Kevin Gutierrez – CSCI 495 Deep Learning – Research Assignment

Introduction

The purpose of this lab report is to describe the process of creating a neural network in python and training it using the EMNIST letters dataset in order to predict handwritten letter values

Equipment List

1. Visual Studio Code

Overview

A neural network was requested to be developed and trained on a particular dataset. I elected to have the neural network classify handwritten letters using the EMNIST dataset. This means that there are 26 possible classes, one for each letter of the alphabet.

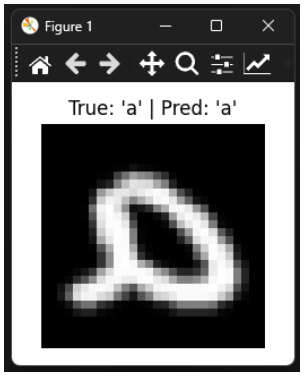
The EMNIST handwritten alphabet dataset was chosen due to its ease of accessibility and similarity between the MNIST handwritten digit dataset. Thus, the logic covered in the lectures could be used as a basis for developing this neural network. The biggest differences between my neural network and the one covered in the lectures are that my neural network:

1. uses the ReLU over the tanh hidden layer function, to avoid vanishing gradients and help with speeding up learning
2. Uses the softmax instead of the sigmoid activation function in the output layer – this is because the softmax function ensures that the output sum is equal to 1. This helps with picking the most probable letter
3. Uses the categorical crossentropy error function over the mean squared error function as it works better for classification
4. Uses 200 neurons
5. Uses 8 epochs

I implemented functionality to predict the EMNIST test data values, and also my own handwritten letters in ./MyHandwrittenLetters.

Problems encountered

At first, while using my own handwritten letters, I couldn't get the model to predict any of my handwritten letters at all. Then I realized that the model was fed mirrored letters in this format:

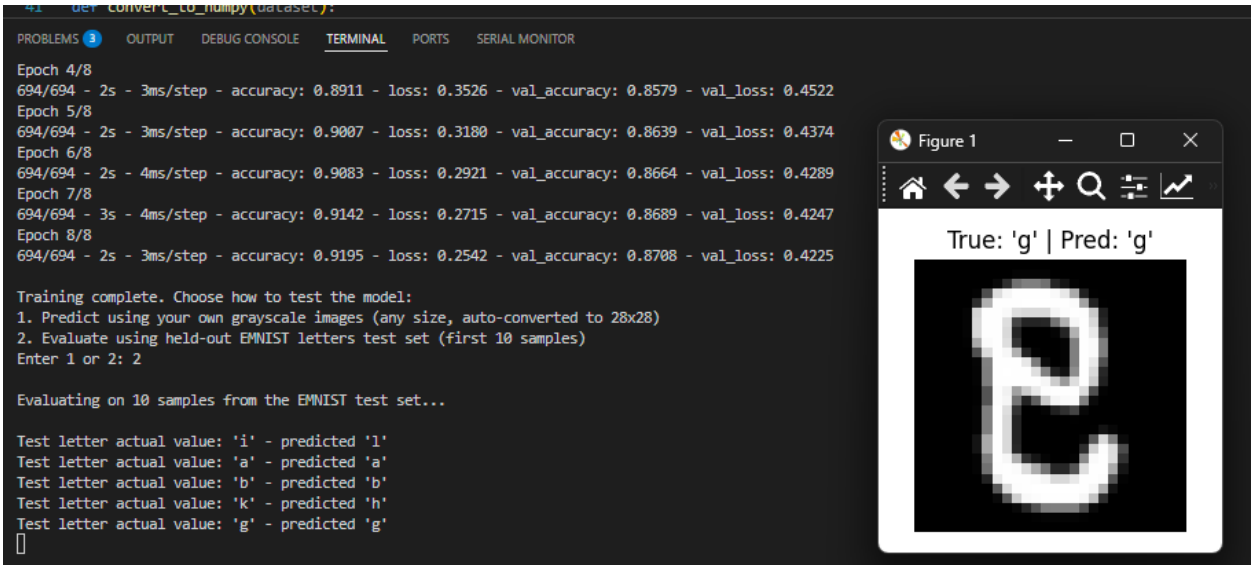


Notice how the image is mirrored. Also notice how the letter is white and the background is black – that was another source of error in this project. Once I programmatically mirrored my letters and inverted the colors, I was able to get predictions to work.

