Vivek Ch. Nair Ms.

Fall 2020

CS543: Neural Networks

Project 3: Transfer Learning

```
In [1]: import os
        import numpy as np
        import tensorflow as tf
        import tensorflow.keras as keras
        from tensorflow.keras.preprocessing.image import ImageDataGenerator ##
        ##
        ##
        
```

```
block_3_project_BN (BatchNorma (None, 12, 12, 96)  384     block_3_project_Conv2D[0][0]
block_3_add (Add)               (None, 12, 12, 96)  0       block_2_add[0][0]
                                                            block_3_project_BN[0][0]

block_4_expand (Conv2D)         (None, 12, 12, 192) 6144    block_3_add[0][0]

block_4_expand_BN (BatchNormali (None, 12, 12, 192) 768     block_4_expand[0][0]

block_4_expand_relu (ReLU)      (None, 12, 12, 192) 0       block_4_expand_BN[0][0]

block_4_pad (ZeroPadding2D)     (None, 13, 13, 192) 0       block_4_expand_relu[0][0]

block_4_depthwise (DepthwiseCon (None, 6, 6, 192)   1728    block_4_pad[0][0]

block_4_depthwise_BN (BatchNorm (None, 6, 6, 192)   768     block_4_depthwise[0][0]

block_4_depthwise_relu (ReLU)   (None, 6, 6, 192)   0       block_4_depthwise_BN[0][0]

block_4_project (Conv2D)        (None, 6, 6, 64)    12288   block_4_depthwise_relu[0][0]

block_4_project_BN (BatchNorma  (None, 6, 6, 64)    256     block_4_project[0][0]

block_5_expand (Conv2D)         (None, 6, 6, 384)   24576   block_4_project_BN[0][0]

block_5_expand_BN (BatchNormali (None, 6, 6, 384)   1536    block_5_expand[0][0]

block_5_expand_relu (ReLU)      (None, 6, 6, 384)   0       block_5_expand_BN[0][0]

block_5_depthwise (DepthwiseCon (None, 6, 6, 384)   3456    block_5_expand_relu[0][0]

block_5_depthwise_BN (BatchNorm (None, 6, 6, 384)   1536    block_5_depthwise[0][0]

block_5_depthwise_relu (ReLU)   (None, 6, 6, 384)   0       block_5_depthwise_BN[0][0]

block_5_project (Conv2D)        (None, 6, 6, 64)    24576   block_5_depthwise_relu[0][0]

block_5_project_BN (BatchNorma  (None, 6, 6, 64)    256     block_5_project[0][0]

block_5_add (Add)               (None, 6, 6, 64)    0       block_4_project_BN[0][0]
                                                            block_5_project_BN[0][0]

block_6_expand (Conv2D)         (None, 6, 6, 384)   24576   block_5_add[0][0]

block_6_expand_BN (BatchNormali (None, 6, 6, 384)   1536    block_6_expand[0][0]

block_6_expand_relu (ReLU)      (None, 6, 6, 384)   0       block_6_expand_BN[0][0]

block_6_depthwise (DepthwiseCon (None, 6, 6, 384)   3456    block_6_expand_relu[0][0]

block_6_depthwise_BN (BatchNorm (None, 6, 6, 384)   1536    block_6_depthwise[0][0]

block_6_depthwise_relu (ReLU)   (None, 6, 6, 384)   0       block_6_depthwise_BN[0][0]

block_6_project (Conv2D)        (None, 6, 6, 64)    24576   block_6_depthwise_relu[0][0]

block_6_project_BN (BatchNorma  (None, 6, 6, 64)    256     block_6_project[0][0]

block_6_add (Add)               (None, 6, 6, 64)    0       block_5_add[0][0]
                                                            block_6_project_BN[0][0]

block_7_expand (Conv2D)         (None, 6, 6, 384)   24576   block_6_add[0][0]

block_7_expand_BN (BatchNormali (None, 6, 6, 384)   1536    block_7_expand[0][0]

block_7_expand_relu (ReLU)      (None, 6, 6, 384)   0       block_7_expand_BN[0][0]

block_7_depthwise (DepthwiseCon (None, 6, 6, 384)   3456    block_7_expand_relu[0][0]

block_7_depthwise_BN (BatchNorm (None, 6, 6, 384)   1536    block_7_depthwise[0][0]

block_7_depthwise_relu (ReLU)   (None, 6, 6, 384)   0       block_7_depthwise_BN[0][0]

block_7_project (Conv2D)        (None, 6, 6, 64)    24576   block_7_depthwise_relu[0][0]

block_7_project_BN (BatchNorma  (None, 6, 6, 64)    256     block_7_project[0][0]

block_7_add (Add)               (None, 6, 6, 64)    0       block_6_project_BN[0][0]
                                                            block_7_project_BN[0][0]

