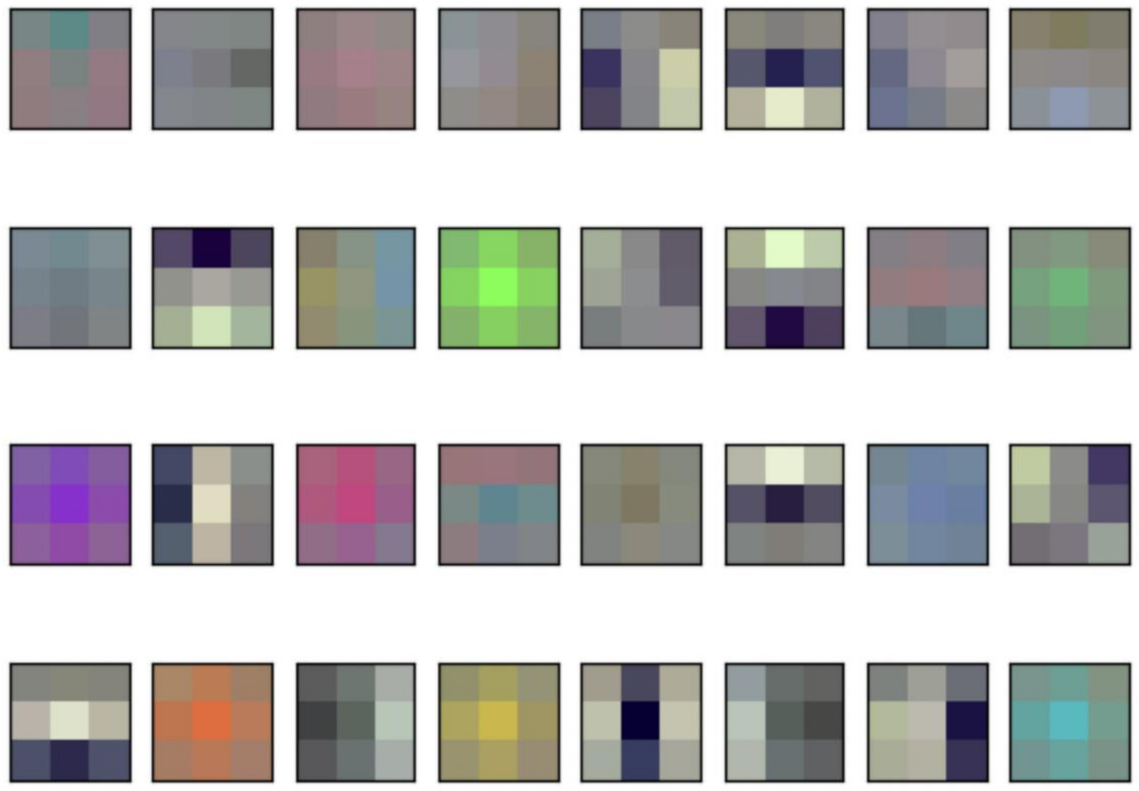
Ryan Filgas
Computer Vision
Program 3

For this program we were tasked with using the inception resnet v2 convolutional neural network as a pretrained model to use with a transfer head for cat/dog image classification. First we were asked to test the pretrained model, add a classification head, then do the same experiment with a sub-net of the pretrained model. Overall results were expected. The untrained model achieved 40% accuracy, the resnet model once trained achieved 97% and a small piece of that network achieved 60%.

1a. In the submitted files is a summary called pre_model.txt
1b.



These are the filters from the reception resnet v2 seen above represented in rgb channels. Some of the filters with dark purple may suggest that they are sensitive to edges in the blue and red channels. One is very green indicating it may be more sensitive to features in the green channel. The low contrast greyer areas may suggest that low contrast features are also considered in detection, potentially contributing to a more robust model.

2. Load and pre-process: For preprocessing I used an image data generator to mean center, normalize, rotate by .2, shift width by .2, shift height by .2, and add horizontal flips. I used flow_from_directory to resize to the 150x150 target size and set a batch type.

3. Below is the model summary of the untrained transfer model.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_resnet_v2 (Functi  (None, 3, 3, 1536)       54336736
 onal)

 flatten (Flatten)           (None, 13824)             0

 dense (Dense)               (None, 256)               3539200

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 57,876,193
Trainable params: 57,815,649
Non-trainable params: 60,544
_________________________________________________________________
```

4a. Evaluate model without training: As expected the model doesn't do well untrained and ends up at random chance.

