Handwritten Letter Recognizer Model Using CNN

By: Paul Parker, Vishal Erninti, Matthew O'Donnell, Samuel Priddy

April 30, 2024

1 Introduction

In this project, we aim to create a Convolutional Neural Network (CNN) model capable of accurately recognizing handwritten alphabet letters. This idea particularly interested our group because it caught our attention that the iPad application, Scribble, attempts to translate a user's written text into "typed" text, yet is sometimes inaccurate in doing so. The following images are samples of handwritten letters from the dataset:

2 Motivation

The motivation of our project comes from the rise in demand for automated systems capable of interpreting handwritten text. Despite the rise of technology and the increasing digitalization of the world, many people and businesses still use and prefer handwritten documents: personal notes, contracts, official documents, medical records are some of the contexts that handwritten text is still prevalent in. If powerful models are created, then signature verification, students' note taking, doctors' notes, etc. can be easily interpreted by any reader. With a handwriting recognition system, these contexts can be converted from handwriting to electronic text, and offers many advantages. It facilitates the use of electronic storage and lessens the need for paper storage which, in turn, reduces the risks that come with paper storage. Electronic documents can also be sorted, organized, and searched faster and easier. Individuals can quickly search and find any information needed, reducing the time-consuming process of searching through pills of documents and manually entering information yourself. Traditional handwritten letter recognition often relies on machine learning algorithms, however these algorithms often struggle to generalize to diverse writing styles, limiting the use of these recognizers in real-world applications. To

address this issue, we utilize Convolutional Neural Networks, CNNs, to develop an effective solution for handwritten letter recognition.

3 Related Work:

Research on Handwritten Alphabet Recognition Systems Based on Extreme Learning Machine is a study conducted by a group in the Hubei key Laboratory of Advanced Control and Intelligent Automation for Complex Systems located in Wuhan, China. Existing handwriting recognition models were traditionally done using the Support Vector Machine (SVM) and Back Propagation (BP) algorithms, however, both of these algorithms came with issues; BP requires a long training time while SVM has a slow calculation speed but requires a large amount of calculations to accurately recognize letters. To address these issues this team instead used the Extreme Learning Machine (ELM) algorithm. The ELM adjustment process only needs to be calculated once and the entire execution process is done by the machine self-help, so ELM can quickly train and independently calculate the optimal solution. ELM's fast learning speed and powerful generalization ability is perfect for handwriting recognition. Using this system, this group was able to accurately recognize handwritten letters at a 98.82% rate and can do this in tens of milliseconds.

Handwritten Letter Recognition By Using Tensorflow with Deep Learning Machine was the Master's Thesis of Mai Ngoc Dang that uses Tensorflow along with CNN to display its ability to recognize handwritten letters. Dang uses Tensorflow to train the dataset, first lowercase letters, then uppercase, then Vietnamese. Dang's model was able to achieve a recognition rate of 100% in some cases.

4 Approach:

4.1 Data Collection:

We found a dataset on Kaggle that contains 26 folders, each containing images of the folder's corresponding letter (one folder per letter). The dataset is in a CSV format, each row containing the pixel information of each image. The images are formatted as 2828 pixels, each character fitting into a centered 2020 pixel box, and the images are in grayscale. The creator of the dataset, Sachin Patel, compiled the dataset from images taken from NIST and NMIST.

4.2 Data Preprocessing:

The first step we took was to clean up and preprocess the data. The dataset did not contain any missing values. However, we found that the data had significantly higher counts of the letters 'O' and 'S,' while having less than 1200 counts of 'F' and 'I.' We trimmed the data down to at most 3000 per letter by randomly selecting a sample of each letter's data points. The resulting dataset

was more evenly distributed between letters. The smaller file size was also less demanding on the limited hardware we had to work with.

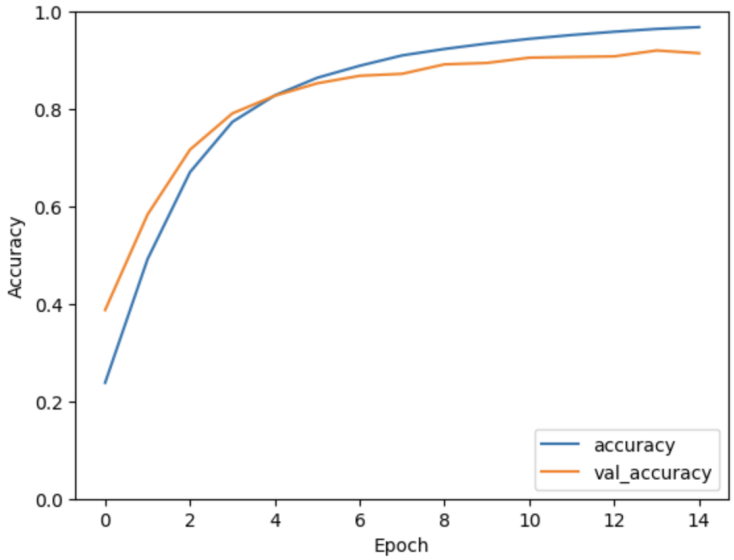
4.3 Creating the Model:

The model was made using the Sequential model from the Keras library in Python. The model has 4 layers in total. The first three layers are Rectified Linear Unit(ReLU) activation function layers with output space dimensionality of 256, 128, and 64 respectively. The fourth layer is a softmax activation function layer with output space dimensionality of 26. The model was optimized using the Adam algorithm with a learning rate of 0.0001. We used a smaller learning rate than the 0.001 that the Adam algorithm defaults to because the dataset was relatively small. The model's loss was found by using Sparse Categorical Cross-entropy.