block_8_expand (Conv2D)         (None, 6, 6, 384)   24576   block_7_add[0][0]

block_8_expand_BN (BatchNormali (None, 6, 6, 384)   1536    block_8_expand[0][0]

block_8_expand_relu (ReLU)      (None, 6, 6, 384)   0       block_8_expand_BN[0][0]

block_8_depthwise (DepthwiseCon (None, 6, 6, 384)   3456    block_8_expand_relu[0][0]

block_8_depthwise_BN (BatchNorm (None, 6, 6, 384)   1536    block_8_depthwise[0][0]

block_8_depthwise_relu (ReLU)   (None, 6, 6, 384)   0       block_8_depthwise_BN[0][0]

block_8_project (Conv2D)        (None, 6, 6, 64)    24576   block_8_depthwise_relu[0][0]

block_8_project_BN (BatchNorma  (None, 6, 6, 64)    256     block_8_project[0][0]

block_8_add (Add)               (None, 6, 6, 64)    0       block_7_add[0][0]
                                                            block_8_project_BN[0][0]

block_9_expand (Conv2D)         (None, 6, 6, 384)   24576   block_8_add[0][0]

block_9_expand_BN (BatchNormali (None, 6, 6, 384)   1536    block_9_expand[0][0]

block_9_expand_relu (ReLU)      (None, 6, 6, 384)   0       block_9_expand_BN[0][0]

block_9_depthwise (DepthwiseCon (None, 6, 6, 384)   3456    block_9_expand_relu[0][0]

block_9_depthwise_BN (BatchNorm (None, 6, 6, 384)   1536    block_9_depthwise[0][0]

block_9_depthwise_relu (ReLU)   (None, 6, 6, 384)   0       block_9_depthwise_BN[0][0]

block_9_project (Conv2D)        (None, 6, 6, 64)    24576   block_9_depthwise_relu[0][0]

block_9_project_BN (BatchNorma  (None, 6, 6, 64)    256     block_9_project[0][0]

block_9_add (Add)               (None, 6, 6, 64)    0       block_8_add[0][0]
                                                            block_9_project_BN[0][0]

block_10_expand (Conv2D)        (None, 6, 6, 384)   24576   block_9_add[0][0]

block_10_expand_BN (BatchNormal (None, 6, 6, 384)   1536    block_10_expand[0][0]

block_10_expand_relu (ReLU)     (None, 6, 6, 384)   0       block_10_expand_BN[0][0]

block_10_depthwise (DepthwiseCo (None, 6, 6, 384)   3456    block_10_expand_relu[0][0]

block_10_depthwise_BN (BatchNor (None, 6, 6, 384)   1536    block_10_depthwise[0][0]

block_10_depthwise_relu (ReLU)  (None, 6, 6, 384)   0       block_10_depthwise_BN[0][0]

block_10_project (Conv2D)       (None, 6, 6, 96)    36864   block_10_depthwise_relu[0][0]

block_10_project_BN (BatchNorma (None, 6, 6, 96)    384     block_10_project[0][0]

block_11_expand (Conv2D)        (None, 6, 6, 576)   55296   block_10_project_BN[0][0]

block_11_expand_BN (BatchNormali (None, 6, 6, 576)  2304    block_11_expand[0][0]

block_11_expand_relu (ReLU)     (None, 6, 6, 576)   0       block_11_expand_BN[0][0]

block_11_depthwise (DepthwiseCon (None, 6, 6, 576)  5184    block_11_expand_relu[0][0]

block_11_depthwise_BN (BatchNor (None, 6, 6, 576)   2304    block_11_depthwise[0][0]

block_11_depthwise_relu (ReLU)  (None, 6, 6, 576)   0       block_11_depthwise_BN[0][0]

block_11_project (Conv2D)       (None, 6, 6, 96)    55296   block_11_depthwise_relu[0][0]

block_11_project_BN (BatchNorma (None, 6, 6, 96)    384     block_11_project[0][0]

block_11_add (Add)              (None, 6, 6, 96)    0       block_10_project_BN[0][0]
                                                            block_11_project_BN[0][0]

block_12_expand (Conv2D)        (None, 6, 6, 576)   55296   block_11_add[0][0]

block_12_expand_BN (BatchNormali (None, 6, 6, 576)  2304    block_12_expand[0][0]

block_12_expand_relu (ReLU)     (None, 6, 6, 576)   0       block_12_expand_BN[0][0]

block_12_depthwise (DepthwiseCo (None, 6, 6, 576)   5184    block_12_expand_relu[0][0]

block_12_depthwise_BN (BatchNor (None, 6, 6, 576)   2304    block_12_depthwise[0][0]

block_12_depthwise_relu (ReLU)  (None, 6, 6, 576)   0       block_12_depthwise_BN[0][0]

block_12_project (Conv2D)       (None, 6, 6, 96)    55296   block_12_depthwise_relu[0][0]

