Handwriting recognition - CIS 572 - Machine Learning

By Alex Xiao Cai - axiaocai@uoregon.edu & Tien Dinh - tvd@uoregon.edu

Abstract

During the recent development of artificial intelligence (AI), classification tasks such as Image Detection, Object Recognition, Tracking and Navigating have been made possible with the help of deep learning and neural network algorithms. Similarly, handwriting recognition is one of the important fields that could be solved in the near future given the strong advancement in this machine learning field. Handwriting recognition, also known as handwritten text recognition, is the computer's ability to take readable human handwriting input, then intelligently translate to the correct letters/words/sentences. In our paper, we have implemented our own handwriting recognition using MNIST [1] and EMNIST [2] datasets, then applied a variety of popular Convolution Neural Network (CNN) models: AlexNet [3], ResNet [4], VGG [5]. The main goal is to reach the highest accuracy with the models with decent runtime and lower loss in training. We hope to contribute our work to solve the handwriting recognition problem as well as the machine learning field in general.

Keywords: Machine Learning (ML), Handwriting Recognition, Handwritten Text Recognition, MNIST datasets, EMNIST datasets, AlexNet, ResNet, VGG, Convolution Neural Network (CNN)

1. About

Handwritten recognition can be divided into many subcategories. However, in our paper, we categorize the task as handwritten digit recognition and handwritten letter recognition. Each of the tasks has its own simplicities and difficulties. Both the digit and text in our research are of latin words from the English characters. There are other languages that come with different characters and symbols such as German, French, Spanish, etc. who use a slightly different Latin or Chinese, Japanese, Korean who uses their own character sets, .. and other languages from different countries. But for the lack of a detailed dataset of those languages, we mainly focus on English latin words and then develop our model in the future.

Handwritten digit recognition is the computer's ability to realize human's handwritten digits from papers, images, touch screens, ... then classify them into 10 classes from 0 to 9. This task is the most simple and a good starting point due to the small scope of the problem. Handwritten digit recognition has a variety of important usage in daily life, for example: online bank checking needs to process the amount of money in human handwriting, vehicle plate could use digit recognition technique for tracking and identifying vehicle property, and so on. Despite the simple nature of the task, digit recognition introduces some challenges in itself. Human writing and even imprinted writing of different cultures have distinguished styles. A "1" - one and a "7" - seven have subtle differences. Some

ones - "1" start with a line up-right connecting with the line straight down, thus very similar to the number seven. Number six - "6" and number nine - "9" are basically of the same shape but flipped vertically. To distinguish between the numbers, we first start with MNIST datasets that provide 60,000 training images and 10,000 testing digit images of 28x28 pixels. The learning of digit recognition will be applied later to letter recognition.

Handwritten letter recognition is the computer's ability to recognize human's text handwriting that can be classified into over 52 classes: lowercase alphabet (a-z) and uppercase alphabet (A-Z). Letter recognition has a bigger scope than digit recognition and is often associated with digit recognition and symbol recognition to solve the general purpose of handwriting, which combines into handwritten text recognition. Handwritten text recognition introduces more complicated matters in distinguishing letters and digits. To name a few, a "o" or "O" and a zero - "0" are usually the main discussion due to how similar the characters are. A two - "2" and a "s" or "S" are also tough cases for machines or even humans to distinguish. Last but not least, a seven - "7", a one - "1", letter "i" or "I", and letter "l" are extremely alike and often confusing to computers if there are no natural language processing tasks involved. Human brain with the natural image recognition, language processing language can easily tell the word "like" in context but it is not so easy for the computer. This is a different field that includes smart AI that understands the context, but in this paper, we mainly focus on characterizing proper handwriting and setting a standard. To achieve the task, we introduce the EMNIST dataset which provides 814255 images including 697932 training sets and 116323 sets of combined letters and digits in 1x28x28 pixels. Note that the dataset in EMNIST flipped so we need to do pre-processing (image rotation) before feeding the data to the model.

In terms of model, we investigated three main different models in our work, AlexNet, ResNet and VGG which will be discussed in section 3. The accuracy of the model is to be understood as the higher accuracy, the more accurate the prediction. A low accuracy of model or low confidence is not applicable to industrial application. As mentioned before, recognizing the correct amount of money in handwritten text in bank checking is crucial. Missing or miscorrecting a single digit may cause tremendous damage. Despite the challenge, a functioning digit recognition is extremely helpful in automating the task, reducing the amount of manual work and also reducing the nature of human making mistakes. The digits character is often accompanied with letters/words describing to clarify the exact amount of money. In our work, we aim to provide the highest accuracy model, display the top three results and choose the correct one accordingly. We will also provide a comparison between different models based on result, loss validation to select the best model with low error, high accuracy, high confidence in predicting that can be used in a real application.

