Kyle Moorhead
12/1/2023
COSC 480

Github link: https://github.com/moorhead212/Hand-Gesture-Neural-Network
See Readme for instructions on how to run the code

Final Project Report:

Throughout this project I struggled endlessly to get the neural network to train. Over and over I tried changing hyperparameters, applying different normalization techniques, more images, less images, augmentation, the whole nine. I kept having this nagging feeling that there was something wrong with my data - or the way I was processing the data. So I went back to the drawing board and made sure everything was being encoded correctly.

That is when I found out that there were some images that had 2 hands in frame, and they had 2 associated labels. I thought I had found my "eureka" moment. However, I was not that lucky. After addressing the label issue I went back to attempting training to no avail.

After another week with no results I threw in the towel and decided I would try a different dataset to see if my architecture was the problem. I downloaded a much simpler dataset and attempted training. To my surprise it shot up to 90% accuracy on the second epoch. Now my suspicion with the dataset was back in the forefront.

With that information gathered, I transferred back to my regular dataset to determine the cause of my frustrations. One by one I started clicking through the images to try to find the variations, to see what was wrong. By about the 10th one I realized something strange, the images weren't all the same aspect ratio - in fact they weren't even the same size in some cases.

Upon further investigation, I determined that the dataset claimed all images were "1920x1080". This turned out to be entirely false. In fact, most images were 1440x1920. Determined to see if this was the cause of my problems, I tried to filter each image while processing the data to check the sizes.

This turned out to be a whole can of worms as there were many images with each having to be checked individually before being processed into data, so I decided to change my approach. I wrote a script to delete any image that did not meet the 1440x1920 ratio I had determined to be the dominant aspect ratio.

With this complete I attempted training again. My dropped as I hit the 15th epoch with the lowly 33% validation accuracy I had throughout my entire problem. But then it hit me, the structure I was using trained the simple dataset very quickly - but my dataset was much more complicated than that one, so I added *many* more Conv2d layers. 3 epochs in, I was still staring at the 33%, frustration mounting.

Abruptly on about the 5th epoch, the validation accuracy shot up to 45% - a number I had never seen it make it to. The 6th made it to 60%. I hit the ceiling with joy. The best validation I've hit is 70%, despite all attempts to adjust, but that is still insane progress from the start of the project.

Thanks for reading, I hope you enjoyed my wild journey to learning more about Neural Networks.

Kyle Moorhead
12/1/2023
COSC 480

1. **Challenges Along the way**
    a. **Pre-processing Images**
        i. Dataset claimed all images were 1920x1080. Reality: Most images 1440x1920. The variation in size and shape of images was distorting training data so the model failed to train.
    b. **Multi-Labeled images**
        i. Some Images had multiple hands in frame confusing the training.
    c. **Network Structure**
        i. I had been using too simple of a structure for the complexity of my problem. Not enough Conv2D layers to allow for the image to be processed properly.
    d. **AMD graphics cards are dumb**
        i. They don't play well with TensorFlow – so I couldn't use it to speed up training.

2. **Lessons Learned**
    a. **Pre-process data**
        i. I realized my network would take much less time if I compiled the images into their numpy versions and saved them separately so I could train over and over on the same set of data (I had a total of 24,000 images per class to choose from).
    b. **L2 normalize**
        i. Attempted to L2 normalize trying to understand why my model was not training. (This was before determining the image size/shape **problem.**
    c. **Dropout Layers**
        i. Attempted multiple dropout layers in different places in the model structure trying to get the model to train. (Again before image size/shape problem was solved).
    d. **Batch Normalization**
        i. Attempted batch normalization after first dense layer, before categorizing layer. (Again before image size/shape problem)
    e. **Data Augmentation**
        i. Briefly attempted data augmentation. Takes existing dataset and shuffles it around. Rotates, flips, zooms – in an attempt to create a more diverse dataset if original is lacking.
        ii. Gave up on this and instead increased number of training images. 5000 was about the most I could process in a "reasonable" amount of time.
    f. **Early Stopping**
        i. Utilizing early stopping to check at the end of each epoch if the loss is still decreasing. If not it will stop and save the best weights.
    g. **Every little thing matters**