block_12_project_BN (BatchNorma (None, 6, 6, 96)    384     block_12_project[0][0]

block_12_add (Add)              (None, 6, 6, 96)    0       block_11_add[0][0]
                                                            block_12_project_BN[0][0]

block_13_expand (Conv2D)        (None, 6, 6, 576)   55296   block_12_add[0][0]

block_13_expand_BN (BatchNormali (None, 6, 6, 576)  2304    block_13_expand[0][0]

block_13_expand_relu (ReLU)     (None, 6, 6, 576)   0       block_13_expand_BN[0][0]

block_13_pad (ZeroPadding2D)    (None, 7, 7, 576)   0       block_13_expand_relu[0][0]

block_13_depthwise (DepthwiseCo (None, 3, 3, 576)   5184    block_13_pad[0][0]

block_13_depthwise_BN (BatchNor (None, 3, 3, 576)   2304    block_13_depthwise[0][0]

block_13_depthwise_relu (ReLU)  (None, 3, 3, 576)   0       block_13_depthwise_BN[0][0]

block_13_project (Conv2D)       (None, 3, 3, 160)   92160   block_13_depthwise_relu[0][0]

block_13_project_BN (BatchNorma (None, 3, 3, 160)   640     block_13_project[0][0]

block_14_expand (Conv2D)        (None, 3, 3, 960)   153600  block_13_project_BN[0][0]

block_14_expand_BN (BatchNormali (None, 3, 3, 960)  3840    block_14_expand[0][0]

block_14_expand_relu (ReLU)     (None, 3, 3, 960)   0       block_14_expand_BN[0][0]

block_14_depthwise (DepthwiseCo (None, 3, 3, 960)   8640    block_14_expand_relu[0][0]

block_14_depthwise_BN (BatchNor (None, 3, 3, 960)   3840    block_14_depthwise[0][0]

block_14_depthwise_relu (ReLU)  (None, 3, 3, 960)   0       block_14_depthwise_BN[0][0]

block_14_project (Conv2D)       (None, 3, 3, 160)   153600  block_14_depthwise_relu[0][0]

block_14_project_BN (BatchNorma (None, 3, 3, 160)   640     block_14_project[0][0]

block_14_add (Add)              (None, 3, 3, 160)   0       block_13_project_BN[0][0]
                                                            block_14_project_BN[0][0]

block_15_expand (Conv2D)        (None, 3, 3, 960)   153600  block_14_add[0][0]

block_15_expand_BN (BatchNormali (None, 3, 3, 960)  3840    block_15_expand[0][0]

block_15_expand_relu (ReLU)     (None, 3, 3, 960)   0       block_15_expand_BN[0][0]

block_15_depthwise (DepthwiseCo (None, 3, 3, 960)   8640    block_15_expand_relu[0][0]

block_15_depthwise_BN (BatchNor (None, 3, 3, 960)   3840    block_15_depthwise[0][0]

block_15_depthwise_relu (ReLU)  (None, 3, 3, 960)   0       block_15_depthwise_BN[0][0]

block_15_project (Conv2D)       (None, 3, 3, 160)   153600  block_15_depthwise_relu[0][0]

block_15_project_BN (BatchNorma (None, 3, 3, 160)   640     block_15_project[0][0]

block_15_add (Add)              (None, 3, 3, 160)   0       block_14_add[0][0]
                                                            block_15_project_BN[0][0]

block_16_expand (Conv2D)        (None, 3, 3, 960)   153600  block_15_add[0][0]

block_16_expand_BN (BatchNormali (None, 3, 3, 960)  3840    block_16_expand[0][0]

block_16_expand_relu (ReLU)     (None, 3, 3, 960)   0       block_16_expand_BN[0][0]

block_16_depthwise (DepthwiseCo (None, 3, 3, 960)   8640    block_16_expand_relu[0][0]

block_16_depthwise_BN (BatchNor (None, 3, 3, 960)   3840    block_16_depthwise[0][0]

block_16_depthwise_relu (ReLU)  (None, 3, 3, 960)   0       block_16_depthwise_BN[0][0]

block_16_project (Conv2D)       (None, 3, 3, 320)   307200  block_16_depthwise_relu[0][0]

block_16_project_BN (BatchNorma (None, 3, 3, 320)   1280    block_16_project[0][0]

Conv_1 (Conv2D)                 (None, 3, 3, 1280)  409600  block_16_project_BN[0][0]

Conv_1_bn (BatchNormalization)  (None, 3, 3, 1280)  5120    Conv_1[0][0]

out_relu (ReLU)                 (None, 3, 3, 1280)  0       Conv_1_bn[0][0]
==================================================================================================
Total params: 2,257,984
Trainable params: 0
Non-trainable params: 2,257,984
```