The paper is organized as follows. We first discuss the related work that has been done in the field, their use of datasets, models and implementation and

the motivation behind. We then discuss the datasets that we are using, what they offer and how to process them. Next, we provide our own implementation based on different models. In the further sections, we will discuss the results and accuracy in detail. Following the result, we will conclude the performance of the models and select the best approach. Last but not least, we will present future works and what could be done to improve our work, as well as how to use our work. The last section offers citations and references to this paper.

2. Related Work

There have been multiple efforts in investigating handwriting recognition by machine learning [6]. Convolution Neural Network (CNN) has been widely used due to the nature of the problem. There are three general layers. The convolutional layer builds the network, extract the image feature and maintain the feature map. The subsampling layer is used to change the features into detailed, smaller size results. The fully connected layer contains input/output and hidden layers. The number of layers in CNN can be different from each implementation, some might include a softmax layer to present output based on probability. Recent work on this paper [7] shows an impressive result of 99.89% accuracy by applying a similar concept using CNN with hyper training parameters, Adam optimizer and on MNIST database. Another paper [8] also achieves a good result of 99.15% accuracy on 40 batch size , 8 epochs. The same layers are applied across many researches: convolutional layer, ReLu, Max Pooling, Padding and Stride, fully connected layer and softmax. This demonstrated that CNN is well fitting and far better than the general classifiers. The results could be improved with more convolution layers and hidden neurons in the network.

In terms of dataset, similar to MNIST, there is NIST [9], an older database that the US National Institute of Science published including sets of character images. NIST provides 800,000 character images from 3,600 writers. Beside character images, there are several more datasets providing English word images. The IAM dataset [10] contains 13,353 images of handwritten text created by 657 writers. The text is divided to 1,539 handwritten pages including 115,320 words. This dataset contains mostly standard English paragraphs compared to single letter and digit in MNIST and EMNIST. Apart from English characters, there is RIMES dataset [11] that includes 12,723 pages of French language. There is also HIT_OR3C [12], a dataset of handwritten Chinese characters. HIT_OR3C has 6,825 classes produced by 122 subjects aand 832,650 samples of Chinese characters, with a total of 909,818 images. There are more datasets other than those that were introduced here, each representing a sample of a particular language for a wider range of handwritten text recognition.

3. Exploring the dataset

In order to implement the handwritten recognition, first it is imperative to know the details of the dataset that it is being worked on. For this case, the EMNIST contains a total images number of

814255, 697932 of these are for training, and 116323 are for testing or evaluation. In this set, there are a total of 62 classes which includes the entire set of alphanumeric characters which is from A-Z including small and capital letters and 0-9. Additionally, each image is (1,28,28) meaning that is 1 channel with size of 28 by 28 pixels. However, the imageset is not correct as seen in fig. 1, and it requires a transformation before being used.

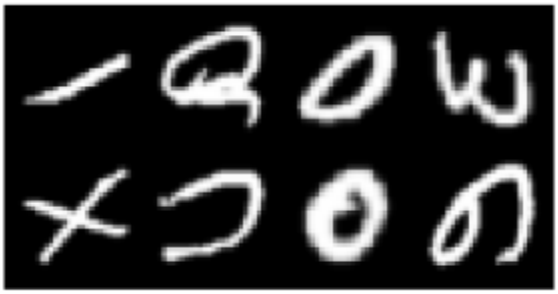

Figure 1. Dataset Before Transformation

So, in order to correct the imageset, pytorch's transformation utilities are being used. The exact code for this transformation is depicted below.

```
transform = tt.Compose([lambda img: tt.functional.rotate(img, -90),
                        lambda img: tt.functional.hflip(img),

tt.ToTensor(),
                        ])
```

In the previous, the order of action of the transformation is as follows:

1. A rotational transformation of -90 degrees (counterclockwise) as the image is "laying" flat.
2. A horizontal flip as the image is mirrored along the horizontal axis.
3. A tensor transformation for a numerical representation of the image.

After the transformation, the imageset can be used for training and testing. A sample of this imageset after transformation can be seen in fig. 2.

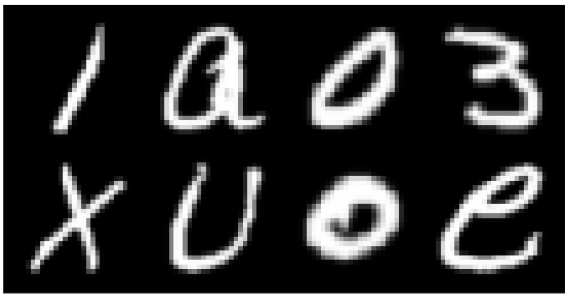

Figure 2. Dataset After Transformation

4. Approach

In order to choose the adequate neural network model for the recognition system, we tested several neural network models. The models that we choose to evaluate are the following: AlexNet, ResNet9, ResNet50, VGG11, and VGG16.

