adding new hyperparameters to the keras model. The lower the loss, the better the model (unless the model has overfitted to the training data). We have calculated loss on our train and test data. In case of neural networks, the loss is usually negative log-likelihood and the residual sum of squares for classification and regression respectively. Then naturally, our objective was to reduce the loss functions value with respect to the model's parameters. The loss with our keras [8] model during training began at 0.68 with the first epoch and ended up at 0.31. As the model chapter previously stated, our classifier contained two Convolution and a single Maxpooling layer with a dropout layer and a dense layer with 512 units. We tried tweaking this by adding a few more back to back convolution layers but this didn't work well for our model and resulted in a lower accuracy.

B. Character Segmentation and Prediction

We begin by inputting an image. The image can be of a single character or a word. We use OpenCV to work with images for this research. Using inbuilt OpenCV library functions, we find contours in the image. After finding contours, we create rectangular bounding boxes around each character in a copied image. This is done because if we create boxes in the original image, the boxes may overlap with each other and hinder the performance of the classifier. Contours can be defined in a simple manner as a curve joining all the continuous points (along the boundary), having same color or intensity. They prove to be a useful tool for shape analysis and object detection. For better accuracy, we use binary images. findContours() function modifies the source image that's the reason we send a copy of image. After we create boxes around each identified character, we extract ROI's(Region of Interest) from the image. Since the size of each character might be different, we resize each image into a 28*28 image using OpenCV again so that this image can be used as an input to our model classifier. For instance, size of 'J' might be bigger than that of 'N' in 'JABULANI' as shown in Figure 4.4. An Example of Pre-processing and segmentation is illustrated below. Once segmentation is completed, we provide each 28*28 ROI as an input to our model and use the converted result to display the outcome in a formatted manner as illustrated in the last figure.

C. Android application

The android application developed for the purpose of easy hands on usage of this project was built with SDKtools 27.0.1 and gradle version 4.4. The android application has a simple layout with a drawview, two buttons and a textview and is inspired by the project uploaded by Sourcell: Link - https://github.com/llSourcell/A_Guide_to_Running_Tensorflow_ Models_on_Android. The drawview is a separate class entity that handles user input. A user can draw on the white canvas. Finger touch movements are used to identify the positions where a black pixel is supposed to be drawn. One button clears the drawview, another predicts the character drawn on the drawview. The android application was linked with our machine learning project created on python using tensorflow. TensorFlow libraries were added to the libs folder and the necessary changes were made in the gradle to import tensorflow[9] classes into our android studio project. After training our model in python, we had frozen its graph into a Protobuf file to be used for android import. We have added this Protobuf file as an asset. Now let's discuss how the application works. Once the user draws a figure on the drawview, we convert this figure to a bitmap using Bitmap class in java. A float array of size 784 is declared and all the bitmap pixel values are stored in that array. Each of the 784-pixel values is normalized by division to bring it between the range of 0 and 1, that is the values used to train our model. Then, the Tensorflowinterface node is filled with the float array. We then call the Protobuf file located in our asset folder and provide the array to the input node. The interface returns us a result array with the probability of the bitmap being each of the 62 characters via the output node. We store this result in a new array of size 62. Now, we display the character with the highest probability using the textview. The android application can be found online for further research work and for the purpose of its improvement at: https://drive.google.com/open?id=1JTgc4ZM0n3UVixXa-HpBoU2mjlhlQ5oV

V. CONCLUSION

Using modern day techniques like neural networks to implement deep learning to solve basic tasks which are done with a blink of an eye by any human like text recognition is just scratching the surface of the potential behind machine learning. There are infinite possibilities and application of this technology. Traditional OCR used to work similar to biometric device. Photo sensor technology was used to gather the match points of physical attributes and then convert it into database of known types. But with the help of modern-day techniques like convolution neural networks we are able to scan and understand words with an accuracy never seen before in history. We used the EMNIST data set to train our model and tested different optimizers to finally select Adamax as it not only yielded a high accuracy with each epoch on our train data but also our test data. A further application of accurate text OCRs is to help the partially sighted and the blind in the absence of braille. By also integrating a simple text to speech module in the app the user can point his phone to any text which will then read out the text for the user. A dedicated device can also be built for this purpose with a more sophisticated image recognition system which can identify objects to tell the user how many steps to walk in which direction and even when to stop and turn.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The EMNIST datasets, a suite of six datasets, considerably increased the challenge faced by employing only the MNIST dataset. Even though the structure of EMNIST dataset is similar to that of MNIST, it provides a higher number of image samples and output classes and an even more complex and varied classification task. It was thus obvious to use it as the backbone of our project. Without the use of EMNIST data set it would be practically impossible to achieve this accuracy. Our current android app requires the user to draw/ write the text on the screen and then analyses it to identify the alphanumeric character. The app can be developed further to import images from the gallery in the user's device and identify the text in present in those images. Another development can be to convert text to speech to further increase the applications of the mobile app. The android app can be developed further using googles cloud natural language API which provides natural language understanding technologies like, sentiment analysis, entity recognition, entity sentiment analysis, and text annotations to understand the text further and better by providing dictionaries that will rectify the mistakes made by the model to provide a meaningful result. Another development can be the use of googles cloud vision API to increase the accuracy of the data read and even to identify different objects. [1]

REFERENCES

1. G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," arXiv preprint arXiv:1702.05373, 2017.
2. L. Eikvil, "Optical character recognition," citeseer. ist. psu. edu/142042. html, 1993.
3. A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in Advances in neural information processing systems, 2009, pp. 545–552.
4. V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. IEEE, 2014, pp. 285–290.
5. S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 4, pp. 767–779, 2011.
6. T. S. Gunawan, A. F. R. M. Noor, and M. Kartiwi, "Development of english handwritten recognition using deep neural network," 2018.
7. C. Wu, W. Fan, Y. He, J. Sun, and S. Naoi, "Handwritten character recognition by alternately trained relaxation convolutional neural network," 2014 14th International Conference on Frontiers in Handwriting Recognition, pp. 291–296, 2014.
8. F. Chollet et al., "Keras," https://github.com/fchollet/keras, 2015.
9. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Man´e, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vi´egas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)