COSC 4368 Final Project

Nibras Sultan **Damian Andaya** **Rehman Essani**

Abstract

Many existing records are handwritten which can vary between texts and are also susceptible to damage. Our goal is to create a machine learning model which can accurately read these texts. To do so, we began training our model with the EMNIST(ByClass) data set which contains more than 800,000 unbalanced, handwritten samples. After sharpening the images, we created a Convolutional Neural Network to interpret the vast amount of samples. After many attempts, the final model was able to properly classify samples with an accuracy of 86%.

1 Introduction

Although society's reliance on technology is continuously increasing, many documents that are needed to be processed are handwritten. Handwriting is potentially a disruptive characteristic causing confusion due to a person's poor handwriting or documents that have faded due to aging. The goal of this project is to reduce the confusion that handwriting brings by using AI to recognize handwriting and what is written. In this project, we will specifically train a Convolutional Neural Network to predict what character or number the provided images from the EMNIST data set shows. We will compare our custom model to two other well known models and compare the accuracy of both.

2 Background

Handwriting recognition is a subject that has had extensive exploration within the field of artificial intelligence. This research has led to a wide variety of applications, including digitization, text processing, and document analysis. There have been numerous approaches to address the obstacles associated with transcribing handwritten text using machine learning.

2.1 Previous Approaches

Before the application of machine learning, the traditional methods of handwriting recognition involved the extraction of features such as slant, and curvature of characters. Support Vector Machines (SVMs) and Hidden Markov Models (HMMs) were frequently utilized for classification problems. However, these approaches were limited by their implementation of manually engineered characteristics and were unable to effectively deal with varied handwriting styles. With the rise of neural networks, there was a significant improvement in the research. Convolutional Neural Networks allowed models to learn hierarchical features instantly and retain contextual information in character sequences. This transition indicated a pivotal moment, allowing for the development of more sophisticated technology.

2.2 Current Approaches

Modern handwriting recognition employs complex models that have bolstered the field to remarkable levels of efficiency and accuracy. Convolutional Neural Networks have been a staple in image-based problems with their ability to capture intricate details in handwritten characters. Our approach utilized this method. Other models exist, such as Transformer architectures. Inspired by their

outstanding results in natural language processing, this approach acquired recognition in recording contextual dependencies within sequences of characters. The Transformer model's mechanisms have the capability for context-aware recognition, which makes it an important player in the current field. This is specifically important because it allows the model to comprehend not only single characters but also the relationships and context within a given sequence. Long Short-Term Memory Networks, a kind of recurrent neural network have been utilized because of their ability to record temporal dependencies in sequential data. There are also ensemble models, which merge predictions from several different models. Several comparative studies have been conducted to evaluate the efficacy of the various approaches. This research provides a comprehensive understanding of the pros and cons of different models under a multitude of conditions. The models are commonly evaluated on several aspects including recognition accuracy, computational efficiency, robustness to variations, scalability, and interpretability.

3 Data

The data that we used is the EMNIST dataset, which is an extention of the MNIST dataset. Out of the many subsets that this dataset had to offer, we chose to go with the "byclass" subset as this contains the most amount of data from the other subsets to train our model. This data contains an image and its corresponding class that the image represented. The images that were included are uppercase letters, lowercase letters, and digits 0 through 9. For the "byclass" subset specifically, there are a total of 814,255 images with a classification from one of 62 classes. The classes are numbered from 0 - 61 with the following representation: 0-9 for the corresponding digits, 10-35 representing uppercase letters in alphabetical order, and 36-61 representing the lowercase letters in alphabetical order as well. For preprocessing, we normalized our data and one-hot-encoded our labels to ensure that training our models would be consistent. Along with normalization of our data, we also did image sharpening to see if having a clearer image would improve the accuracy of our models. More of our data preprocessing can be found in section 5.1 of our report.

4 Methods

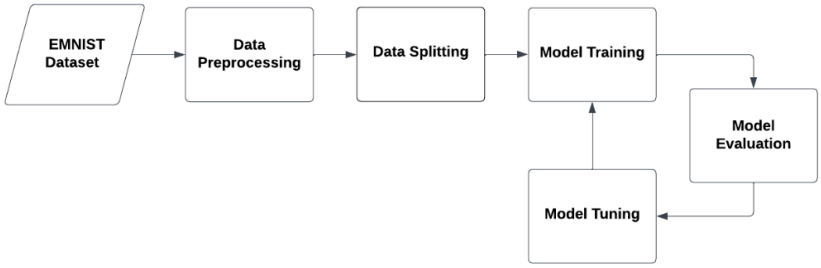


Figure 1: Workflow for our methods

To solve the problem of recognizing handwriting, we used a convolutional neural network to help solve our task. A convolutional neural network is a feedforward type of neural network that is useful for classification problems like our problem. This type of neural network takes an image and uses each layer to either detect patterns in a certain area in convolutional layers or reduce noise by focusing on the important details in pooling layers. For our approach to the problem, we first obtained our data and performed preprocessing to normalize our data. Our data is split into a training size of 86% of the data and a testing size of 14% of the data. To create the neural network, we utilized the TensorFlow and Keras libraries to create a neural network that consist of a combination of 2D convolutional layers, 2D max pooling layers and a final Dense layer with 'Leakyrelu'as our activation function. From there, we trained our model and validated our model over 15 epochs with an early stopping point if our validation loss did not see an improvement over 5 epochs. We then analyzed our results to then modify our model and then continued to train that model until we reach a result that was favorable.

