American University of Armenia

Zaven & Sonia Akian College of Science and Engineering

CS 251/340, Machine Learning Course Project

Armenian Handwritten Letters Recognition

Team:
Barsegh Atanyan
Myasnik Khachkalyan
Petros Mkheyan

Spring, 2021

Abstract

We built a model which recognizes Armenian handwritten letters. There are several models built for this kind of problem for other languages. We have designed a new model which is suitable for Armenian language, considering the already existing models for other languages. Since we are concentrating only on Armenian letters, we expect our model to work faster and more accurately. In this paper we show how we have reached to our final model. We also show the experiments performed using our model and the results of those experiments.

Keywords: handwritten, armenian, letters, NN, CNN, sparse categorical crossentropy

Contents

List of Figures

Chapter 1

Introduction

1.1 The Structure of the Project Paper

In this chapter the problem setting and motivation are described. Later the dataset is presented. Next, the model design and training is discussed followed by the corresponding results.

1.2 Problem Setting, Project Motivation and Description

The goal of this project is to build an application that can help to recognize handwritten Armenian letters. To be more precise, given the image of some letter as an input, it will predict the most probable corresponding Armenian letter regardless of it being lowercase or uppercase. Overall there are 39 Armenian letters. Thus, the number of possible outcomes is 78.
One of the most famous Machine Learning problems is to classify handwritten digits. Also, various models have been developed to classify handwritten letters of different languages. Hence, it became a challenging task for us to build such model for our native language. Furthermore, we strongly believe that solving the problem of Armenian handwritten letter recognition may become a crucial impetus in the field of digitizing Armenian documents in both public and private sectors. Recognizing the individual letters is the basis of recognizing words and sentences, and this project can serve as a great starting point for having an application with the ability of recognizing and digitizing texts in many spheres.
We were not able to find any model that is designed and trained for Armenian letters. Though, Mashtots Collection has collected the data and plans to build a model. Thus, when there is a complete model we might compare that with the

model discussed in this paper. To continue, we have looked through several models for different languages such as English, Russian, Bengali [Roy20].
To achieve our goal various models have been trained and evaluated to get better results. In the following sections those models will be discussed in more detail and the corresponding results will be presented.

Chapter 2

Data and Preprocessing, Performance Measurement

2.1 Dataset

The dataset was obtained from Mashtots Collection [Mas]. In total there are 17075 64x64px grey scale images grouped according to their labels. The data was mostly collected from school students. Moreover, considering that the number of school-aged girls and boys is balanced there is no bias towards the gender.
In Figure 2.1 the barplot of the dataset is represented. On average, the dataset contains 219 unique samples of each letter.

2.2 Feature selection and Preprocessing

As the images were already resized to be 64x64 images, we were able to skip the cutting and resizing step. For better performance and ease of use, we have read these images as matrices, flattened and stored the data in a CSV file. We have also created a separate CSV file for the labels. Furthermore, the data was split into three parts - train, validation and test sets.

While experimenting with the dataset, we have been trying to reduce the dimensions of each image. However, the results were not satisfactory. We believe the main reasons are that the actual sizes of the letters are quite different(consider having both uppercase and lowercase letters) and the actual written letter was not always in the center. Thus, simple cropping was not optimal. Also, considering the curvatures of Armenian letters the dimensionality reduction of images resulted in huge information loss.

