Samuel Kersebet
CS411G
05/18/21

CS411G: CNN Image data exploration

Task: To use an existing data set of images and compare and contrast the effects of training and testing in my own image data to the set. The experiment is to learn how to build a CNN using different dataset tools than I had been exposed to and to see if I can make my own images play nice with the dataset.

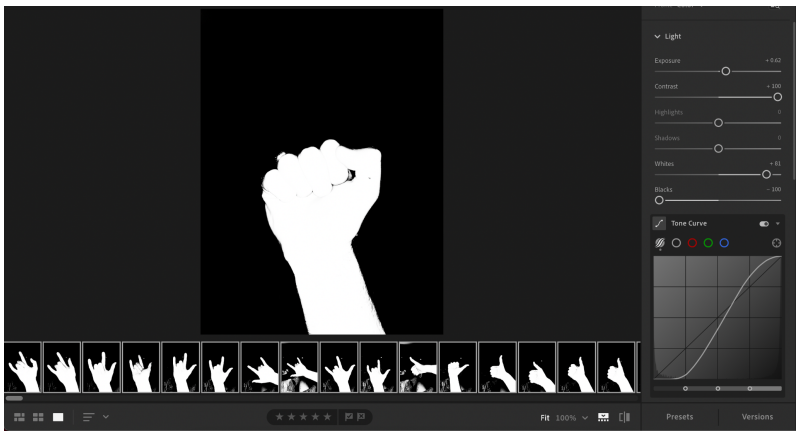
Kaggle Dataset: This data set is a series of 2646 images of hand symbols from Kaggle.com. On download it was given to me as folders of 320x320 pixel black and white image files organized by the specific hand symbols. These 6 classes of symbols were Blank, Fist, Five, Thumbs up, Two, and Yo. The blank files were images with no hands but some potential noise in the void of the hand symbol. Fist images were all different images of hands in fists. Five images were images of a hand spread with five fingers distinctly visible. Thumbs up are images as they would sound. Two is the same as a peace sign. And yo is the rock and roll hand. https://www.kaggle.com/flintytub49/hand-gestures-black-and-white

Excerpt of kaggle dataset

Personal Dataset: On top of the images I found online I also created some 47 photos of my own hand that fell into one of the Kaggle Datasets categories.

Excerpt of my hand images.

Tools: For this assignment I used the Keras library to create my datasets and to build my CNN models and test them. I also used a confusion matrix from scikit learn to display my prediction results. For my own data I used my Iphone with a flash on as well as Adobe Lightroom to manipulate the contrast and exposure levels to make the images more similar to the Kaggle images.

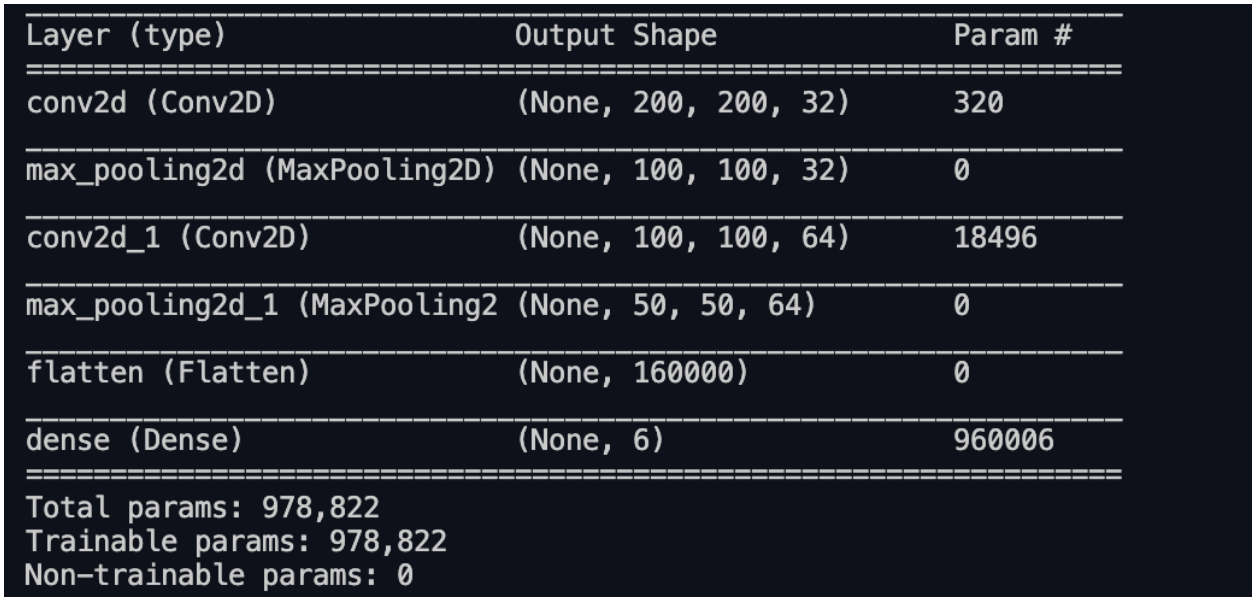


Using adobe lightroom to manipulate my hand images.