5 Experiments

Once we had our initial model, we began testing the effectiveness of data preprocessing, model optimizers, and overfitting prevention methods. Individually, the alterations did not yield significant improvements, but the accumulated improvements were worthwhile in improving our CNN model and maximizing its potential. In order to fairly compare the results of our experimentation, each model was limited to 15 epochs and included Early-Stopping parameters which would also help in avoiding overfitting and extraordinarily long run-times.

5.1 Data Preprocessing

First, we took the standard steps of one-hot-encoding the labels and normalizing the corresponding images. This is generally good practice for any sort of model creation. Then, we attempted to improve the quality of the images. Because this data set contains images of size 28x28, the image clarity is hindered by rough edges and slight blooming. Using the UnSharp filter from SKimage smoothed the edges of the symbols and yielded these results:

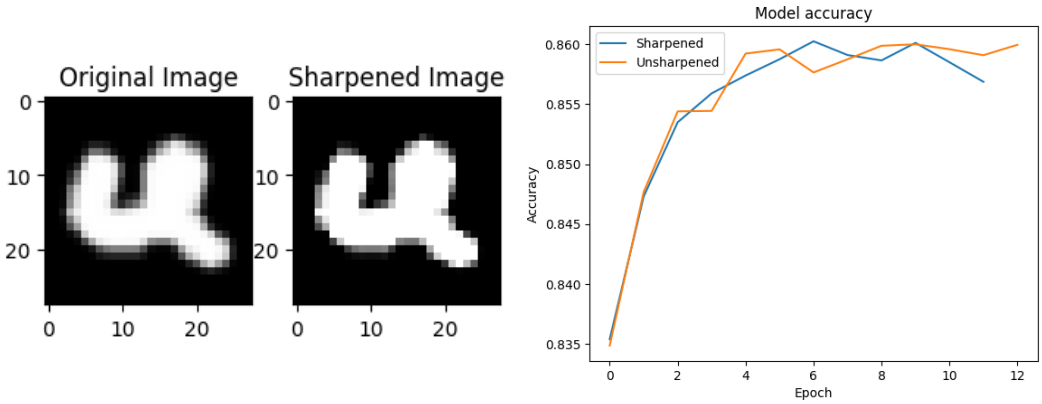


Figure 2: Unsharpened VS. Sharpened Results.

As we can see, sharpening the images did not improve the validation accuracy of our model, and further experimentation is required. Fortunately, an 86% validation accuracy is quite impressive already, and moving forward, we will be trying to close the gap between our model and pre-existing models.

5.2 Optimizers

Another way to improve our CNN model is to test different optimizers. For image classification, some popular optimizers are Stochastic Gradient Descent, Adam/Nadam, and Root Mean Square Propagation. Here are the results of our model using each:

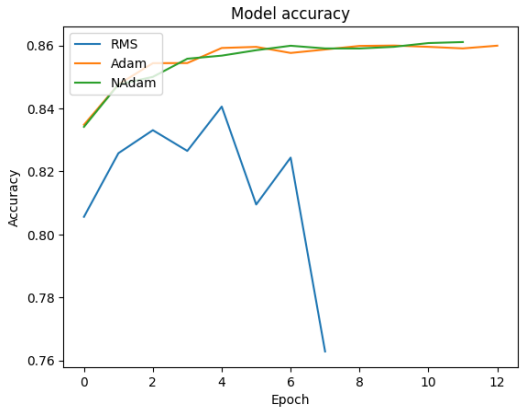


Figure 3: Model Accuracy with different optimizers. (SGD omitted because of a 5% accuracy)

As shown in the plot, the Adam and Nadam optimizers have the greatest performance for our use case. Because Nadam combines two optimizers together, it yields slightly better results within fewer epochs of training. Moving forward we will implement the Nadam optimizer as well as the unsharpened images from the previous section to begin merging our results together.