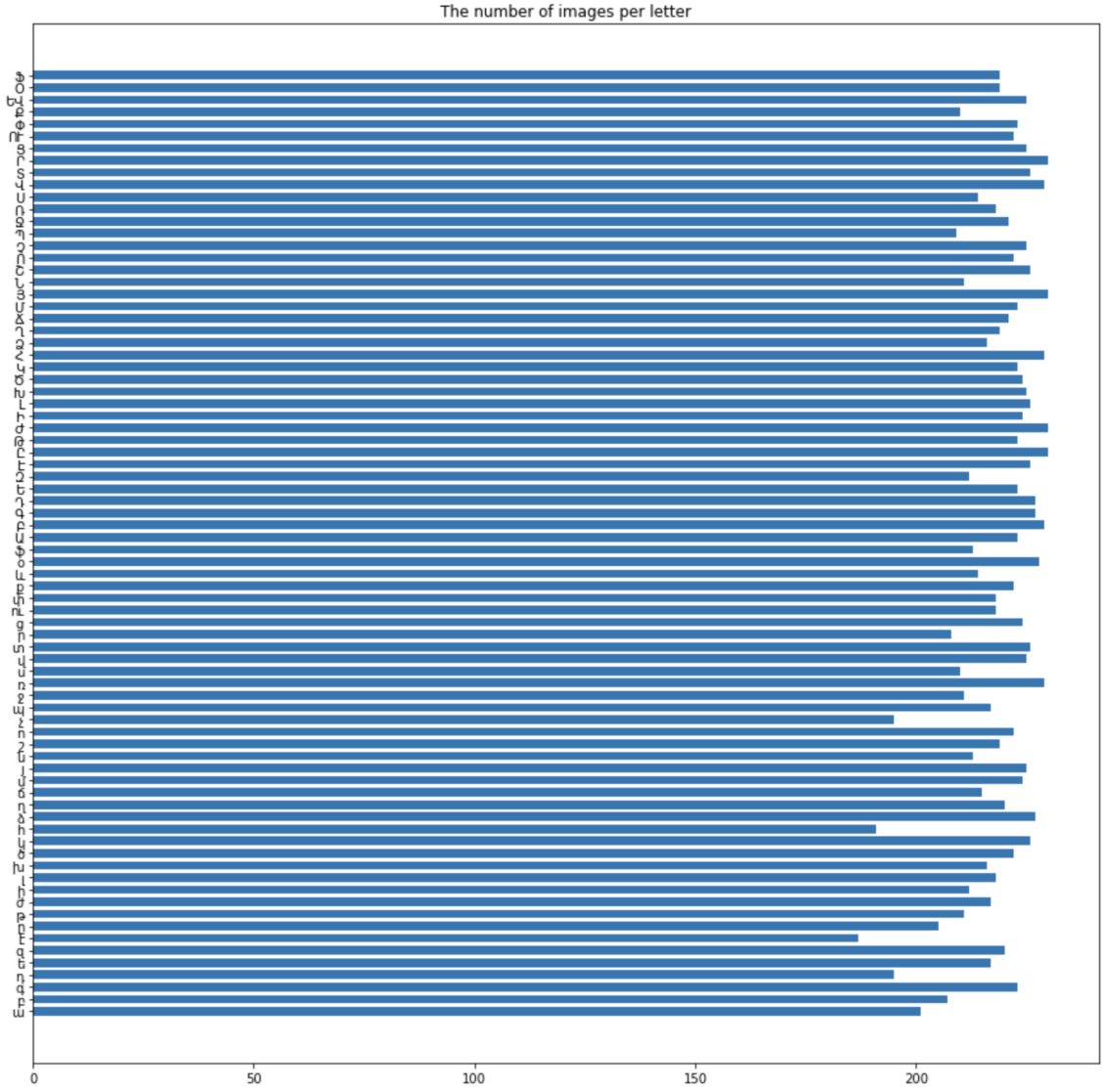

Figure 2.1: BarPlot of the number of images per letter

2.3 Performance Measurement

The research showed that in such image classification problems Categorical Cross Entropy loss function suits very well. However, in our case the labels are not one-hot vectors, so Sparse Categorical Cross Entropy is used which is the combination of SoftMax and Cross Entropy losses. Correspondingly, we have used Sparse Categorical Accuracy as a performance measurement metric.

	Dataset	N of images	% of images
0	Train Data	10245	60%
1	Validation Data	3415	20%
2	Test Data	3415	20%

