Handwritten Text Recognition using Image Processing and Deep Learning

CS419M Course Project
Instructor: Prof. Abir De, CSE, IITB

Gopalan Iyengar
UG 2nd Year Student
Department of Mechanical Engineering
IIT Bombay
Roll No: 19D170009

Shreyas Nadkarni
UG 2nd Year Student
Department of Electrical Engineering
IIT Bombay
Roll No: 19D170029

Aayush Srivastava
UG 2nd Year Student
Department of Electrical Engineering
IIT Bombay
Roll No: 19D070002

***Abstract*—Handwritten text recognition finds a lot of application in today's world like reading postal addresses, bank check amounts, annotating handwritten documents and many more. With an increasing demand of digital services scanning forms, documents becomes a necessity. However the computers cannot understand our handwritten text directly hence a handwritten character recognition is required. This work is an attempt to develop a deep learning model using image processing techniques from OpenCV and convolutional neural networks using the PyTorch library to recognise handwritten English text. The model has been trained on a portion of the dataset and tested on the other part, followed by the integration with image processing. The complete details of the model, our approach, results and related discussion is presented in this report.**

I. Introduction

As the world we thrive in is getting digital day-by-day, the preferred mode of documenting literature, research and other general work is shifting to typewritten format from the traditional handwritten one. However, a lot of handwritten material still exists and needs to be made compatible with its digital counterparts. This requirement to link the handwritten text with electronic devices is where the techniques for handwritten text recognition come into picture. The recent trends in Machine Learning and the Deep Learning Boom have paved the way for the development of intelligent systems which can not only automate complicated human tasks but also use the outcomes in a constructive way in a greater system. This project utilises the well known Machine Learning Library PyTorch and the image processing library OpenCV. A CNN model has been trained and tested on the EMNIST Dataset, and integrated with OpenCV to take an input as a photograph of handwritten text and predict the characters written. This project has been done as a part of the Minor Course: CS419M: Introduction to Machine Learning, offered by Prof. Abir De, Department of Computer Science and Engineering, IIT Bombay.

II. Approach Adopted For The Task

A. Data Processing

The EMNIST Dataset used for the project is the extended version of the MNIST dataset. It is a collection of images of handwritten characters (A to Z, a to z, 0 to 9) having 62 classes of labels. The data is in the form of 28 x 28 pixel grayscale images. This dataset's ByClass split was downloaded and was split into training and testing data and transformed to a torch 'tensor' for numerical operations. The data contained 697932 samples in the training set and 116323 samples in the training sest amounting to a total of 814255 samples. Using the DataLoader funtion of PyTorch, the data was fed into the model.

B. Training The CNN Model

First the model was trained taking arbitrary values of hyperparameters. A CNN architecture having 4 convolutional layers was declared and ReLU activation was used, and finalized after trial and error of multiple convolutional modules. A single fully-connected layer was used with a arbitrary number of units. This model was trained on the training data and the accuracy was calculated on the testing set. To improve the performance of the model, some hyperparameter tuning was necessary.

C. Hyperparameter Tuning

The hyperparameters of our model needed to be tuned to optimal values. Some of the hyperparameters in a CNN model are learning rate, number of hidden layers, number of units in each layer, momentum, number of epochs, batch size, activation function, etc. Due to limitations of time and computing power we decided to optimise only the architecture of the neural network, learning rate and the number of epochs. Hyperparameters like momentum can be tuned in order to improve the model even further. We chose a default batch size of 32 and the activation function used was ReLU. After this extensive trial and error working, we found a model which worked best. We chose this model's hyperparameters

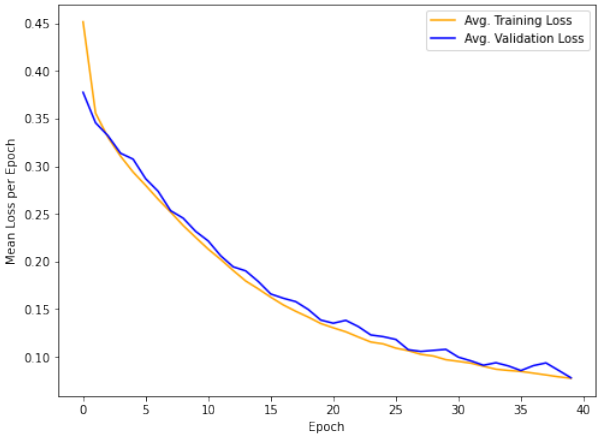

Fig. 1. The training and validation losses vs Epochs

as the final values. The FCN part of the neural network was made of two hidden layers of number of units 1984 and 992 respectively. The learning rate was chosen to be 0.0001. It was seen as mentioned in [2] that an "overcomplete" first hidden layer, i.e. a first hidden layer with number of units greater than the length of the input vector usually works better. Therefore for an input vector of 1568 (coming from the CNN part of the model) we chose a first hidden layer size of 1984. Also, the learning rate is the most significant hyperparameter in training a deep learning model and this must be tuned at the end after tuning others since the optimal value can change and it is preferred to have a value for learning rate which is within a factor of 2 of the optimal value. A "learning rate schedule" can be chosen to make the learning rate decreasing with time (decreasing in each subsequent iteration), however since the optimizer we chose was the Adam Optimizer, we did not prefer to use the scheduler.

```
ModelOfNet(
  (convolve): Sequential(
    (0): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU()
    (10): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fullyconnect1): Linear(in_features=1568, out_features=1984, bias=True)
  (fullyconnect2): Linear(in_features=1984, out_features=992, bias=True)
  (fullyconnect3): Linear(in_features=992, out_features=62, bias=True)
  (ReLU): ReLU()
)
```

Fig. 2. The Architecture of the Deep Learning Model

The above graphic is not completely accurate, as the ReLU unit exists after the first two linear layers after the convolutional sequential module (depicted accurately).