2e. Create augmented model

Create a new `base_augmented` augmented model with an output layer that has the correct number of units to deal with the hot-dog or not problem

Compile the augmented model with:

- Adam optimizer (learning rate of 0.0001)
- binary_crossentropy loss
- accuracy metric.

Here is the final part of the summary from the augmented model that you should get:

```
Total params: 2,259,505
Trainable params: 11,521
Non-trainable params: 2,257,984
```

Helpful links

https://www.tensorflow.org/api_docs/keras/layers/Dense https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam https://www.tensorflow.org/api_docs/python/tf/losses

```
In [64]: avg_model = tf.keras.Sequential(
             [
                 model,
                 tf.keras.layers.Flatten(),
                 tf.keras.layers.Dense(units=1, activation='sigmoid')
             ]
         )

         avg_model.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_96 (Functio (None, 3, 3, 1280)        2257984
_________________________________________________________________
flatten_17 (Flatten)         (None, 11520)             0
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 11521
=================================================================
Total params: 2,269,505
Trainable params: 11,521
Non-trainable params: 2,257,984
_________________________________________________________________
```

```
In [65]: adam = tf.optimizers.Adam(lr=0.0001)
         avg_model.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])
```

2f. Questions

Question 4: What is the accuracy and loss for the network with the untrained output layer on the test set? Explain why you got the accuracy value that you did.

Question 5: Briefly defend your choice of number of units in the output layer.

Answer 4:
The loss for the test set is 0.87.
The accuracy for the test set is 52.6%.
The accuracy is moderate. Given that the images in this dataset are all photos, and we are doing a binary classification, it is easy to have an "initial" accuracy around 50%.

Answer 5:
We only need one unit in the output layer (very similar to the SLP) because it is a binary classification, and it is a yes-or-no question.

```
In [66]: test_loss, test_acc = avg_model.evaluate(hotdog_test_x, hotdog_test_y, verbose=1)
         print("Test loss: {test_loss}, accuracy: {test_acc}")
```

```
Test loss: 0.871276139917675, accuracy: 0.526499966669559
```

2g. Fit the augmented model on the hotdog training data

Setting the verbose optional parameter to 2 will give you helpful printouts of performance on the validation set as it completes every epoch of training.

Training goal

You should aim to achieve at least 75% accuracy on the validation set.

Note

If training time is taking much more than a few minutes per epoch on your computer, you could try reducing the number of data samples in train and validation. For example, by default train `X` = `[:2800]`. Try `X` = `X[:400]` instead. You could do the same for the validation set.

```
In [73]: hist = avg_model.fit(hotdog_train_x, hotdog_train_y, batch_size=5, epochs=25, validation_data=(hotdog_val_x, hotdog_v
         al_y), verbose=1)
```