Figure 2.2: The structure of data

Chapter 3

Algorithms and Models

3.1 Algorithms and Models

Before building our own model for Armenian letters, we have considered models built for other languages. We have researched those models and came up with our own Sequential model for Armenian letters. We have started with one Convolutional layer followed by a MaxPooling and a Dense layer. For the final output, we have added another Dense layer with Softmax activation. In all the other Convolutional and Dense layers we use RELU (Rectified Linear Unit) activation. We were using "accuracy" instead of "sparse categorical accuracy" during the early stages of constructing our model. At this step our model reached 80% accuracy.
Afterwards, we have added two more pairs of Convolutional and MaxPooling layers on the top of our previous model, before the Dense layers. At this step our model reached 84% accuracy.
In the next step, we have added another pair of Convolutional and MaxPooling layers, as well as padding for all the Convolutional layers. We have also added Dropout layers after each pair of those layers to avoid overfitting. In this model we started using "sparse categorical accuracy" and reached around 86%.
At this step we decided to make major changes in our model instead of adding additional layers. We applied VGG16 architecture on our model but it didn't perform well enough on our dataset. We combined the ideas of VGG16 with our model by adding and removing some Convolutional layers. We have also removed padding from the last two Convolutional layers. We reached 87% "sparse categorical accuracy."
We have trained all the above mentioned models in 10 epochs. We thought that we are underfitting so we decided to increase the number of epochs to 20. However, we found out that this caused a major overfitting. We decided to modify the

optimizer. We have used Adam instead of RMSProp, which didn't worked very well. Then we saved the model with 10 epochs and restarted the training using lower learning rate with RMSProp optimizer. Our final model reached around 90% "sparse categorical accuracy." [Sch18] [AWS] [Mat21] [Hcr]