4.4 Training and Evaluation:

```
Epoch 1/15
1625/1625 [==============================] - 14s 8ms/step - loss: 6.3801 - accuracy: 0.2382 - val_loss: 2.3300 - val_accuracy: 0.3875
Epoch 2/15
1625/1625 [==============================] - 14s 8ms/step - loss: 1.8774 - accuracy: 0.4920 - val_loss: 1.5213 - val_accuracy: 0.5835
Epoch 3/15
1625/1625 [==============================] - 11s 7ms/step - loss: 1.1979 - accuracy: 0.6705 - val_loss: 1.0763 - val_accuracy: 0.7166
Epoch 4/15
1625/1625 [==============================] - 11s 7ms/step - loss: 0.8161 - accuracy: 0.7737 - val_loss: 0.8101 - val_accuracy: 0.7912
Epoch 5/15
1625/1625 [==============================] - 12s 7ms/step - loss: 0.6098 - accuracy: 0.8281 - val_loss: 0.6863 - val_accuracy: 0.8275
Epoch 6/15
1625/1625 [==============================] - 11s 7ms/step - loss: 0.4767 - accuracy: 0.8642 - val_loss: 0.6190 - val_accuracy: 0.8529
Epoch 7/15
1625/1625 [==============================] - 11s 7ms/step - loss: 0.3842 - accuracy: 0.8889 - val_loss: 0.5315 - val_accuracy: 0.8683
Epoch 8/15
1625/1625 [==============================] - 11s 7ms/step - loss: 0.3098 - accuracy: 0.9101 - val_loss: 0.5271 - val_accuracy: 0.8724
Epoch 9/15
1625/1625 [==============================] - 11s 7ms/step - loss: 0.2585 - accuracy: 0.9234 - val_loss: 0.4758 - val_accuracy: 0.8918
Epoch 10/15
1625/1625 [==============================] - 12s 7ms/step - loss: 0.2170 - accuracy: 0.9346 - val_loss: 0.4541 - val_accuracy: 0.8947
Epoch 11/15
1625/1625 [==============================] - 12s 8ms/step - loss: 0.1856 - accuracy: 0.9442 - val_loss: 0.4274 - val_accuracy: 0.9055
Epoch 12/15
1625/1625 [==============================] - 13s 8ms/step - loss: 0.1572 - accuracy: 0.9521 - val_loss: 0.4284 - val_accuracy: 0.9069
Epoch 13/15
1625/1625 [==============================] - 15s 9ms/step - loss: 0.1361 - accuracy: 0.9588 - val_loss: 0.4258 - val_accuracy: 0.9082
Epoch 14/15
1625/1625 [==============================] - 13s 8ms/step - loss: 0.1194 - accuracy: 0.9646 - val_loss: 0.3927 - val_accuracy: 0.9203
Epoch 15/15
1625/1625 [==============================] - 12s 7ms/step - loss: 0.1042 - accuracy: 0.9682 - val_loss: 0.3993 - val_accuracy: 0.9147
```

Using the above results, we plotted the training accuracy. For 15 epochs, we resulted in accuracy increasing from 23.8% to 96.8% and the loss decreasing from 6.3 to 0.1. After we created and trained our model, we decided to test the model on handwritten letters of our own. However, the model was overall unsuccessful in predicting our handwritten letters, with accuracy of about 0.35. The unexpected disparity between the model's performance on the training set and our own data prompts a need for investigation into potential factors.. We believe that a potential factor in this subpar performance is limited data diversity, and to solve this problem, we would need to train the model on more than 3000 samples per letter.



5 Conclusion:

Our project aims to address the demand for accurate handwritten letter recognition systems by utilizing a Convolutional Neural Network model. Using a dataset of handwritten letters from Kaggle, we constructed our model using Python's Keras library. After 15 epochs of training, the model increased to around 97% recognition accuracy. When applied to our own handwritten letters, the model's accuracy reduced to only 35%. Researchers have utilized other models, such as Extreme Learning Machines, to achieve higher rates of accuracy than we were able to accomplish. The primary use for these handwritten letter recognition models is automated processing of handwritten forms, which has governmental, historical, economic, and medical benefits.

6 Reproducibility:

The code implementation of our project, including data reading and preprocessing, model creation, model training, and evaluation can be found in our GitHub repository in the references below. Others can reproduce our method and results by running our code and using the provided dataset. Note: the dataset's CSV is larger than GitHub allows, so it is stored as a .zip file, and the CSV will need to be extracted.

7 References:

1. https://github.com/Matt4228/Intro-To-Machine-Learning-Final-Project

2. Dang, Mai Ngoc. “Handwritten Letter Recognition By Using Tensorflow With Deep Learning Machine.” California State Polytechnic University, Pomona, 2018.

3. Song, J. et al. ”Research on Handwritten Alphabet Recognition System Based on Extreme Learning Machine.” Proceedings of the 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp. 9448-9451. doi: 10.23919/ChiCC.2018.8482669.

4. Dataset source: https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format