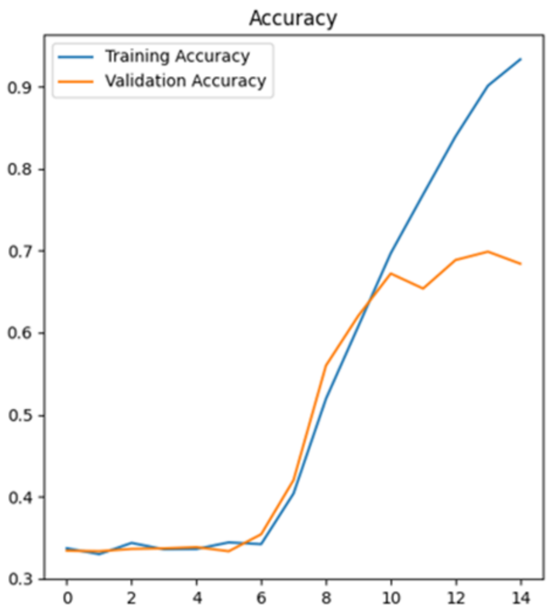
Kyle Moorhead
12/1/2023
COSC 480

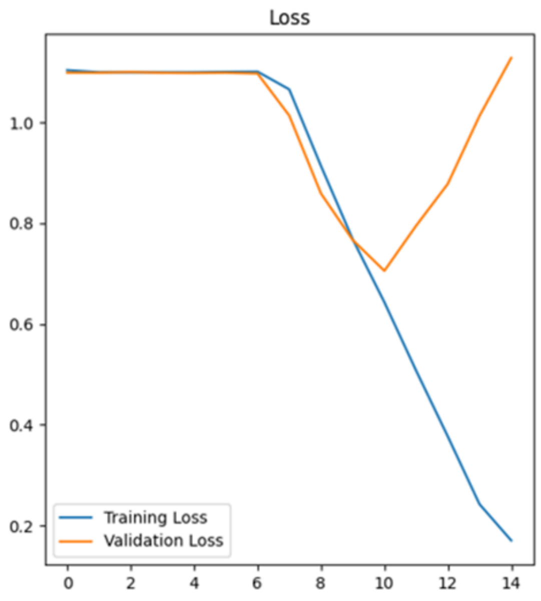
i. After I solved the image shape/size problem training still didn't seem to work. So, I grabbed a different dataset and tried to train my on my architecture. It worked great – but the dataset was much simpler than mine. I realized my architecture would need to be more complex to handle the task at hand. When I increased the number of layers to 6 the model trained for the first time ever!

3. Design Choices

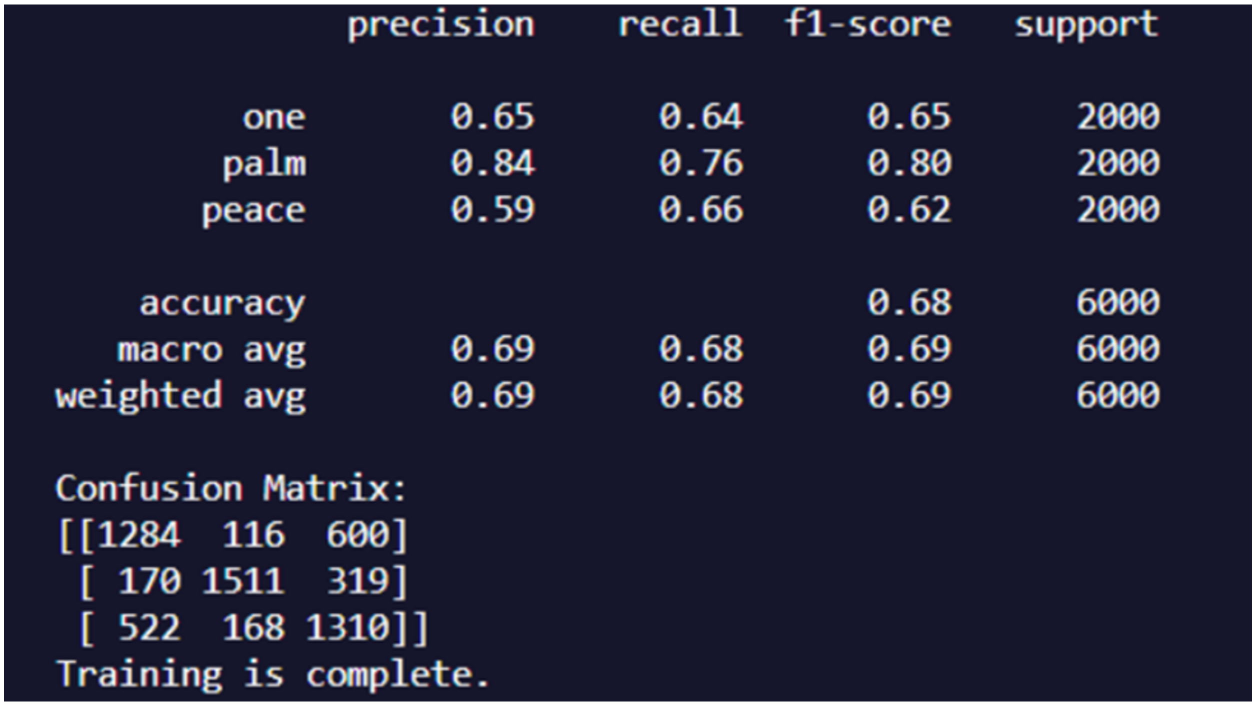
a. **Epochs:** 15
b. **Conv2d Layers:** 6
c. **Optimizer:** Adam
d. **Images:** 5000 per gesture
e. **Batch Size:** 32
f. **Loss:** Categorical Crossentropy
g. **Train/Test Split:** 60/40

4. Best Result:





Kyle Moorhead
12/1/2023
COSC 480



```
              precision    recall  f1-score   support

         one       0.65      0.64      0.65      2000
        palm       0.84      0.76      0.80      2000
       peace       0.59      0.66      0.62      2000

    accuracy                           0.68      6000
   macro avg       0.69      0.68      0.69      6000
weighted avg       0.69      0.68      0.69      6000

Confusion Matrix:
[[1284  116  600]
 [ 170 1511  319]
 [ 522  168 1310]]
Training is complete.
```