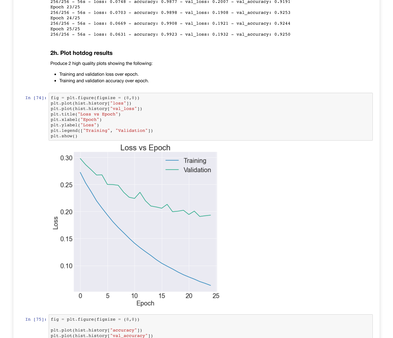
```
Epoch 1/25
256/256 - 55s - loss: 0.2726 - accuracy: 0.8838 - val_loss: 0.2963 - val_accuracy: 0.8703
Epoch 2/25
256/256 - 53s - loss: 0.2527 - accuracy: 0.8953 - val_loss: 0.2867 - val_accuracy: 0.8756
Epoch 3/25
256/256 - 54s - loss: 0.2368 - accuracy: 0.9041 - val_loss: 0.2775 - val_accuracy: 0.8809
Epoch 4/25
256/256 - 54s - loss: 0.2139 - accuracy: 0.9134 - val_loss: 0.2676 - val_accuracy: 0.8825
Epoch 5/25
256/256 - 54s - loss: 0.2064 - accuracy: 0.9209 - val_loss: 0.2679 - val_accuracy: 0.8866
Epoch 6/25
256/256 - 54s - loss: 0.1936 - accuracy: 0.9245 - val_loss: 0.2501 - val_accuracy: 0.8953
Epoch 7/25
256/256 - 54s - loss: 0.1812 - accuracy: 0.9359 - val_loss: 0.2499 - val_accuracy: 0.8888
Epoch 8/25
256/256 - 54s - loss: 0.1706 - accuracy: 0.9415 - val_loss: 0.2486 - val_accuracy: 0.8916
Epoch 9/25
256/256 - 54s - loss: 0.1606 - accuracy: 0.9460 - val_loss: 0.2356 - val_accuracy: 0.8988
Epoch 10/25
256/256 - 54s - loss: 0.1506 - accuracy: 0.9527 - val_loss: 0.2264 - val_accuracy: 0.9038
Epoch 11/25
256/256 - 55s - loss: 0.1412 - accuracy: 0.9555 - val_loss: 0.2242 - val_accuracy: 0.9031
Epoch 12/25
256/256 - 54s - loss: 0.1332 - accuracy: 0.9617 - val_loss: 0.2156 - val_accuracy: 0.9091
Epoch 13/25
256/256 - 55s - loss: 0.1257 - accuracy: 0.9634 - val_loss: 0.2159 - val_accuracy: 0.9125
Epoch 14/25
256/256 - 55s - loss: 0.1186 - accuracy: 0.9675 - val_loss: 0.2102 - val_accuracy: 0.9128
Epoch 15/25
256/256 - 55s - loss: 0.1109 - accuracy: 0.9724 - val_loss: 0.2050 - val_accuracy: 0.9153
Epoch 16/25
256/256 - 55s - loss: 0.1042 - accuracy: 0.9755 - val_loss: 0.2041 - val_accuracy: 0.9147
Epoch 17/25
256/256 - 55s - loss: 0.0987 - accuracy: 0.9774 - val_loss: 0.2113 - val_accuracy: 0.9153
Epoch 18/25
256/256 - 55s - loss: 0.0935 - accuracy: 0.9795 - val_loss: 0.1996 - val_accuracy: 0.9197
Epoch 19/25
256/256 - 55s - loss: 0.0878 - accuracy: 0.9812 - val_loss: 0.2006 - val_accuracy: 0.9197
Epoch 20/25
256/256 - 55s - loss: 0.0829 - accuracy: 0.9850 - val_loss: 0.2023 - val_accuracy: 0.9159
Epoch 21/25
256/256 - 55s - loss: 0.0787 - accuracy: 0.9861 - val_loss: 0.1945 - val_accuracy: 0.9219
Epoch 22/25
256/256 - 54s - loss: 0.0748 - accuracy: 0.9877 - val_loss: 0.2007 - val_accuracy: 0.9191
Epoch 23/25
256/256 - 54s - loss: 0.0703 - accuracy: 0.9898 - val_loss: 0.1908 - val_accuracy: 0.9253
Epoch 24/25
256/256 - 54s - loss: 0.0669 - accuracy: 0.9909 - val_loss: 0.1921 - val_accuracy: 0.9244
Epoch 25/25
256/256 - 54s - loss: 0.0631 - accuracy: 0.9923 - val_loss: 0.1932 - val_accuracy: 0.9250
```

2h. Plot hotdog results

Produce 2 high quality plots showing the following:

- Training and validation loss over epoch.
- Training and validation accuracy over epoch.

```
In [76]: fig = plt.figure(figsize = (7,5))
         plt.plot(hist.history['loss'])
         plt.plot(hist.history['val_loss'])
         plt.title('Loss vs Epoch')
         plt.xlabel('Epoch')
         plt.ylabel('Loss')
         plt.legend(['Training', 'Validation'])
         plt.show()
```




```
In [75]: fig = plt.figure(figsize = (7,5))
         plt.plot(hist.history['accuracy'])
         plt.plot(hist.history['val_accuracy'])
         plt.title('Accuracy vs Epoch')
         plt.xlabel('Epoch')
         plt.ylabel('Accuracy')
         plt.legend(['Training', 'Validation'])
         plt.show()
```

2i. Questions

Question 6: What is the best accuracy that you are able to achieve on the test set? Briefly summarize the hyperparameters that were used in your model.

Answer 6:

accuracy on the test set: 81%
batch size: 50
n_epochs: 25

```
In [76]: test_loss, test_acc = avg_model.evaluate(hotdog_test_x, hotdog_test_y, verbose=1)
         print("Test loss: {test_loss}, accuracy: {test_acc}")
```

```
Test loss: 0.563432012690734, accuracy: 0.810000002384185
```