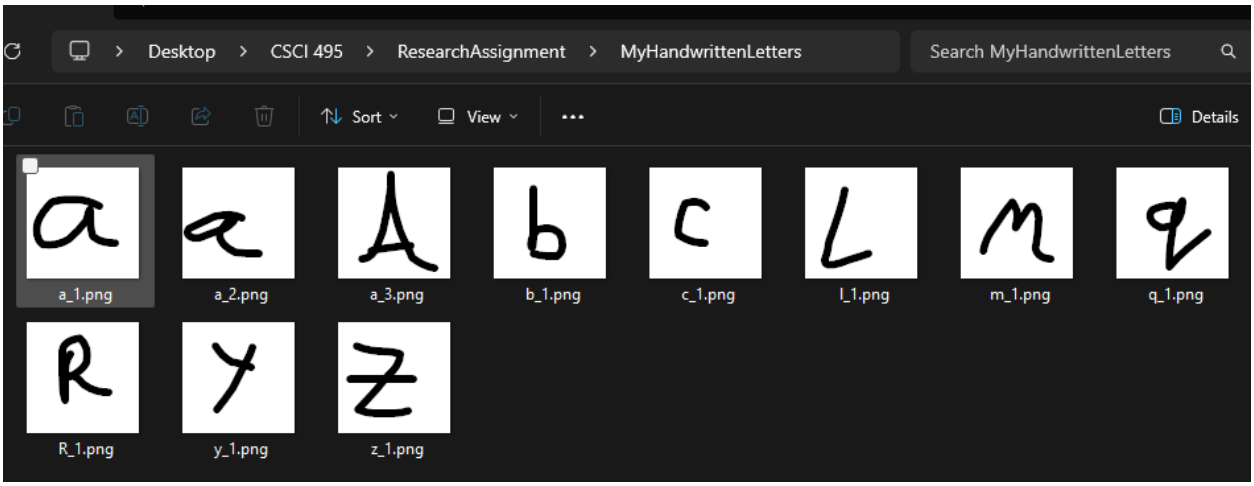
I also encountered a very low accuracy rate for my own handwritten letters – this was improved by increasing the epochs and number of neurons. I believe this was caused due to my letters being