```
Confusion Matrix
[[886 114]
 [886 114]]
Classification Report
              precision    recall  f1-score   support

        cats       0.50      0.89      0.64      1000
        dogs       0.50      0.11      0.19      1000
```

```
    accuracy                           0.50      2000
   macro avg       0.50      0.50      0.41      2000
weighted avg       0.50      0.50      0.41      2000
```

4b. Evaluate the model trained. The model trained uses binary cross entropy loss and the Adam optimization algorithm. The model was trained for 7 epochs using a batch size of 64. Results are below including test loss and confusion matrix (highlighted). This model performed very well on the test set, classifying 97% of models accurately.

```
test loss, test acc: [0.07334686815738678, 0.9700000286102295]

Confusion Matrix
[[982  18]
 [ 42 958]]

Classification Report
              precision    recall  f1-score   support

        cats       0.96      0.98      0.97      1000
        dogs       0.98      0.96      0.97      1000

    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```

```
Epoch 1/7
125/125 [==============================] - 74s 547ms/step - loss: 0.9322 - accuracy: 0.9211 - val_loss:
0.2363 - val_accuracy: 0.9680
Epoch 2/7
125/125 [==============================] - 64s 511ms/step - loss: 0.4715 - accuracy: 0.9359 - val_loss:
0.4816 - val_accuracy: 0.9435
Epoch 3/7
125/125 [==============================] - 64s 513ms/step - loss: 0.3822 - accuracy: 0.9396 - val_loss:
0.3563 - val_accuracy: 0.9375
Epoch 4/7
125/125 [==============================] - 66s 525ms/step - loss: 0.3162 - accuracy: 0.9414 - val_loss:
0.1400 - val_accuracy: 0.9675
Epoch 5/7
125/125 [==============================] - 67s 536ms/step - loss: 0.2596 - accuracy: 0.9401 - val_loss:
0.1459 - val_accuracy: 0.9590
Epoch 6/7
125/125 [==============================] - 69s 549ms/step - loss: 0.1443 - accuracy: 0.9499 - val_loss:
0.0737 - val_accuracy: 0.9690
Epoch 7/7
125/125 [==============================] - 71s 570ms/step - loss: 0.1328 - accuracy: 0.9493 - val_loss:
0.0733 - val_accuracy: 0.9700
1/1 [==============================] - 14s 14s/step - loss: 0.0733 - accuracy: 0.9700
```