For AlexNet, it was chosen because of its influence in the computer vision field as it won the Imagenet large-scale visual recognition challenge in 2012. AlexNet is a convolutional neural network that is 8 layers deep. The architecture of this neural network can be seen below.

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Figure 3. AlexNet Architecture [13]

The ResNet models were chosen because of its simplicity and high accuracy in other models. Additionally, ResNet are the “backbones” of many other neural networks focused on computer vision tasks. As there are many ResNet variants, ResNet 9 and ResNet 50 were chosen to have a comprehensive review on their performance as the number of layers are significantly different from each other. A typical ResNet architecture can be seen below.

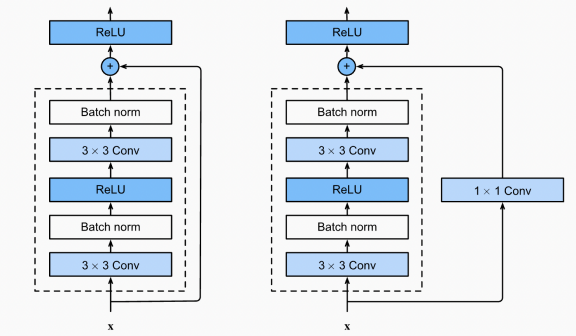

Figure 4. ResNet Architecture [14]

The VGG models were chosen as it is a denser neural network and its accuracy on other applications were considerably high. Additionally, VGG is often studied to develop denser neural networks. The selected VGG models are 11 and 16 in order to have a comparison between the same model but different depth. VGG16 was chosen instead of a deeper model because of physical limitations in this case Google Colab’s GPU memory. A deeper VGG model was not possible as the VGG16 already takes around 10 Gb of GPU memory and high running time. The model architecture for VGG16 can be seen below.

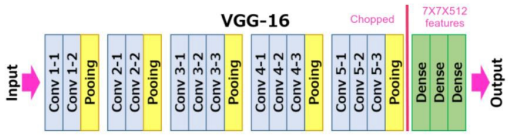

Figure 5. VGG Architecture [15]

5. Result

This section discusses in detail the results and conclusion of several network models. By comparing these models, a final model is chosen to be tuned for the recognition system. The results of this consist of a learning rate curve to check for uniform learning across several models, an accuracy curve to see the accuracy score at each epoch, and a loss training and validation curve to understand the behavior of the model.

5.1.1. AlexNet

For this model, the following parameters were chosen.

- Batch size: 400
- Number of epochs: 8
- Optimizer: Adam
- Max learning rate: 0.01
- Weight decay: 1e-4