Week 5b, Notes 5b

Fall 2020

CS230: Deep Neural Networks

Prepar 2 Transfer Learning

Backmatter reminder (Do you answer all 3 questions?

```
# clone the repo
!git clone -q --depth 1 -b master -q https://github.com/lisacorrigan/CS230-Transfer-Learning.git
!rm -rf sample_datasets/*
!rm -rf models/*
!rm -rf results/*
!rm -rf .git*
!rm -rf README.md
!rm -rf LICENSE
!rm -rf requirements.txt
!rm -rf .gitignore
!ls -lhtrsa .
!ls -lhtrsa sample_datasets/
!ls -lhtrsa models/
!ls -lhtrsa results/
```

```
total 20K
 4K drwxr-xr-x  1 root root 4096 Nov  2 19:54 .
 4K drwxr-xr-x  1 root root 4096 Nov  2 19:54 sample_datasets/
 4K drwxr-xr-x  1 root root 4096 Nov  2 19:54 models/
 4K drwxr-xr-x  1 root root 4096 Nov  2 19:54 results/
 4K drwxr-xr-x  1 root root 4096 Nov  2 19:54 ..
```

Task 3: DeepDream

You will make use of a pre-trained Inception network (TensorFlow) to implement the **DeepDream symbolic gradient** algorithm using TensorFlow 2. The algorithm projects the expected Gaussian pixel values in the input image to maximize the output of certain neurons in the network. It uses gradient ascent to find an input image that maximizes the activations of certain neurons in the network. Gradient ascent is the same as gradient descent except we add the gradient to the input image instead of subtracting it.

This task will expose you to TensorFlow's low-level API that provides a lot more control than Keras does.

Overview

- Load in the VGG19 pre-trained network without the output layer
- Choose layers from the network to use for the loss function (we will use layers from the original VGG19 network)
- Define the loss of layers (e.g., conv, pooling, etc) that we want to use to calculate the gradient. Use Gaussian to influence the input image
- Calculate the gradient of the loss function with respect to the input image using TensorFlow's API
- Do gradient ascent until reach a predefined number of iter gradient ascent steps or reach the target or some number of deviations.

3a. Import mountain image

Run the below cells to load in the mountain image (or 224 x 224 resolution) The notebook the resolution of images on which VGG19 was trained.

Ifyou have a more powerful machine or are willing to wait longer for runtime purposes, you can use the provided code on its images.

```
img_filename = 'sample_datasets/mtn.jpg'
!ls -lhtrsa sample_datasets/mtn.jpg
!file sample_datasets/mtn.jpg
!pip install -q -U pillow
import PIL.Image, os.path
import numpy as np
print("image shape is:", img.shape)
```

```
4K -rw-r--r--  1 root root  3.9K Nov  2 19:54 sample_datasets/mtn.jpg
sample_datasets/mtn.jpg: JPEG image data, JFIF standard 1.01, resolution defined, 1x1, 1x1
     |[K
     |[K
     |[K
```

```
epoch [ 4/5]
Image Shape: [2, 5,0,0,3, 3] obtained 0.5-0,3
```

Image of the combined tensor corresponding to concatenation.

```
epoch [ 5/5]
Image Shape: [2, 5,0,0,3, 3] obtained 0.5-0,3
```

Image of the other combined tensor corresponding to concatenation.

a.3a) (3)

```
epoch [ 1/5] [ 1/ 1] Generatorloss: 1.2662
Image : Shape[batch,w,h,c]: [batch=2,128,128,3] [c1,c2,c3,c4,c5,c6,c7,c8]: [128,64,32,16,8,4,2,1]
epoch [ 2/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 3/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 4/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 5/5] [ 1/ 1] Generatorloss: 1.2662
```

a.3a) (4)

```
epoch [ 1/5] [ 1/ 1] Generatorloss: 1.2662
Image : Shape[batch,w,h,c]: [batch=2,128,128,3] [c1,c2,c3,c4,c5,c6,c7,c8]: [128,64,32,16,8,4,2,1]
epoch [ 2/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 3/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 4/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 5/5] [ 1/ 1] Generatorloss: 1.2662
```

The visual patterns are much more evident compared to the other image.

Often the number of features c1, c2, c3 ... c8 are, especially for new runs, a little 2logp.

