Homework 5
Name: Chakrya Ros

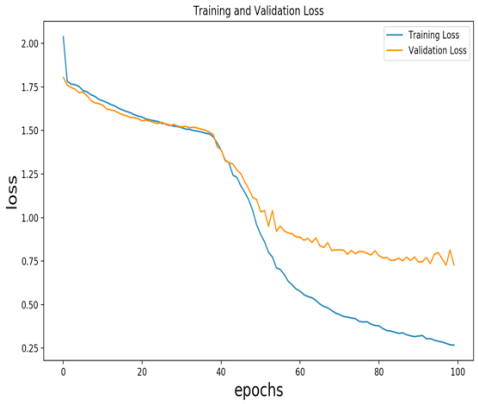
Report

Part 1:

- The quality of the dataset is pretty good, but some of the images didn't work so I just deleted it. It mostly provided the bounding boxes accurate.
- The preprocessing is important in the context of classifying faces because it's cleaned, speed up training like centering image and scaling techniques. The smaller image models are much faster to train.
- The effect of resizing on the images is that some images are blurry, but it's good to train model faster with smaller images sizes.

Part 2:

- Plot the loss graphs



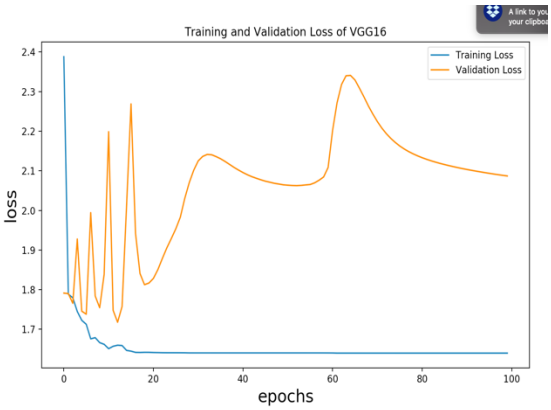

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                115232
_________________________________________________________________
activation_1 (Activation)    (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
_________________________________________________________________
activation_2 (Activation)    (None, 6)                 0
=================================================================
Total params: 115,430
Trainable params: 115,430
Non-trainable params: 0
```

- The model show that the validation loss did not improve from epochs 50 to 100 while the training loss was decreasing. It means that it's overfitting. The test loss is about 0.72 and the test accuracy is about 0.79. That's pretty good.
- The description of my model, firstly I use sklearn.model_selection for splitting dataset into train set and test set and I set test size to 30% of the dataset. I called preProcessImages() function to crop and resize the input images from raw input by using PIL to crop and resize. I initialized the weight by using default parameter kerner_initializer= "random_normal". I used Relu activation function for the first hidden layer and sigmoid activation function for the output layer. I used only one hidden layer that have 32 nodes.

Part 3:

- Block4_pool of VGG16 used as input because it has four block of hidden layers and each block have 2 to 3 Cov2D and one MaxPooling2D.
- Using features from VGG16 using raw images from part2, it is worse than performance model in part 2 because it has more trainable parameter and take features that are already train on Vgg16 to train on sequential model, therefore it gets worse.

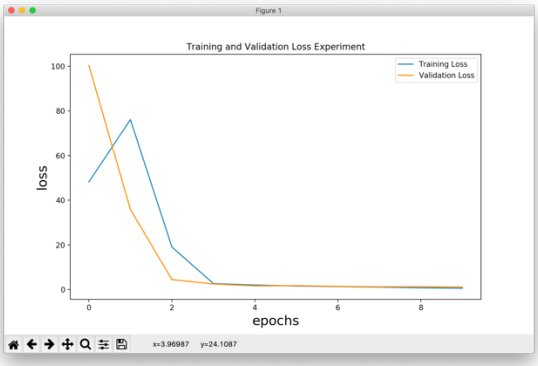
-




```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                3211296
_________________________________________________________________
activation_1 (Activation)    (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
_________________________________________________________________
activation_2 (Activation)    (None, 6)                 0
=================================================================
Total params: 3,211,494
Trainable params: 3,211,494
Non-trainable params: 0
_________________________________________________________________
```

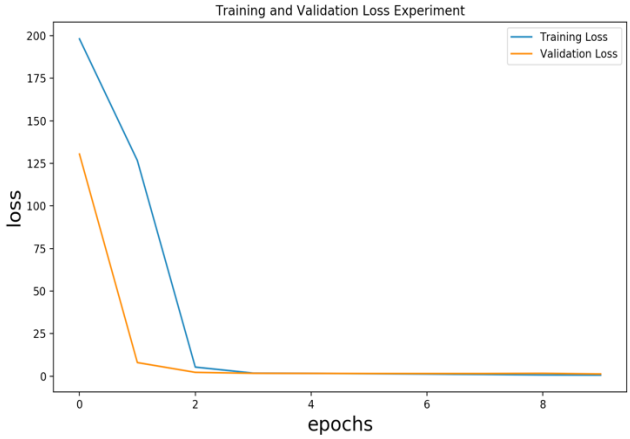
Part 4

- First experiment on trainFaceClassifier, I wrote the new function that is called ExperimentFaceClassifier(). Inside that function, I used three Conv2D and three MaxPooling2D hidden layer to build a linear stack of layers with the sequential model. The first layer, 2D convolution layer that have 16 nodes and 448 parameters that produce output shape equal to (None,224, 224, 16). The second layer, 2D max pool that have 16 nodes that produce output shape to (None, 112, 112, 16), it means that it's subsampling from (None,224, 224, 16) to (None, 112, 112, 16).
  The third layer, 2D convolution layer that have 32 nodes and 4640 parameters that produce out shape equal to (None, 112, 112, 32). The four layer, 2D max pool that have (None, 56, 56, 32). The fifth layer, 2D convolution layer that have 62 nodes and 17918 parameters that produce out shape equal to (None, 56, 56, 62). The sixth layer, 2D max pool that have output shape equal to (None, 26, 26, 62). The next layer is Flattern to make it into 1D. And the last layer is Dense layer that produce 6 output using SoftMax activation function. This experiment produced a pretty good graph of train loss and Validation loss. The loss scores equal to 173.34 at epoch 1 and keep decreasing to 0.7327 at epoch 10. And the accurate score = 0.7725 is also good at epoch 10. Comparing this experiment model to part 2 and part 3, this one is the best.
  - Test loss: 1.3118161227968004
  - Test accuracy: 0.1666666716337204