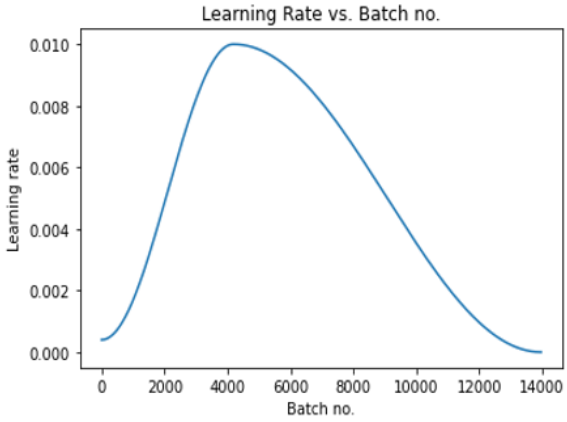

Figure 6. AlexNet Learning Rate

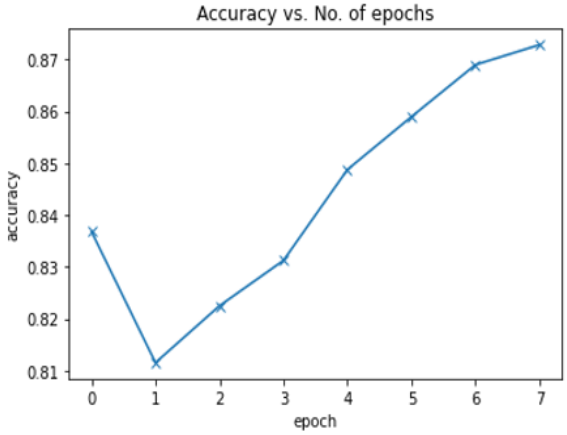

Figure 7. AlexNet Accuracy

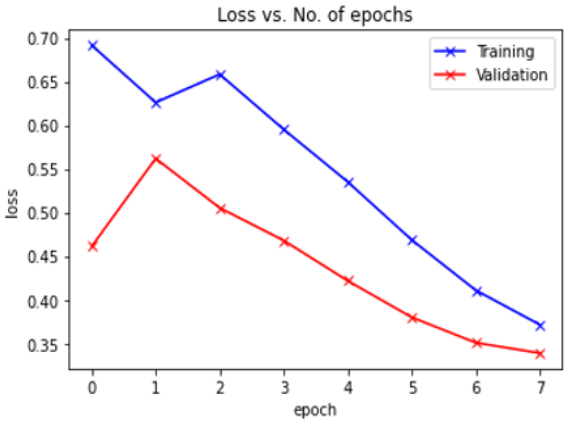

Figure 8. AlexNet Loss

From the learning curve, we can observe that the learning rate is uniform as intended. The accuracy curve shows that the learning process is effective. However, the loss curves show that it may have an underfitting as the training losses are greater than the validation losses for all the 8 epochs. Training the model for a longer epoch may indeed solve this issue. However, for comparison we decided to run on 8 epoch in every testing model as time and resources are limited.

5.1.2. ResNet-9

For this model, the following parameters were chosen.

- Batch size: 400
- Number of epochs: 8
- Optimizer: Adam
- Max learning rate: 0.01
    - Weight decay: 1e-4

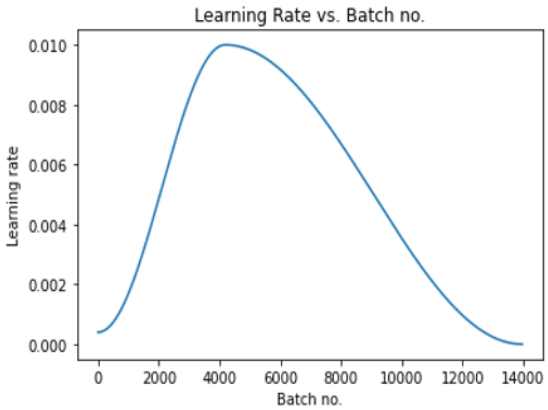

Figure 9. ResNet-9 Learning Rate

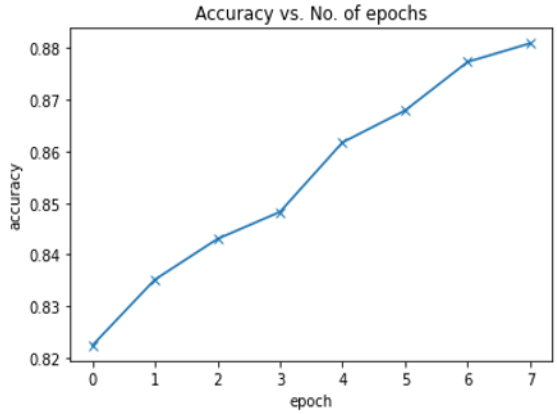

Figure 10. ResNet-9 Accuracy

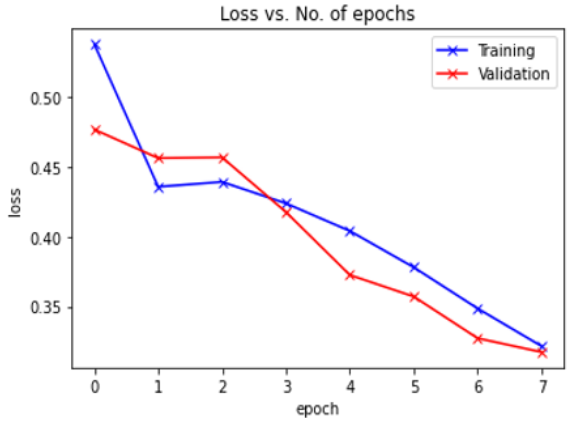

Figure 11. ResNet-9 Loss

From the learning curve, we can observe that the learning rate is uniform as intended. The accuracy curve shows that the learning process is effective. However, the loss curves show that it may have an underfitting as the training losses are greater than the validation for epoch 3 to 7. Nonetheless, it seems that at 7 the training loss will go under the validation loss.

5.1.3. ResNet-50

For this model, the following parameters were chosen.

- Batch size: 400
- Number of epochs: 8
- Optimizer: Adam
- Max learning rate: 0.01
- Weight decay: 1e-4