4c. Evaluate another transfer model using a sub-network of the first. For my model I kept all parameters the same, and used up to the 15th convolutional layer as a transfer head. As can be seen below, for the same 7 epochs of previous tests the model this time converges at 60% accuracy instead of 97% with a heavy bias towards identifying most images as a dog. The test loss reported was 0.6288089156150818. These results make sense given a slice was taken from an already well performing network.

```
test loss, test acc: [0.6288089156150818, 0.6360000371932983]

Confusion Matrix
[[978  22]
 [706 294]]

Classification Report
              precision    recall  f1-score   support

        cats       0.58      0.98      0.73      1000
        dogs       0.93      0.29      0.45      1000

    accuracy                           0.64      2000
   macro avg       0.76      0.64      0.59      2000
weighted avg       0.76      0.64      0.59      2000
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 150, 150, 3  0           []
                                )]

 conv2d (Conv2D)                (None, 74, 74, 32)   864         ['input_1[0][0]']

 batch_normalization (BatchNorm  (None, 74, 74, 32)  96          ['conv2d[0][0]']
 alization)

 activation (Activation)        (None, 74, 74, 32)   0           ['batch_normalization[0][0]']
```

```
conv2d_1 (Conv2D)              (None, 72, 72, 32)   9216        ['activation[0][0]']

batch_normalization_1 (BatchNo  (None, 72, 72, 32)  96          ['conv2d_1[0][0]']
rmalization)

activation_1 (Activation)      (None, 72, 72, 32)   0           ['batch_normalization_1[0][0]']

conv2d_2 (Conv2D)              (None, 72, 72, 64)   18432       ['activation_1[0][0]']

batch_normalization_2 (BatchNo  (None, 72, 72, 64)  192         ['conv2d_2[0][0]']
rmalization)

activation_2 (Activation)      (None, 72, 72, 64)   0           ['batch_normalization_2[0][0]']

max_pooling2d (MaxPooling2D)   (None, 35, 35, 64)   0           ['activation_2[0][0]']

conv2d_3 (Conv2D)              (None, 35, 35, 80)   5120        ['max_pooling2d[0][0]']

batch_normalization_3 (BatchNo  (None, 35, 35, 80)  240         ['conv2d_3[0][0]']
rmalization)

activation_3 (Activation)      (None, 35, 35, 80)   0           ['batch_normalization_3[0][0]']

conv2d_4 (Conv2D)              (None, 33, 33, 192)  138240      ['activation_3[0][0]']

batch_normalization_4 (BatchNo  (None, 33, 33, 192)  576        ['conv2d_4[0][0]']
rmalization)

activation_4 (Activation)      (None, 33, 33, 192)  0           ['batch_normalization_4[0][0]']

max_pooling2d_1 (MaxPooling2D)  (None, 16, 16, 192)  0          ['activation_4[0][0]']

conv2d_8 (Conv2D)              (None, 16, 16, 64)   12288       ['max_pooling2d_1[0][0]']

batch_normalization_8 (BatchNo  (None, 16, 16, 64)  192         ['conv2d_8[0][0]']
rmalization)

activation_8 (Activation)      (None, 16, 16, 64)   0           ['batch_normalization_8[0][0]']

conv2d_6 (Conv2D)              (None, 16, 16, 48)   9216        ['max_pooling2d_1[0][0]']
```

```
conv2d_9 (Conv2D)              (None, 16, 16, 96)   55296       ['activation_8[0][0]']

batch_normalization_6 (BatchNo  (None, 16, 16, 48)  144         ['conv2d_6[0][0]']
rmalization)

batch_normalization_9 (BatchNo  (None, 16, 16, 96)  288         ['conv2d_9[0][0]']
rmalization)

activation_6 (Activation)      (None, 16, 16, 48)   0           ['batch_normalization_6[0][0]']

activation_9 (Activation)      (None, 16, 16, 96)   0           ['batch_normalization_9[0][0]']

average_pooling2d (AveragePool  (None, 16, 16, 192)  0          ['max_pooling2d_1[0][0]']
ing2D)

conv2d_5 (Conv2D)              (None, 16, 16, 96)   18432       ['max_pooling2d_1[0][0]']

conv2d_7 (Conv2D)              (None, 16, 16, 64)   76800       ['activation_6[0][0]']

conv2d_10 (Conv2D)             (None, 16, 16, 96)   82944       ['activation_9[0][0]']

conv2d_11 (Conv2D)             (None, 16, 16, 64)   12288       ['average_pooling2d[0][0]']

batch_normalization_5 (BatchNo  (None, 16, 16, 96)  288         ['conv2d_5[0][0]']
rmalization)

batch_normalization_7 (BatchNo  (None, 16, 16, 64)  192         ['conv2d_7[0][0]']
rmalization)

batch_normalization_10 (BatchN  (None, 16, 16, 96)  288         ['conv2d_10[0][0]']
ormalization)

batch_normalization_11 (BatchN  (None, 16, 16, 64)  192         ['conv2d_11[0][0]']
ormalization)

activation_5 (Activation)      (None, 16, 16, 96)   0           ['batch_normalization_5[0][0]']

activation_7 (Activation)      (None, 16, 16, 64)   0           ['batch_normalization_7[0][0]']

activation_10 (Activation)     (None, 16, 16, 96)   0           ['batch_normalization_10[0][0]']
```

```
 activation_11 (Activation)     (None, 16, 16, 64)   0           ['batch_normalization_11[0][0]']

 mixed_5b (Concatenate)         (None, 16, 16, 320)  0           ['activation_5[0][0]',
                                                                  'activation_7[0][0]',
                                                                  'activation_10[0][0]',
                                                                  'activation_11[0][0]']

 conv2d_15 (Conv2D)             (None, 16, 16, 32)   10240       ['mixed_5b[0][0]']

==================================================================================================
Total params: 452,160
Trainable params: 450,304
Non-trainable params: 1,856

Epoch 1/7
125/125 [==============================] - 37s 289ms/step - loss: 1.2744 - accuracy: 0.6345 - val_loss: 0.8572 -
val_accuracy: 0.5770
Epoch 2/7
125/125 [==============================] - 32s 254ms/step - loss: 0.7270 - accuracy: 0.6626 - val_loss: 0.5215 -
val_accuracy: 0.7405
Epoch 3/7
125/125 [==============================] - 32s 257ms/step - loss: 0.5924 - accuracy: 0.7005 - val_loss: 0.8128 -
val_accuracy: 0.5650
Epoch 4/7
125/125 [==============================] - 33s 263ms/step - loss: 0.5952 - accuracy: 0.6971 - val_loss: 1.2710 -
val_accuracy: 0.5040
Epoch 5/7
125/125 [==============================] - 32s 251ms/step - loss: 0.5840 - accuracy: 0.7046 - val_loss: 0.6503 -
val_accuracy: 0.6415
Epoch 6/7
125/125 [==============================] - 31s 251ms/step - loss: 0.5711 - accuracy: 0.7093 - val_loss: 0.6391 -
val_accuracy: 0.6480
Epoch 7/7
125/125 [==============================] - 31s 249ms/step - loss: 0.5750 - accuracy: 0.7016 - val_loss: 0.6288 -
val_accuracy: 0.6360
1/1 [==============================] - 4s 4s/step - loss: 0.6288 - accuracy: 0.6360
```