```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 16)      448
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 112, 112, 32)      4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 56, 56, 62)        17918
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 62)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 48608)             0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 291654
=================================================================
Total params: 314,660
Trainable params: 314,660
Non-trainable params: 0
_________________________________________________________________
```

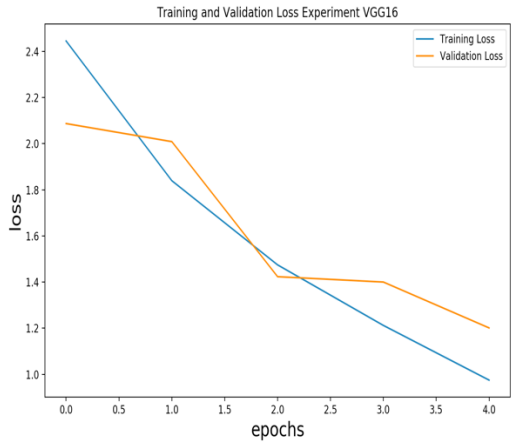
- The second experiment, everything is the same as the first experiment, but I add two more layer for dropout and set it to 25%. It means during training of each hidden layer; the nodes are random deleted about 25% of them. Thus, the loss is converged very faster than the first experiment. The loss scores equal to 1933.16 at epoch 1 and keep decreasing to 0.6032 at epoch 10. And the accurate score = 0.7884 is also good at epoch 10.
    - Test loss: 1.1947390503353543
    - Test accuracy: 0.5555555820465088



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 14, 14, 6)	3078
conv2d_1 (Conv2D)	(None, 14, 14, 16)	880
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 16)	0
dropout_1 (Dropout)	(None, 7, 7, 16)	0
conv2d_2 (Conv2D)	(None, 7, 7, 32)	4640
max_pooling2d_2 (MaxPooling2	(None, 3, 3, 32)	0
conv2d_3 (Conv2D)	(None, 3, 3, 62)	17918
max_pooling2d_3 (MaxPooling2	(None, 1, 1, 62)	0
dropout_2 (Dropout)	(None, 1, 1, 62)	0
flatten_1 (Flatten)	(None, 62)	0
dense_2 (Dense)	(None, 6)	378

Total params: 26,894
Trainable params: 26,894
Non-trainable params: 0

- The third experiment, I wrote another function “experimentOnVGG16(directory)”, In this function, I initialed model = sequential() and then I added all the hidden layer from Vgg16 expect the last one that is prediction. And then I added Dense(6, activation= “softmax”) to produce output. This model was bad and it didn’t improve the performance. Looking at the graph, it went steeper down to bottom.

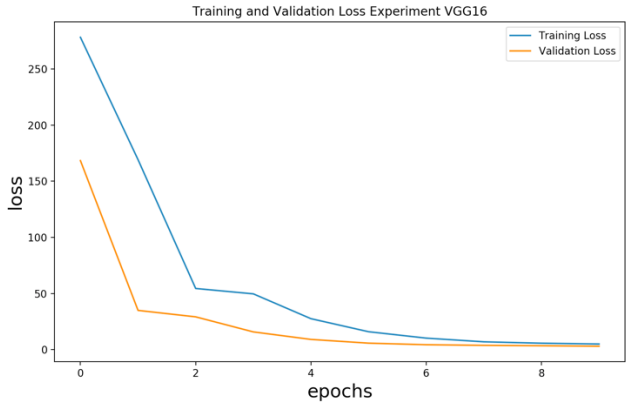


Model: "sequential_1"

Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
dense_1 (Dense)	(None, 6)	24582

Total params: 134,285,126
Trainable params: 24,582
Non-trainable params: 134,260,544

- The last experiment, it's similar to the second experiment with the same Cov2D and MaxPooling2D but I used features that were trained by Vgg16 and I used block2_pool for input layer. The graph look better than the third experiment.
    - Test loss: 3.020546736540618
    - Test accuracy: 0.25925925374031067



In conclusion, the second experiment is the best model comparing to all models that I created. It had high accuracy score and high accuracy test score among other. It also has lower loss score and loss test score among other. It's the best model because it had the least parameters to train, and I used three 2D convolutional layers with increasing nodes for each layer. And I also used dropout to improve this performance. I used SoftMax activation function to classify of these six labels. Therefore, it's best model.