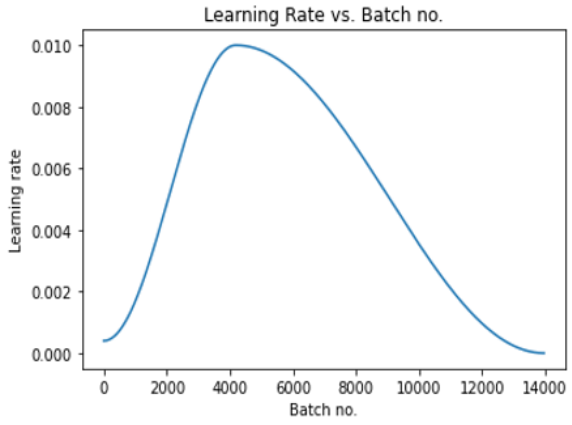

Figure 12. ResNet-50 Learning Rate

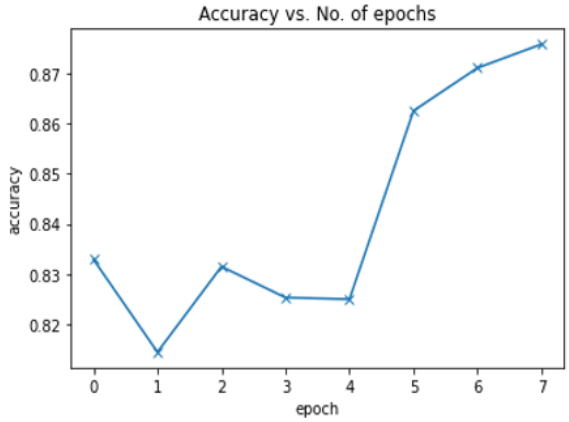

Figure 13. ResNet-50 Accuracy

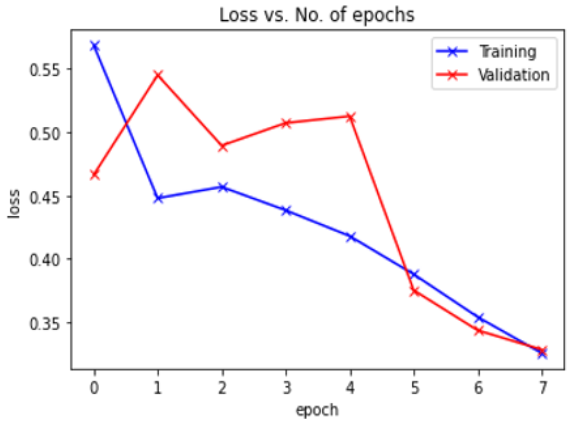

Figure 14. ResNet-50 Loss

From the learning curve, we can observe that the learning rate is uniform as intended. The accuracy curve shows that the learning process is effective. However, the loss curves show that it may have an underfitting as the training losses between epoch 5 and 7 are about the same as the validation losses.

5.1.4. VGG-11

For this model, the following parameters were chosen.

- Batch size: 100
- Number of epochs: 8
- Optimizer: Adam
- Max learning rate: 0.01

- Weight decay: 1e-4

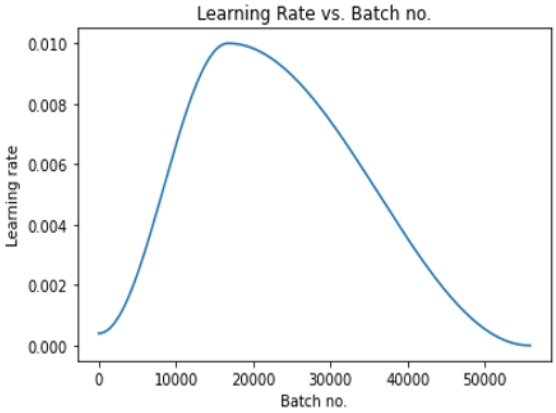

Figure 15. VGG-11 Learning Rate

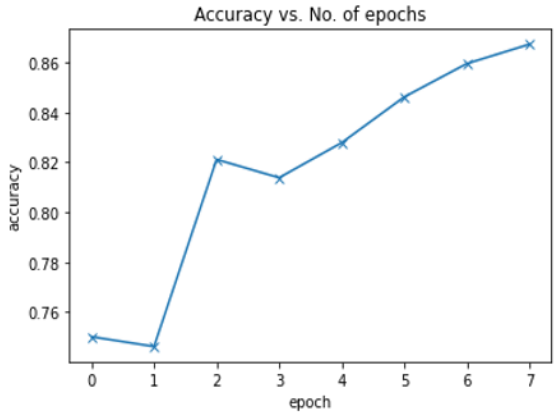

Figure 16. VGG-11 Accuracy

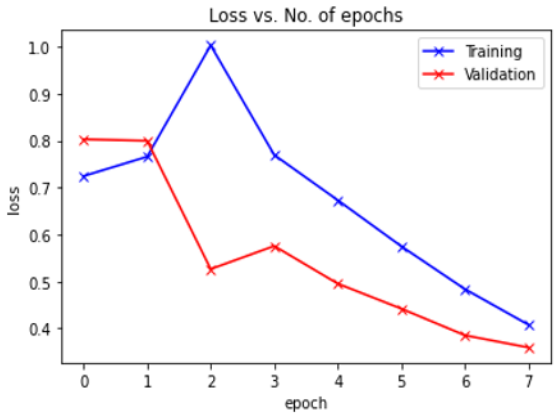

Figure 17. VGG-11 Loss

From the learning curve, we can observe that the learning rate is uniform as intended. The accuracy curve shows that the learning process is effective. However, the loss curves show that it may have an underfitting as the training losses are greater than the validation losses from epoch 1 to 7. Nonetheless, it seems that if trained for a longer epoch it may solve this issue.