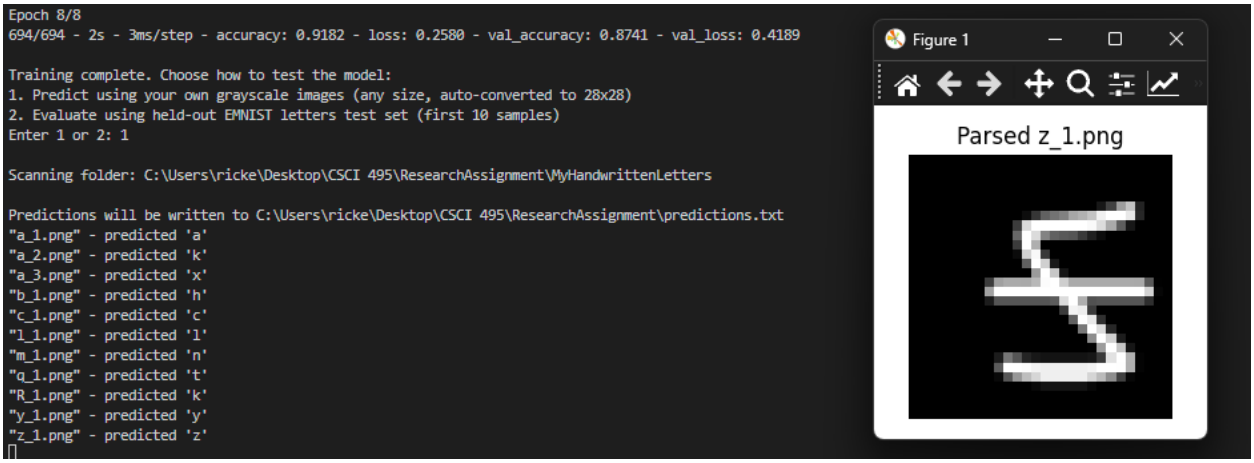
Example Run

In this run, I used EMNIST test data letters

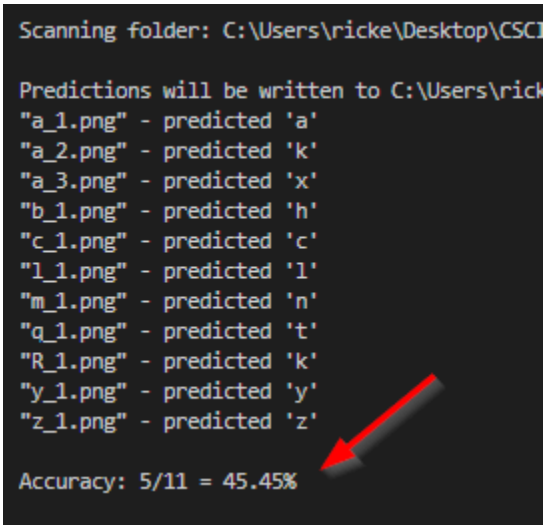


In this run, I used my own personal letters. Here are a few of them:





I got up to 45.45% accuracy with 11 of my handwritten letters:



```
ONEDNN_OPTS=0`.
Loading EMNIST Letters dataset...
2025-06-15 18:24:22.060972: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-06-15 18:24:52.777661: I tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
2025-06-15 18:24:58.219953: I tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
C:\Users\ricke\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
n the model instead.
  super().__init__(**kwargs)

Training model on EMNIST Letters...
Epoch 1/8
694/694 - 3s - 4ms/step - accuracy: 0.7356 - loss: 0.8926 - val_accuracy: 0.8046 - val_loss: 0.6515
Epoch 2/8
694/694 - 2s - 4ms/step - accuracy: 0.8465 - loss: 0.5080 - val_accuracy: 0.8371 - val_loss: 0.5345
Epoch 3/8
694/694 - 2s - 3ms/step - accuracy: 0.8742 - loss: 0.4099 - val_accuracy: 0.8507 - val_loss: 0.4824
Epoch 4/8
694/694 - 2s - 3ms/step - accuracy: 0.8892 - loss: 0.3571 - val_accuracy: 0.8614 - val_loss: 0.4542
Epoch 5/8
694/694 - 2s - 4ms/step - accuracy: 0.8993 - loss: 0.3220 - val_accuracy: 0.8657 - val_loss: 0.4379
Epoch 6/8
694/694 - 3s - 4ms/step - accuracy: 0.9067 - loss: 0.2961 - val_accuracy: 0.8707 - val_loss: 0.4272
Epoch 7/8
694/694 - 2s - 3ms/step - accuracy: 0.9131 - loss: 0.2753 - val_accuracy: 0.8728 - val_loss: 0.4208
Epoch 8/8
694/694 - 2s - 3ms/step - accuracy: 0.9182 - loss: 0.2580 - val_accuracy: 0.8741 - val_loss: 0.4189

Training complete. Choose how to test the model:
1. Predict using your own grayscale images (any size, auto-converted to 28x28)
2. Evaluate using held-out EMNIST letters test set (first 10 samples)
Enter 1 or 2: 1

Scanning folder: C:\Users\ricke\Desktop\CSCI 495\ResearchAssignment\MyHandwrittenLetters

Predictions will be written to C:\Users\ricke\Desktop\CSCI 495\ResearchAssignment\predictions.txt
"a_1.png" - predicted 'a'
"a_2.png" - predicted 'k'
"a_3.png" - predicted 'x'
"b_1.png" - predicted 'h'
"c_1.png" - predicted 'c'
"l_1.png" - predicted 'l'
"m_1.png" - predicted 'n'
"q_1.png" - predicted 't'
"R_1.png" - predicted 'k'
"y_1.png" - predicted 'y'
"z_1.png" - predicted 'z'

Accuracy: 5/11 = 45.45%
```

Conclusion

This was a phenomenal introduction to developing neural networks. I learned even more about hidden layer functions and their use cases (tahh vs ReLU), and activation functions (softmax vs sigmoid).