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_42 (Conv2D)           (None, 64, 64, 64)        640
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 32, 32, 64)        0
_________________________________________________________________
dropout_36 (Dropout)         (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 16, 16, 128)       0
_________________________________________________________________
dropout_37 (Dropout)         (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 16, 16, 256)       295168
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 8, 8, 256)         0
_________________________________________________________________
dropout_38 (Dropout)         (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 6, 6, 512)         1180160
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 2, 2, 512)         0
_________________________________________________________________
flatten_9 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_18 (Dense)             (None, 1024)              2098176
_________________________________________________________________
dropout_39 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_19 (Dense)             (None, 78)                79950
=================================================================
Total params: 6,087,758
Trainable params: 6,087,758
Non-trainable params: 0
```

Figure 3.1: Final Model Summary

Chapter 4

Results and Conclusion

4.1 Experiments and Results

Throughout the construction of our final model, we were not saving the initial ones or drawing graphs for them, as we considered them too raw at that stage. Later, when we have obtained accuracies over 86% we started saving our models, which you can find in a separate drive referenced in the next chapter. We have also extracted the most common mistakes of our best model, which performs with around 90% accuracy. One might think that this is still low accuracy compared to other models, but we have found out that these letters were quite similar, and sometimes even humans can confuse them. Hence, we can claim that the accuracy of our model could have performed better had there not been the difficulty of Armenian letters. Below we are presenting several statistics about our best model, the confusion matrix and the most common mistakes that it made.