5.1.5. VGG-16

For this model, the following parameters were chosen.

- Batch size: 100
- Number of epochs: 8
- Optimizer: Adam
- Max learning rate: 0.01
- Weight decay: 1e-4

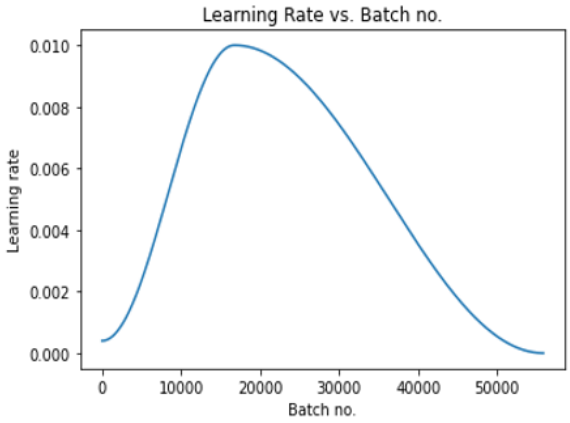

Figure 18. VGG-16 Learning Rate

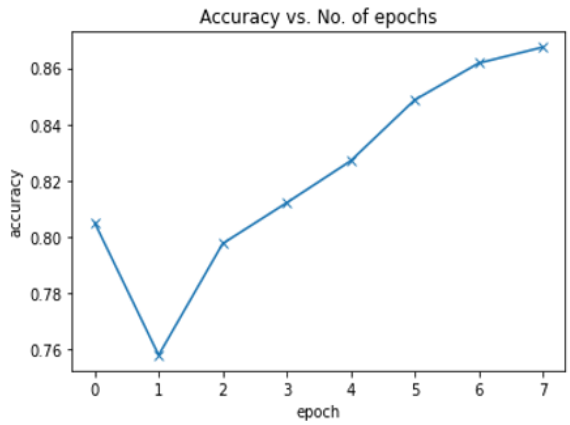

Figure 19. VGG-16 Accuracy

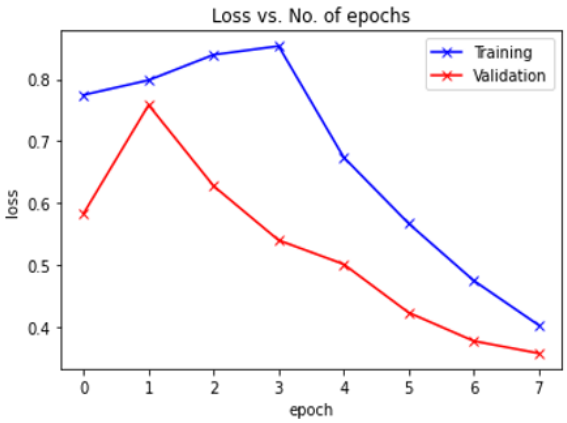

Figure 20. VGG-16 Loss

From the learning curve, we can observe that the learning rate is uniform as intended. The accuracy curve shows that the learning process is effective. However, the loss curves show that it may have an underfitting as the training losses are greater than the validation losses for all the 8 epochs. Nonetheless, training the model for a longer epoch may solve this issue.

5.1.6 ResNet-9 (Tuned)

After carefully analyzing the previous models, we decided to choose ResNet-9 to tune. The reason for this is because it is faster than VGG and AlexNet, it is less complex than ResNet-50, and more resources can be allocated. To tune this model, the following parameters were assigned.

- Epochs: 30
- Max learning rate: 0.02
- Gradient clip :0.2
- Weight decay: 1e-8
- Batch size: 64
- Optimizer : Adam

Additionally, data augmentation was performed in order to reduce the overfitting and bias. The type of data augmentation performed was a random rotation from -5 to 5 degrees and a perspective distortion with 0.05 degree of distortion.

