Introduction

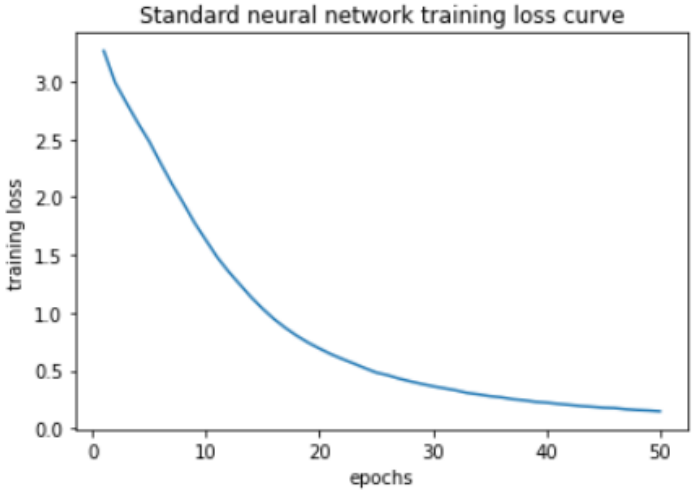
The project objective is to train models in TensorFlow to reliably classify handwritten Greek letters. The dataset is published by Katiana Kontolati and can be found on Kaggle: kaggle.com/datasets/katianakontolati/classification-of-handwritten-greek-letters/. The dataset consists of 14x14 images of handwritten Greek letters: there are 240 training images and 96 testing images. We shall try a simple connected neural network on the images formatted as one-dimensional vector arrays, and a convolution neural network on the images as 14x14 matrix arrays. It is anticipated that some of the letters will be misclassified more than others; some letters have more similar shapes than others.

Method

We begin by importing the dataset using a pandas framework. The grayscale value (0 to 255) for each pixel is given in a row vector, with the final column containing the label. The label of the image is given as a value of 1 to 24 corresponding in order of the Greek alphabet. In order to properly use a sequential Keras neural network, I have used numpy broadcasting to subtract 1 from each label to relabel using a value of 0 to 23. The grayscale values are also normalized by dividing each dataset by 255 to achieve better results when training.

Model 1

The first neural network model is trained using a simple connected layer of 96 nodes with relu activation then passed to the classifying connected layer of 24 nodes with softmax activation. The training loss curve was ideal (below) and evaluating the model on the testing set yielded good results (0.90 acc, 0.39 loss).

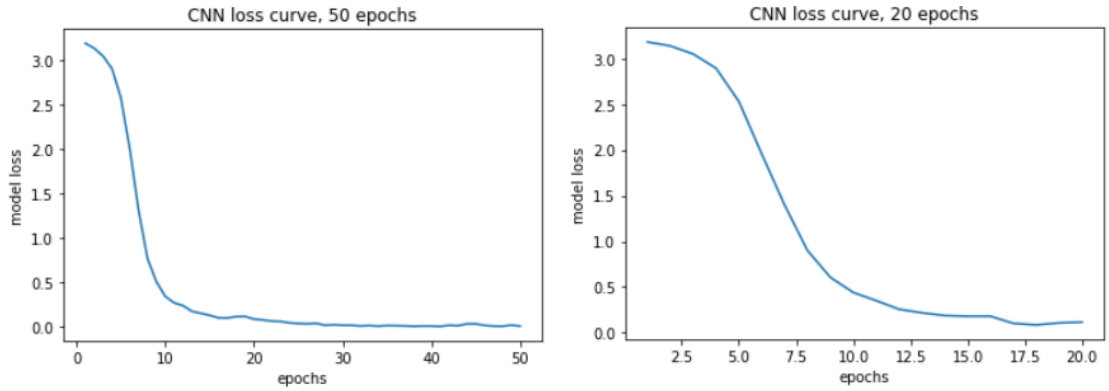


Next, we present a table of misclassifications of the letters by the neural network. Upon inspection of the table, it is apparent how these letters could have been misclassified: Pi and Eta have ‘n’ shapes; Phi and Psi have ‘pitchfork’ shapes. The cases of single misclassifications also make sense, depending on how the letter was written: Sigma and Omicron have ‘o’ shapes; Chi and Kappa have ‘x’ or ‘cross’ shapes. Strangely, we see that delta and epsilon are confused for gamma.