4.2 Conclusion

We have used Convolutional Neural Networks, which performed quite well for our model. We have found out that the number of epochs and Dropout layers are very important for preventing overfitting. For our project 90% accuracy is quite high, also considering the similarities among the mistakenly predicted letters. Surely, there is a room for improvement, and further tuning of layers and hyperparameters will result to a better result.

4.3 Further Work that can be done etc

Our model is able to recognize individual Armenian letters with quite high accuracy. We will try to further increase the accuracy of our model to 95%. The next step can be the implementation of recognizing words and sentences using Recurrent Neural Networks.

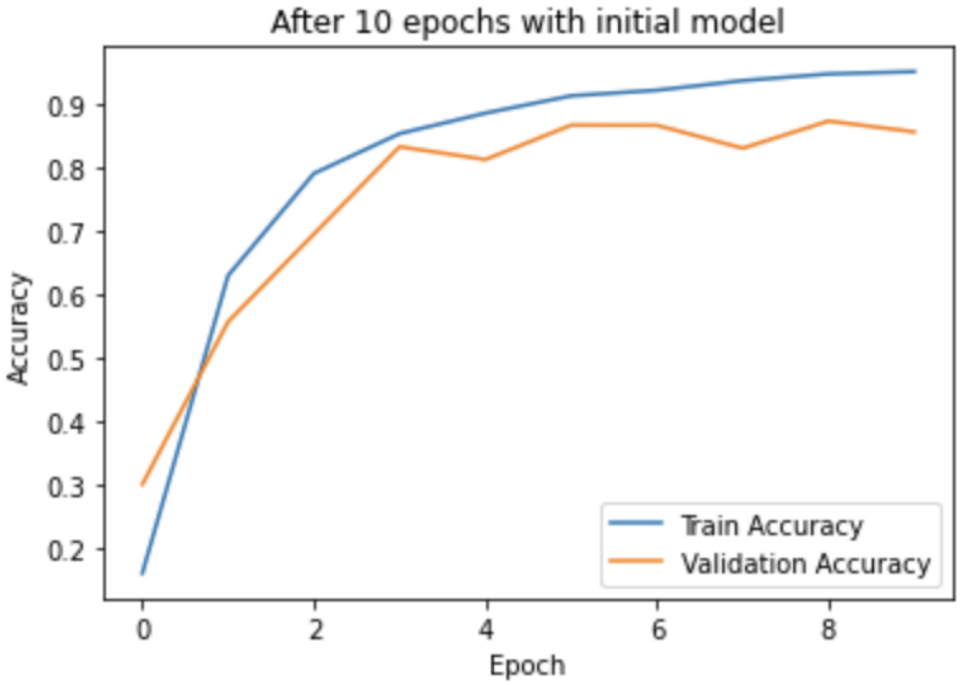

Figure 4.1: Accuracy after 10 epochs

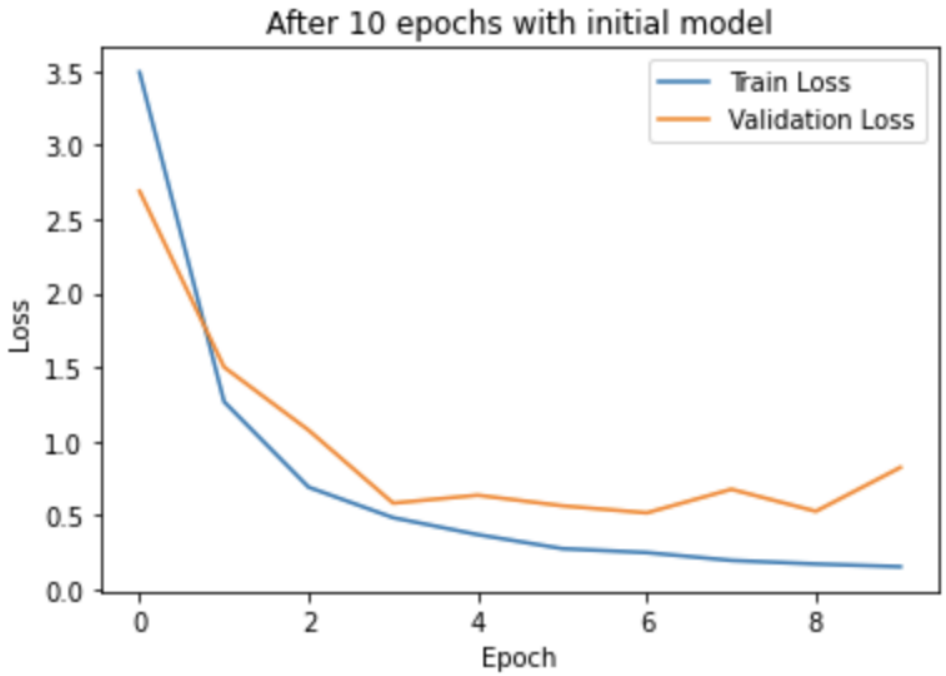

Figure 4.2: Loss after 10 epochs

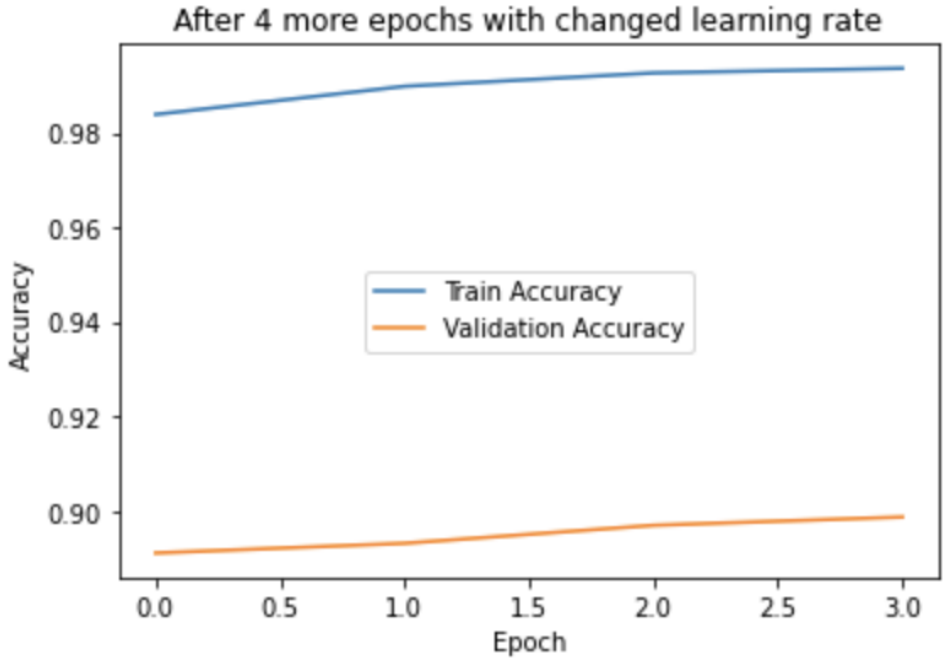

Figure 4.3: Accuracy after 4 more epochs

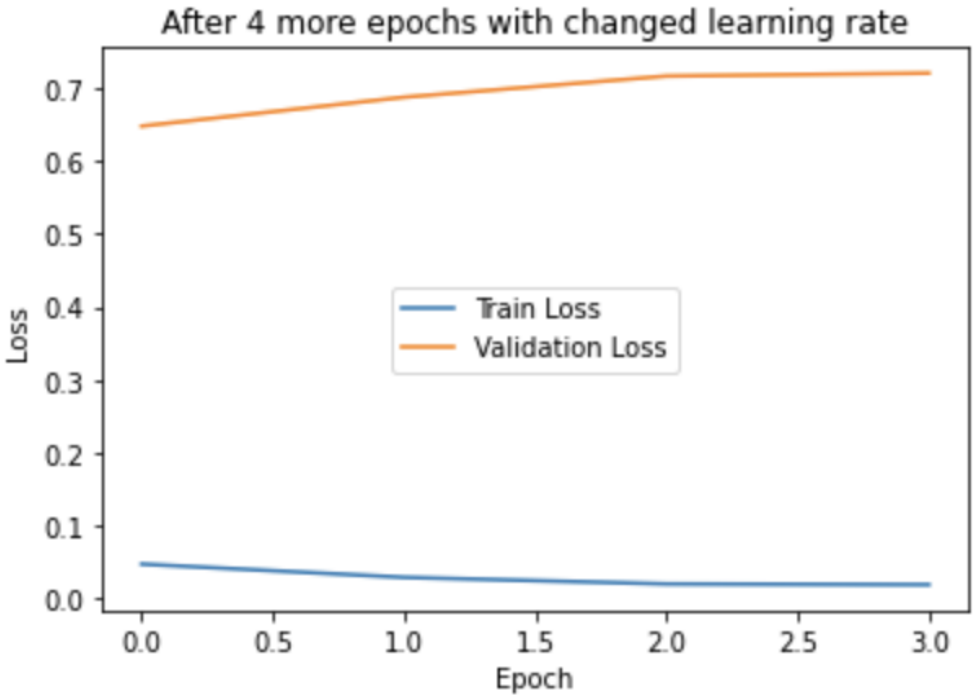

Figure 4.4: Loss after 4 more epochs

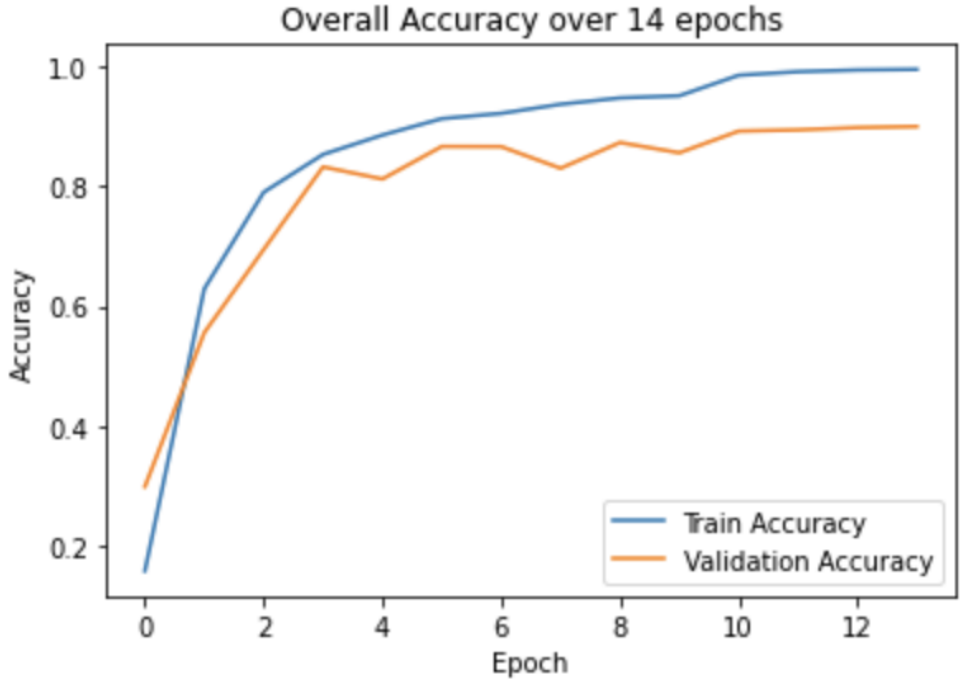