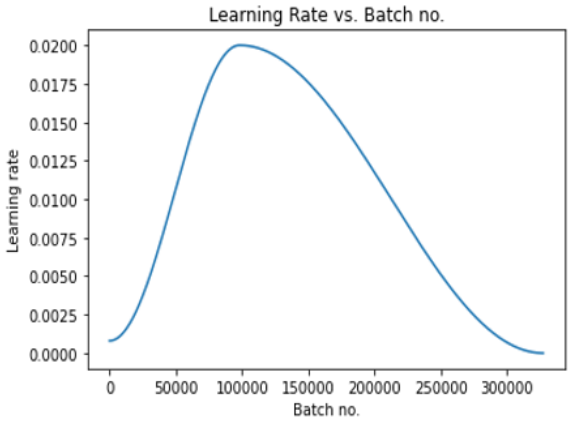

Figure 21. ResNet-9(Tuned) Learning Rate

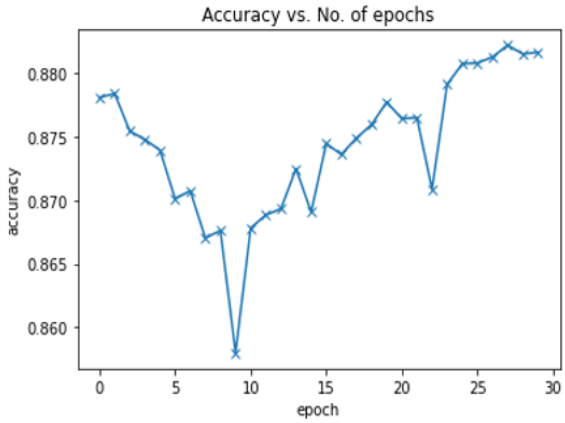

Figure 22. ResNet-9(Tuned) Accuracy

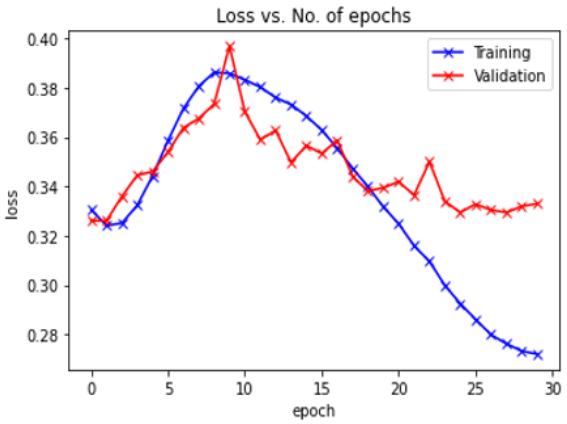

Figure 23. ResNet-9(Tuned) Loss

From the learning curve, we can observe that the learning rate is uniform as intended. The accuracy curve shows that the learning process is effective. The loss curves show that it may be in the optimal performance of the model as the validation losses remain the same and the training losses begin to increase after reaching a bottom.

5.2. Summary

This section contains a summary of results obtained in the previous subsection. The information contained in this summary are the accuracy of the model, the validation loss, and the training time.

Net	Accuracy	Loss	Time (hour)
AlexNet	0.873	0.339	1
ResNet9	0.881	0.317	0.5
ResNet50	0.876	0.329	0.5
VGG11	0.867	0.360	8
VGG16	0.867	0.357	17
ResNet9 (tuned)	0.882	0.333	3.5

6. Implementation

To implement our model into a real world scenario, we developed a web application. In this web application, we used python to implement the server of our application through Flask. Then, we created a python file that loads the model and calculates three confidence scores and three predicted values for that certain user input. The confidence level is calculated with a softmax function.Then, we developed the drawing board with javascript, and implemented it in the HTML file. Styling was done in a CSS format. The final result of this implementation can be seen in fig 24.

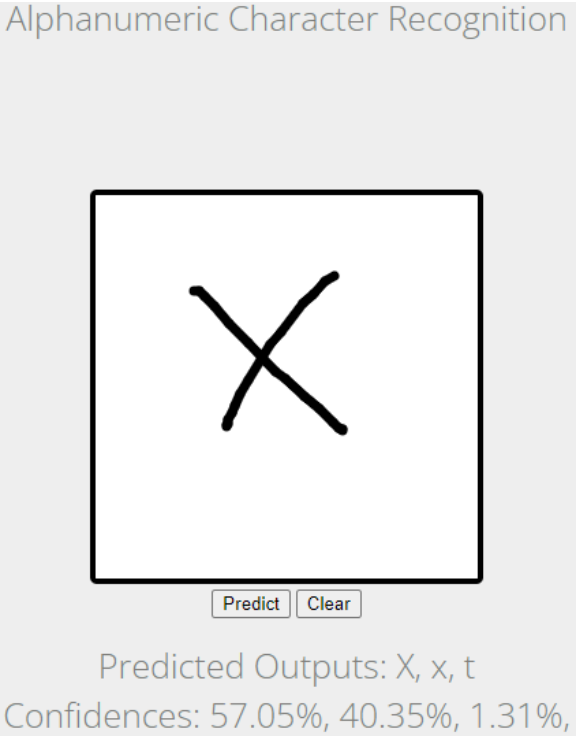

Figure 24.Web Application of the Recognition System

As depicted in the previous figure, the user wrote the letter X. When the predict button is pressed, the program analyzes the image and gives 3 outputs with confidence level in order. In this case, the predicted output is “X” with a confidence level of 57.05 %. A second possible solution of “x” with 40.35% confidence is predicted as well. Additionally, a third prediction of “t” with a confidence of 1.31% is also predicted.