2. Research other pretrained networks in tensorflow and run DeepDream on their layers/filters.

I used a mixed5b layer out of the inceptionv3 model.

```
!pip install keras
```

```
pip install -q -U keras-preprocessing
Collecting keras-preprocessing
  Downloading https://files.pythonhosted.org/packages/ed/ca/a8a7ebb4a0e690cb9a56621d8a2b2f1e9ca0a4d764a6616f8778a2a6b0d2/Keras_Preprocessing-1.1.0-py2.py3-none-any.whl (42kB)
     |100%| 51kB, 2.8MB/s, ETA 0:00:00
Installing collected packages: keras-preprocessing
Successfully installed keras-preprocessing-1.1.0
```

```
import numpy
from keras.applications import InceptionV3
from keras.preprocessing.image import load_img, save_img
model = InceptionV3(weights='imagenet',
                    include_top=False)
layer_contributions = {
    'mixed2': 0.2,
    'mixed3': 0.5,
    'mixed4': 2.,
    'mixed5': 1.5,
}
```

```
Using TensorFlow backend.
Downloading data from https://github.com/fchollet/keras-models/release/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) in layer-9. Expected shape (None, 299, 299, 3) in layer-9. Things may perform oddly.
layer_name, coeff, shape: ('mixed2', 0.2, (None, None, None, 288))
layer_name, coeff, shape: ('mixed3', 0.5, (None, None, None, 768))
layer_name, coeff, shape: ('mixed4', 2.0, (None, None, None, 768))
layer_name, coeff, shape: ('mixed5', 1.5, (None, None, None, 768))
```

```
layer_dict = dict(layer.name, layer)
print('Layer contributions: ', layer_contributions)
print('Layer dict: ', layer_dict)
```

```
Layer contributions:  {'mixed2': 0.2, 'mixed3': 0.5, 'mixed4': 2.0, 'mixed5': 1.5}
Layer dict:  {'mixed2': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed3': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed4': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed5': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>}
```

```
loss = K.variable(0.)
for layer_name in layer_contributions:
    coeff = layer_contributions[layer_name]
    activation = layer_dict[layer_name].output
    scaling = K.prod(K.cast(K.shape(activation), 'float32'))
    loss = loss + coeff * K.sum(K.square(activation[:, 2: -2, 2: -2, :])) / scaling
```

```
Instructions for updating:
Use tf.cast instead.
```

```
dream = model.input
grad = K.gradients(dream, loss)[0]
grad = grad / K.maximum(K.mean(K.abs(grad)), 1e-7)
outputs = [loss, grad]
fetch_loss_and_grads = K.function([dream], outputs)
def eval_loss_and_grads(x):
    outs = fetch_loss_and_grads([x])
    loss_value = outs[0]
    grad_values = outs[1]
    return loss_value, grad_values
def resize_img(img, size):
    img = np.copy(img)
    factor = (float(size[1]) / img.shape[1],
              float(size[2]) / img.shape[2])
    return scipy.ndimage.zoom(img, (1, factor[0], factor[1], 1),
                              order=0)
def gradient_ascent(x, iterations, step, max_loss=None):
    for i in range(iterations):
        print('...Loss value at', i, ':', loss_value)
        x *= (1 - 127.5) / 255.
        x += step * grad_values
        if max_loss is not None:
            save_img('/tmp/stopped.jpg', x)
            print('Stopping at iteration', i, 'due to exceeding max_loss')
            break
        else:
            print('Saving image at iteration', i)
    return x
step = 0.01
num_octave = 3
octave_scale = 1.4
iterations = 20
max_loss = None
base_image_path = '/tmp/base.jpg'
base_image = scipy.misc.imread(base_image_path)
original_shape = base_image.shape[:2]
print('Original shape: ', original_shape)
for shape in successive_shapes:
    print('Processing image shape', shape)
    base_image = resize_img(base_image, shape)
    img = gradient_ascent(img,
                          iterations=iterations,
                          step=step,
                          max_loss=max_loss)
    upscaled_shape = shape[:2]
    lost_detail = np.copy(base_image) - resize_img(upscaled_shape,
                                                   base_image)
    print('Saving image at iteration', i, 'due to exceeding max_loss')
    break
    else:
    print('Saving image at iteration', i)
return x
```

```
Using TensorFlow backend.
Downloading data from https://github.com/fchollet/keras-models/release/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) in layer-9. Expected shape (None, 299, 299, 3) in layer-9. Things may perform oddly.
layer_name, coeff, shape: ('mixed2', 0.2, (None, None, None, 288))
layer_name, coeff, shape: ('mixed3', 0.5, (None, None, None, 768))
layer_name, coeff, shape: ('mixed4', 2.0, (None, None, None, 768))
layer_name, coeff, shape: ('mixed5', 1.5, (None, None, None, 768))