Preprocessing: To create a data structure from the kaggle dataset I used a tool in the Keras library called ImageDataGenerator, while this was a very easy to use tool in the end it did take some adjustment from using numpyArrays to make everything work. Some great features of the ImageDataGenerator were that I was able to classify each image based on the name of the directory they were stored in. I also was able to easily split a portion of the data into training and testing sets. I used the same tool to create a test data set of my own images and also made training and test sets of the combination of my data and the kaggle data. All images were converted down to greyscale images with a target size of 200x200 pixels and I used a batch size of 20 images.

The models: For this assignment I created two different Sequential models. All models used 10 epochs of training.

Model one was the simpler one and its summary includes a convolutional layer, into pooling layer, into another convolutional layer into another pooling layer, then flattened and finally output to a dense layer using softmax. All specifics will be shown in a screencap of the summary.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 200, 200, 32)	320
max_pooling2d (MaxPooling2D)	(None, 100, 100, 32)	0
conv2d_1 (Conv2D)	(None, 100, 100, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 50, 50, 64)	0
flatten (Flatten)	(None, 160000)	0
dense (Dense)	(None, 6)	960006

Total params: 978,822
Trainable params: 978,822
Non-trainable params: 0

Time
hands.h5 - Model 1 training on the Kaggle only training set: 151.92 seconds
hands2.h5 - Model 1 training on kaggle and personal data: 165.98