7. Conclusion

In our work, we investigated three different models AlexNet, ResNet and VGG for handwriting recognition using EMNIST datasets. Our solution used a machine learning algorithm and convolution neural network based on the discussed model. There are upside and downside to each model in comparison. AlexNet seems to perform below expectation, with validation loss value lower than training value, 87.3% accuracy and training time of around 1 hour. ResNet-9 (a modified version of ResNet-18) and ResNet-50 has the best score with decent training time (30 minutes), 88.1% accuracy, and expected loss graph with only 0.317. A hypertuned option for ResNet-9 gives a slightly better result of 88.2% accuracy but with a trade off of tripling the amount of training time. VGG-11 and VGG-16 seem to perform the worst with only 86.7% accuracy, a high loss value of 0.360 and took significantly long time to train: 8 hours for VGG-11 and 17 hours for VGG-16. We can safely say that the ResNet model gave the best accuracy, along with reasonable training time. This makes the ResNet model suitable for a real-time application based on recognizing any handwritten text. Our research uses 8 epochs in training to overcome luck-based prediction as well as provide enough training due to limited setting and dataset. Later, we discuss that increasing the number of epochs does not improve the accuracy and even introduce overfitting from a small dataset.

8. Future work

Compared to other results discussed in section 2, we obviously need to work on expanding our model, specifically increasing the batchsize, epochs to exhaust the possibility of training to improve the accuracy. There are still several indistinctive cases that we need to work on, for example: our program yet to tell the difference between a “i”, “I”, “l” and “1” with high confidence. Due to the borrowed Google Colab setup, the GPU

capability and storage size came limited, thus lengthening the training time. With a more powerful computer setup, we will be able to explore many more options as well as speeding up the training. Once the training is up to high accuracy (more than 95%), we will expand the research on recognizing words (sequence of characters) and sentences (sequence of words). The future work should aim for the ability to read a whole letter of human writing. Last but not least, to extend the scope of the project, handwriting recognition needs to combine with natural language processing so that the machine can fully understand the context of the paragraph. In the end, we are confident that handwriting recognition will be a solved problem in the near future.

Reference

[1] Deng, L. (2012). *The mnist database of handwritten digit images for machine learning research*. IEEE Signal Processing Magazine, 29(6), 141–142.

[2] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). *EMNIST: an extension of MNIST to handwritten letters*. Retrieved from http://arxiv.org/abs/1702.05373.

[3] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). *ImageNet classification with deep convolutional neural networks*. Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386. ISSN 0001-0782. S2CID 195908774.

[4] Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2016). *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE. pp. 770–778. arXiv:1512.03385. doi:10.1109/CVPR.2016.90. ISBN 978-1-4673-8851-1.

[5] K. Simonyan, A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 10 April 2015. doi:10.48550/arXiv.1409.1556.

[6] S. S. Rosyda, T. W. Purboyo. *A Review of Various Handwriting Recognition Method*s. 2018. Volume 13, Number 2, pp. 1155-1164.

[7] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, B. Yoon. *Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)*. 12 June 2020.

[8] A. Hossain, M. Ali. *Recognition of Handwritten Digit using Convolutional Neural Network (CNN)*. 2019. Volume 19 Issue 2 Version 1.0.

[9] M. D. Garris, S. Janet. *NIST FORM-BASED HANDPRINT RECOGNITION SYSTEM*. 15 March 1994.

[10] U. V. Marti, H. Bunke. *The IAM-database: an English sentence database for offline handwriting recognition*. 2022. pp 39-46.

[11] Grosicki, Emmanuele, et Haikal El-Abed. 2011. *ICDAR 2011 - French*

Handwriting Recognition Competition. In, 145963. IEEE.

[12] Q. Chen, S. Zhou. *Harbin Institute of Technology Opening Recognition Corpus for Chinese Characters (HIT-OR3C)*. 30 April 2020.

[13] shipra_saxena. *Introduction to The Architecture of Alexnet.* https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

[14*] yasho_191. How to code your ResNet from scratch in Tensorflow*? https://www.analyticsvidhya.com/blog/2021/08/how-to-code-your-resnet-from-scratch-in-tensorflow/

[15] tanaya17. *Transfer Learning using VGG16 in Pytorch.* https://www.analyticsvidhya.com/blog/2021/06/transfer-learning-using-vgg16-in-pytorch/