5.3 Overfitting Prevention

Currently our model performs as shown below:

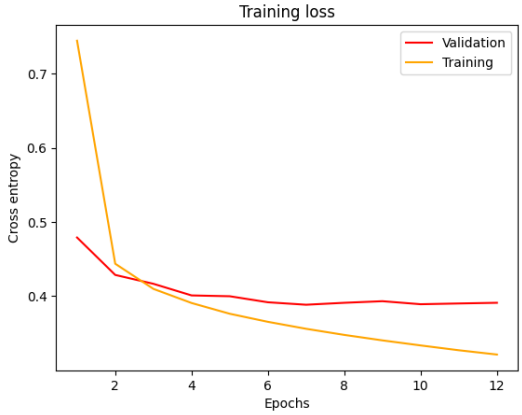


Figure 4: Training and Validation Loss of Current Model

This is a plot of our current model using Nadam and unsharpened images. We can see that the model quickly adapts to the training data and learns it very well. Unfortunately, this is causing some overfitting which is why the cross entropy of the validation process isn't quite as low. As of now, our model uses large amounts of data and is done with training when the epoch limit or early-stopping tells it terminate. To build on these efforts, we'll create more iterations of the model. One version of the model uses drop out layers which operate by randomly "dropping" a few of the neurons to prevent a dependence on any specific neurons during training. While testing, the neurons will be scaled by their drop rate, and as a result the model should be better generalized to new data. Another approach is to just create a simpler model with few layers and/or neurons. Rather than the model "forgetting" what it knows about the data, it will learn the training data less thoroughly on its initial pass. The results of this experimentation are as follows:

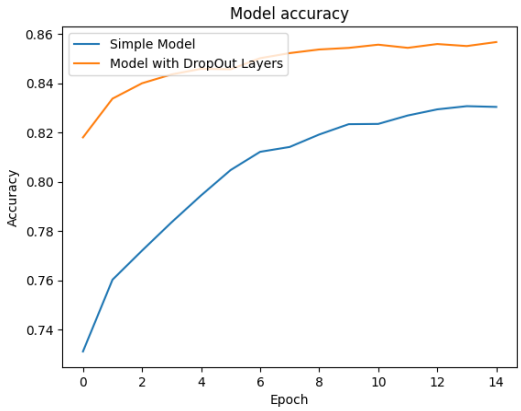
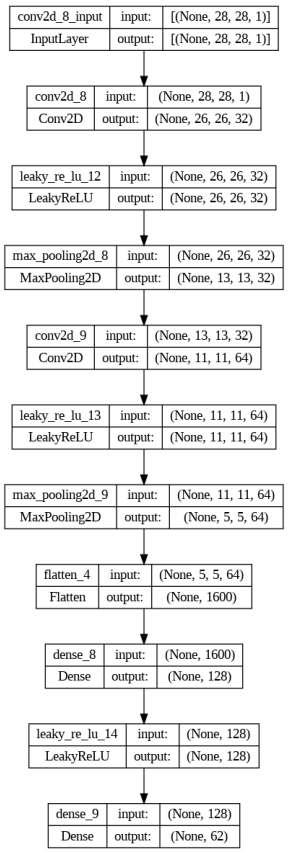


Figure 5: Accuracy of Model with Reduced Layers and Model with Dropout Layers

Although we have decent performance with the Dropout layers, both the model with Dropout layers and the simple model have poorer performance than our last models, so we'll proceed without implementing these means of prevention.

6 Conclusion

In conclusion, our Convolutional Neural Network was able to achieve an accuracy of 86% after all of our experimentation. This is the final architecture of our model:



Although models like VGG and ResNet can be slightly better (88-91% accuracy) we should keep in mind that those models are also relatively more complex when compared to CNN. However, these models are more efficient because of how advanced they are which could make them more viable for certain tasks. Through our experimentation we learned that CNN models can not be pushed much further in this case and that it is better to spend the time creating more intricate models for the EMNIST(ByClass) data set because of how much more potential there is. We may be able to implement our model to interpret individually written letters or numbers as we've seen in Android phones and Apple watches which allow users to hand write and convert to text. Another idea would be to start collecting samples of writing in other languages as well as directional data which illustrates how symbols are actually written. All of these options could further the usability of our model and expand on what we've learned thus far.

7 Referenced Material

7.1 Analytics Vidhya, Jitendra Sharma

https://www.analyticsvidhya.com/blog/2021/06/building-a-convolutional-neural-network-using-tensorflow-keras/

7.2 Kaggle, Amy Jang

https://www.kaggle.com/code/amyjang/tensorflow-mnist-cnn-tutorial/notebo

7.3 Towards Data Science, Richmond Alake

https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342

7.4 Springer Link, Christopher Wick, Jochen Zollner, Tobias Gruning

https://link.springer.com/chapter/10.1007/978-3-030-86334-0_8

7.5 World Scientific, Muna Khayyat, Nicola Nobile

https://www.worldscientific.com/doi/abs/10.1142/9789811203527_0004#:~:text=In%201933%2C%20Paul%20W.,proposed%20with%20statistical%20meth%2D%20ods.

7.6 EMNIST: an extension of MNIST to handwritten letters

Cohen, Gregory, et al. "EMNIST: Extending mnist to handwritten letters." 2017 International Joint Conference on Neural Networks (IJCNN), 2017, https://doi.org/10.1109/ijcnn.2017.7966217.