Figure 4.5: Accuracy after 14 epochs

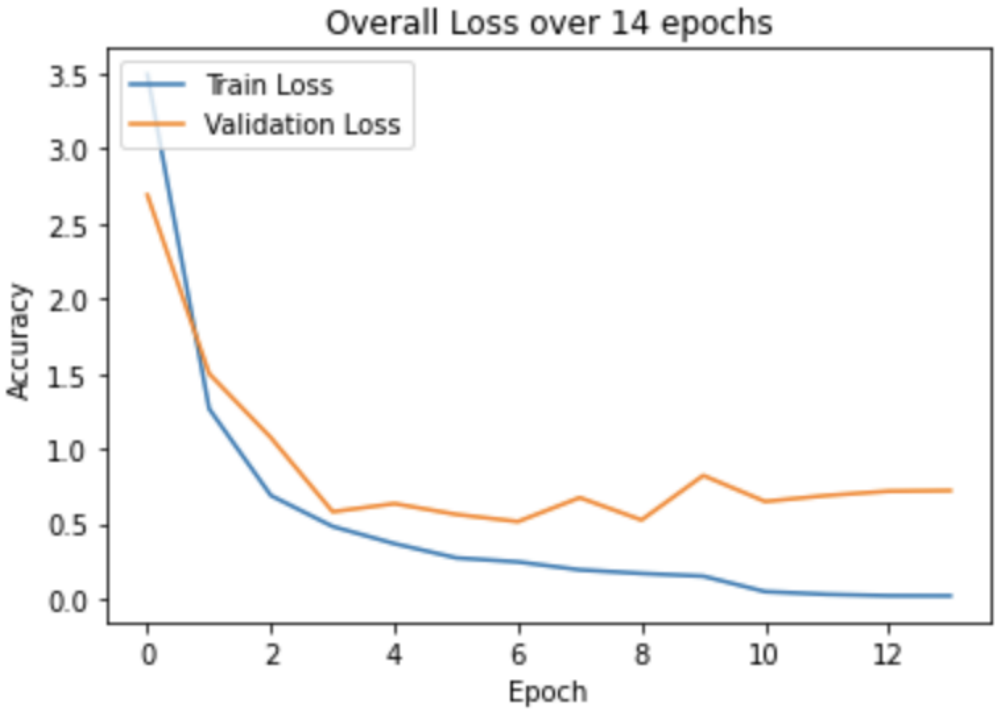

Figure 4.6: Loss after 14 epochs

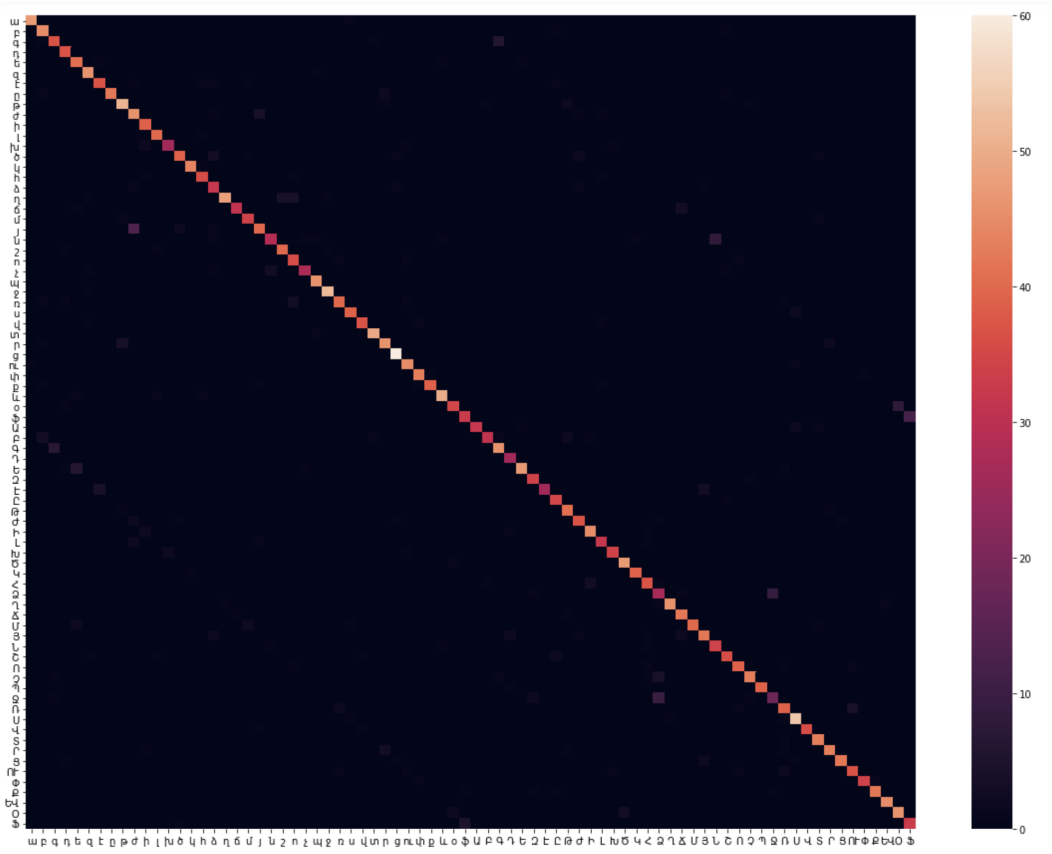

Figure 4.7: Confusion Matrix

Actual Letter	Predicted Letter	N
ჯ	ჯ	12
ს	ძ	11
ჯ	ჯ	11
ფ	ფ	10
ძ	ს	8
ი	ი	8
ყ	ყ	7
ղ	շ	6
ყ	ყ	6
ფ	ფ	6
ჱ	ჱ	5

Figure 4.8: Common Mistakes

Chapter 5

Appendices

5.1 Algorithm Implementation in Python

Github Repo: https://github.com/MyasnikKhachkalyan/ML-Project
Dataset and models will be shared separately, in a private drive.
Prerequisites:
Python: 3.8 or above, TensorFlow: 2.0 or above, SKLearn: Latest, OpenCV2: Latest, Numpy: Latest, Pandas: Latest, Seaborn: Latest

Bibliography

[Sch18] Harald Scheidl. *Build a Handwritten Text Recognition System using TensorFlow*. 2018.

[Roy20] Akash Roy. *AKHCRNet: Bengali handwritten character recognition using deep learning*. 2020.

[Mat21] Anil Chandra Naidu Matcha. *How to easily do Handwriting Recognition using Deep Learning*. 2021.

[AWS] AWS. *Neural Network Essentials: Convolution and Max Pooling*.

[Hcr] *Handwritten Character Recognition with Neural Network*.

[Mas] *Mashtots Collection*.