D. Testing the Model

Since the model hyperparameters were optimised, it was to be tested on the testing samples of the EMNIST data set which were set aside for this purpose. The tesing samples were fed into the model and an accuracy of 84.3 % was achieved. The testing data was not used in training the model and hence and accuracy of 84.3% on the testing dataset indicates that the model has not overfit the training set. This was also apparent from Fig. 1, where the training and validation loss both decreased and there was not need of early stopping which would have been the case if we took too many epochs and the validation loss crossed its minimum and increased with number of epochs.

E. Image Processing

After testing the model, the next task was to take a completely new image of handwritten text, other than the EMNIST Dataset and check whether the model worked on the new text. We used the library OpenCV to take inputs in the form of images and used contours to separate the words into characters. Each character was converted to a 28 x 28 pixel sized image, and preprocessed in a similar fashion to the EMNIST Dataset, as required by the trained CNN model. It was seen that the model did not work so accurately but some letters were correctly predicted. This was probably because the EMNIST dataset had a particular kind of handwriting and kind of images and the model did not perform well on images of different colour composition or texture. This can perhaps be countered in future work by including a wide variety of data and tuning more hyperparameters to increase the accuracy further while using same strategies to prevent overfitting.

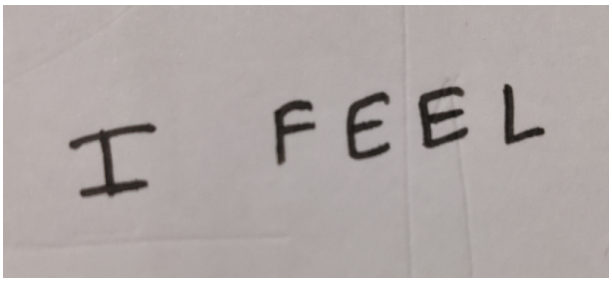

Fig. 3. Input to the Image processing block

The above image is given as input to the image processing block that performs the operations on the image as explained in the Image Processing section to segment all the characters.

First, the image was converted to gray-scale and Canny Edge Detection was performed. This returned a set of contours of the edges of the letters, which were then encompassed by a bounding box, and filled with solid colour. Then, Canny Edge Detection was performed again and the outermost contours per letter were bounded by rectangles.

As we can see in the figure the block bounds each character with a rectangle. These rectangles then can be segmented out and then preprocessed and resized to 28x28 image size and

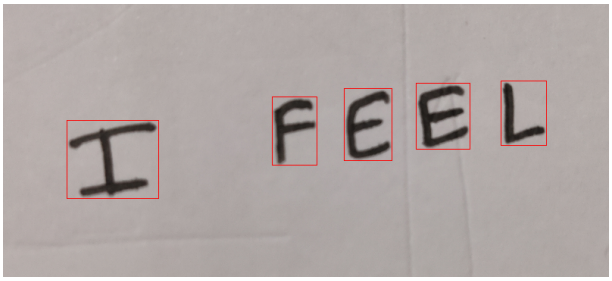

Fig. 4. Character segmentation result after Image processing block

can be given to the model as an input on which it can predict the character.

III. Results and Discussion

The model gave a accuracy of 84.83% on training on 40 epochs. The average train and validation cross entropy loss during training was 0.07740 and 0.07787 respectively.

Thus, we have obtained a model with an impressive performance on the test set, which is very reliable for images within the dataset distribution. However, it is not robust, as an image from a different distribution, or adhering to a different set of rules governing the letters' boundary is not handled well by the model.

Currently the model takes only a word as a input image and classifies it. What further improvements that can be done to make the model more robust is the image given to a model is a page segmented image that contains only handwritten text paragraphs. Moreover we can incorporate to take a image which has both printed text and handwritten text as an input and our code segments out the handwritten text from the printed text, recognise it and convert it into digital text. More training data can be added that includes vast hand writings, cursive hand writings etc and other languages beside English to make the model more robust and general.

References

[1] Cohen, S. Afshar, J. Tapson and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2921-2926, doi: 10.1109/IJCNN.2017.7966217.

[2] Yoshua Bengio, "Practical Recommendations for Gradient-Based Training of Deep Architectures", Sept 16th, 2012

[3] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[4] https://docs.opencv.org/master/d9/d8b/tutorial_py_contours_hierarchy.html

[5] https://www.kaggle.com/crawford/emnist