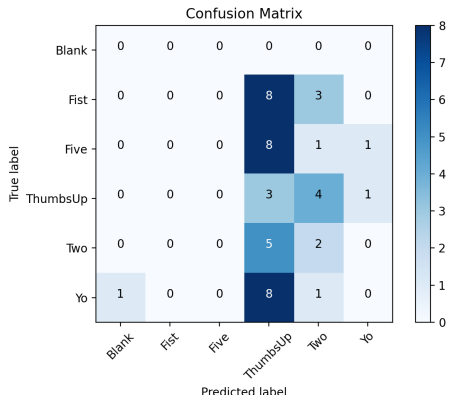
Model two was more complicated: it began with a convolutional layer into another convolutional layer, into a pooling layer. Then had a 25% dropout layer. Back into a convolutional layer, again convolutional layer, pooling layer, another 25% dropout, a flatten layer into a 256 node dense layer, into a 50% drop out and finally the output layer. Specifics below.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 200, 200, 32)      832
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 200, 200, 32)      25632
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 100, 100, 32)      0
_________________________________________________________________
dropout_3 (Dropout)          (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 100, 100, 64)      18496
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 100, 100, 64)      36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 50, 50, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 50, 50, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 160000)            0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               40960256
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 1542
=================================================================
Total params: 41,043,686
Trainable params: 41,043,686
```

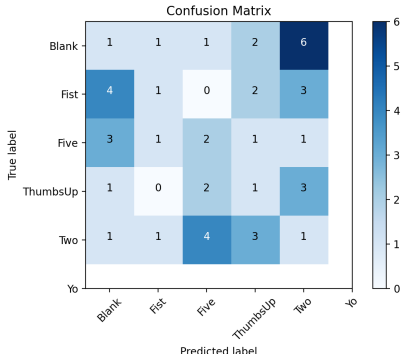
Time
hands3 - Model 2 training on Kaggle only training set : 1028.019 seconds
hands4 - Model 2 training on Kaggle and personal data : 1132.312 seconds

Results:
I performed a total of 4 tests.

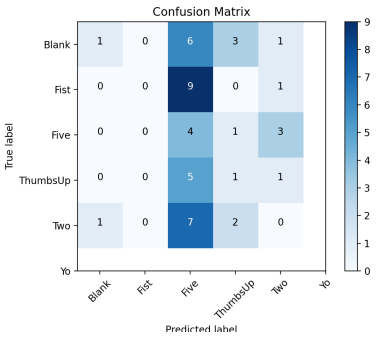
Test1: hands.h5(Model1 kaggle only training) vs personal test data set
Accuracy: 0.304



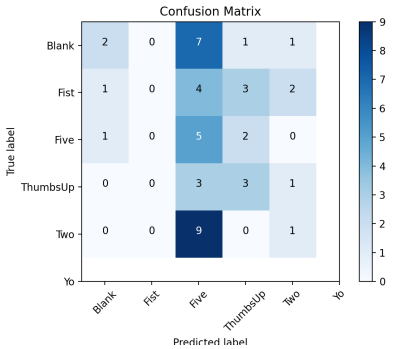
Test2: hands2.h5(Model1 kaggle and personal data) vs personal test data set
Accuracy: 0.826



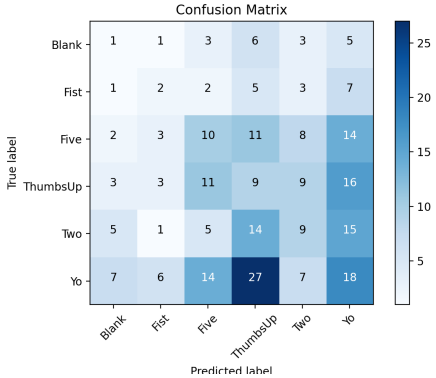
Test3: hands3.h5(Model2 kaggle only training) vs personal test data set
Accuracy: 0.304



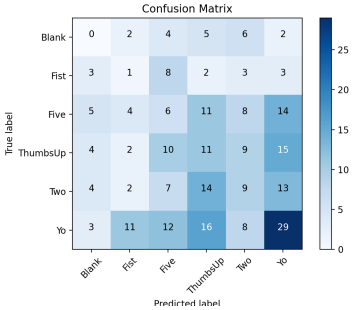
Test4: hands4.h5(Model2 kaggle and personal data) vs personal test data set
Accuracy: 0.369



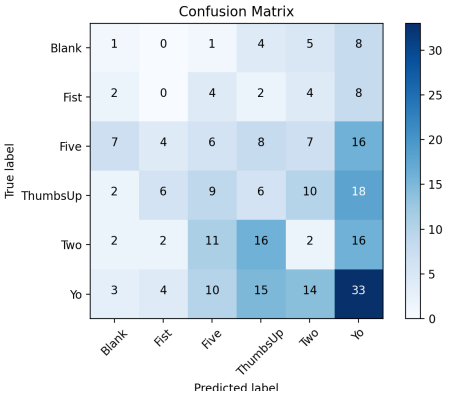
Test5: hands.h5(Model1 kaggle only training) vs combo set
Accuracy: 0.736



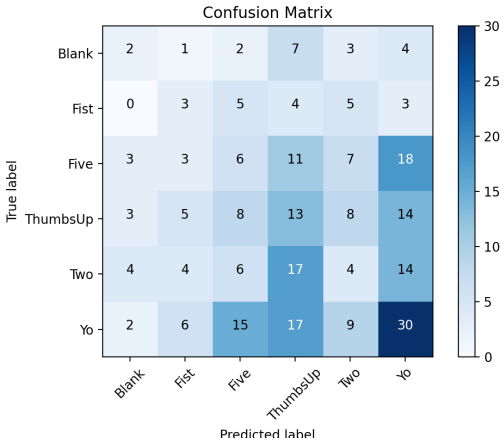
Test6: hands2.h5(Model1 kaggle and personal data) vs combo test
Accuracy: 0.786



Test7: hands3.h5(Model2 kaggle only training) vs combo test
Accuracy: 0.793



Test8: hands4.h5(Model2 kaggle and personal data) vs combo test
Accuracy: 0.782



Comments:
It is apparent to see that some of these tests are redundant, specifically test 2 which stands out with its 82% accuracy due to testing on already seen training examples. Test 4 is also impure but due to also training on model2 with training data in the test set, yet interestingly enough has a hard time identifying the inputs at a 37% accuracy rating. The purest of the tests are test 1 and 3, which use the models trained on only the kaggle training data and tests on only my personal data, these both saw accuracy ratings of 30%. For tests 5-8 I used the same models but with a test set that had a mix of kaggle and personal data, this was created by sampling 20% of the combined set. Test 5 and 7 had potential for overlap since models hands and hands3 used training data from a kaggle only set which was split for validation independently. Tests 6 and 8 used training and test sets independent from each other and therefore are the most insightful results, giving us a 79% accuracy when trained on model 1 and a 78% accuracy when trained on model 2.

Conclusions:

While I realize it is not good practice to test on data in the training set I think that in this situation it can actually give us valuable insights. The main question here is about the possibility to combine data that has not been created and pre-processed under the same parameters. While I did do some preprocessing to the images I created myself I think the test results show how apparent the importance of making the input data play nice with the models is. I was initially excited to see that 30% accuracy was even able to be achieved by the model but given the number of classes this is essentially random guessing. The tests 6 and 8 show promising results with no overlap of test and training data, yet the ratio of my personal data to the kaggle data is not substantial enough to make a serious comment about and these tests are more useful for their insight as a baseline to compare tests 1 and 3 to. The last observation that I could make is about the unnecessary depth of the neural network that is described in model2. If anything the complexity of the NN made it even harder for the models to interpret what my personal image data was feeling to them. Even though model2 generally too 10x the time to train, it could not understand the images I took on my iphone. This entire process and experiment has made me realize that in order to get useful predictions a lot of work must be done to process the images to be somewhat cohesive. An extension of this project would include building a larger data set of my personal images and learning more about the preprocessing, training a model based on my personal images and testing the kaggle data against it.