Image number (out of 96)	Real label (out of 24)	Predicted label (out of 24)
14	Delta	Gamma
17	Epsilon	Gamma
38	Kappa	Chi
60	Pi	Eta
61	Pi	Eta
67	Sigma	Omicron
80	Phi	Psi
81	Phi	Psi
83	Chi	Kappa
87	Psi	Nu

Model 2

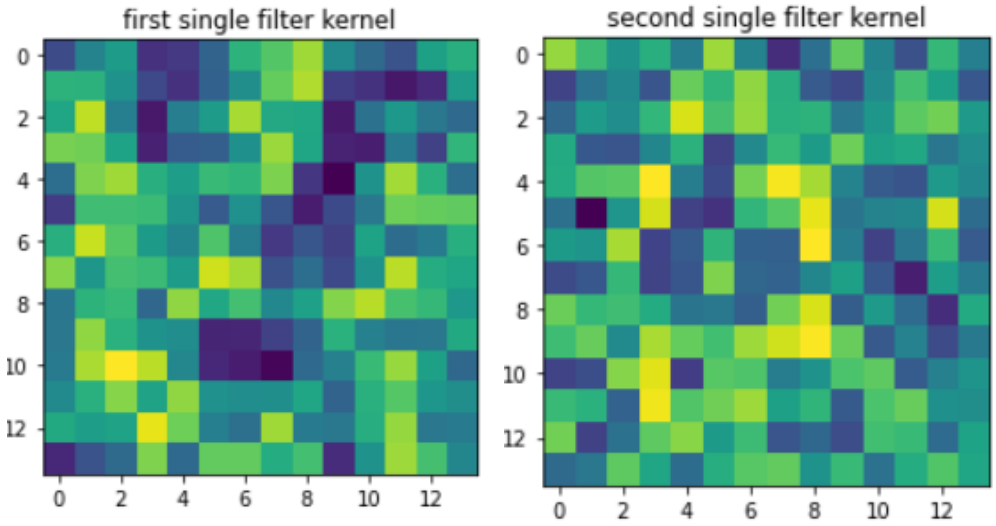
The second neural network model is trained using a variety of convolutional, pooling, and connected layers. Notice in the graphs below that for the small 14x14 size of the image data, it turns out that fewer epochs make for a better loss curve. This is because the small training set is easy to overfit: indeed, we find that when evaluating the model on the small testing set in this 50 epoch case, the accuracy is low. However, when we remove a pooling and convolutional layer and reduce to 20 epochs, the testing set has good accuracy and loss.



Below, we present another misclassification table of the 20-epoch CNN. Again, there are standard misclassifications between similarly shaped letters. We also see the strange misclassifications between delta, epsilon, and gamma.

Image number (out of 96)	Real label (out of 24)	Predicted label (out of 24)
14	Delta	Gamma
17	Epsilon	Beta
27	Theta	Omicron
38	Kappa	Chi
60	Pi	Sigma
63	Rho	Mu
68	Chi	Kappa
83	Psi	Nu
87	Phi	Psi
90	Phi	Psi

Below are sample image kernels when we use only one filter for each of the convolutional layers. Using single filters yields decent (0.95 acc, 0.21 loss) accuracy during training, and subpar performance when evaluating the model on the testing set (0.78 acc, 0.70 loss).



We attain our optimal results (0.92 acc, 0.27 loss) on the testing set when we use multiple filters in two convolutional layers of size 14 or less.

Conclusion

This project revealed that in the case of this small dataset with small image sizes, a convolutional neural network does not necessarily outperform the standard network with the image data in vector form. We also find that less is more: fewer layers contribute to a better loss curve since the network does not overfit the training data. Lastly, while it is interesting to see how the CNN creates an image kernel filter to detect each letter, it does not yield as good results as using several filters.

Future projects

Using the models that I have trained, it would be very interesting to classify more images of handwritten Greek letters. Another interesting project could be analyzing some English letters on these models considering the similarity of some Greek letters to English ones. Obviously, plenty of letters would be misclassified but it would be interesting to see which ones my models could classify correctly.