Layer contributions:  {'mixed2': 0.2, 'mixed3': 0.5, 'mixed4': 2.0, 'mixed5': 1.5}
Layer dict:  {'mixed2': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed3': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed4': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed5': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>}
Instructions for updating:
Use tf.cast instead.
Original shape:  (1, 5,0,0,3, 3)
Processing image shape (1, 5,0,0,3, 3)
...Loss value at 0 : 1.2662
Saving image at iteration 0
...Loss value at 1 : 1.2662
Saving image at iteration 1
...Loss value at 2 : 1.2662
Saving image at iteration 2
...Loss value at 3 : 1.2662
Saving image at iteration 3
...Loss value at 4 : 1.2662
Saving image at iteration 4
...Loss value at 5 : 1.2662
Saving image at iteration 5
...Loss value at 6 : 1.2662
Saving image at iteration 6
...Loss value at 7 : 1.2662
Saving image at iteration 7
...Loss value at 8 : 1.2662
Saving image at iteration 8
...Loss value at 9 : 1.2662
Saving image at iteration 9
...Loss value at 10 : 1.2662
Saving image at iteration 10
...Loss value at 11 : 1.2662
Saving image at iteration 11
...Loss value at 12 : 1.2662
Saving image at iteration 12
...Loss value at 13 : 1.2662
Saving image at iteration 13
...Loss value at 14 : 1.2662
Saving image at iteration 14
...Loss value at 15 : 1.2662
Saving image at iteration 15
...Loss value at 16 : 1.2662
Saving image at iteration 16
...Loss value at 17 : 1.2662
Saving image at iteration 17
...Loss value at 18 : 1.2662
Saving image at iteration 18
...Loss value at 19 : 1.2662
Saving image at iteration 19
Processing image shape (1, 7,0,0,3, 3)
...Loss value at 0 : 1.2662
Saving image at iteration 0
...Loss value at 1 : 1.2662
Saving image at iteration 1
...Loss value at 2 : 1.2662
Saving image at iteration 2
...Loss value at 3 : 1.2662
Saving image at iteration 3
...Loss value at 4 : 1.2662
Saving image at iteration 4
...Loss value at 5 : 1.2662
Saving image at iteration 5
...Loss value at 6 : 1.2662
Saving image at iteration 6
...Loss value at 7 : 1.2662
Saving image at iteration 7
...Loss value at 8 : 1.2662
Saving image at iteration 8
...Loss value at 9 : 1.2662
Saving image at iteration 9
...Loss value at 10 : 1.2662
Saving image at iteration 10
...Loss value at 11 : 1.2662
Saving image at iteration 11
...Loss value at 12 : 1.2662
Saving image at iteration 12
...Loss value at 13 : 1.2662
Saving image at iteration 13
...Loss value at 14 : 1.2662
Saving image at iteration 14
...Loss value at 15 : 1.2662
Saving image at iteration 15
...Loss value at 16 : 1.2662
Saving image at iteration 16
...Loss value at 17 : 1.2662
Saving image at iteration 17
...Loss value at 18 : 1.2662
Saving image at iteration 18
...Loss value at 19 : 1.2662
Saving image at iteration 19
Processing image shape (1, 9,0,0,3, 3)
...Loss value at 0 : 1.2662
Saving image at iteration 0
...Loss value at 1 : 1.2662
Saving image at iteration 1
...Loss value at 2 : 1.2662
Saving image at iteration 2
...Loss value at 3 : 1.2662
Saving image at iteration 3
...Loss value at 4 : 1.2662
Saving image at iteration 4
...Loss value at 5 : 1.2662
Saving image at iteration 5
...Loss value at 6 : 1.2662
Saving image at iteration 6
...Loss value at 7 : 1.2662
Saving image at iteration 7
...Loss value at 8 : 1.2662
Saving image at iteration 8
...Loss value at 9 : 1.2662
Saving image at iteration 9
...Loss value at 10 : 1.2662
Saving image at iteration 10
...Loss value at 11 : 1.2662
Saving image at iteration 11
...Loss value at 12 : 1.2662
Saving image at iteration 12
...Loss value at 13 : 1.2662
Saving image at iteration 13
...Loss value at 14 : 1.2662
Saving image at iteration 14
...Loss value at 15 : 1.2662
Saving image at iteration 15
...Loss value at 16 : 1.2662
Saving image at iteration 16
...Loss value at 17 : 1.2662
Saving image at iteration 17
...Loss value at 18 : 1.2662
Saving image at iteration 18
...Loss value at 19 : 1.2662
Saving image at iteration 19
```

```
!cp /tmp/base.jpg /tmp/base.jpg
!cp /tmp/stopped.jpg /tmp/stopped.jpg
!cp /tmp/base.jpg /tmp/base.jpg
!cp /tmp/stopped.jpg /tmp/stopped.jpg
```

```
cp: cannot stat '/tmp/base.jpg': No such file or directory
cp: cannot stat '/tmp/stopped.jpg': No such file or directory
cp: cannot stat '/tmp/base.jpg': No such file or directory
cp: cannot stat '/tmp/stopped.jpg': No such file or directory
```

```
from google.colab.patches import cv2_imshow
from PIL import Image
from IPIL import display
from matplotlib.pyplot import imshow
from numpy.random.mtrand import uniform
from numpy.random.mtrand import randint
from numpy.random.mtrand import seed
from numpy.random.mtrand import shuffle
from numpy.random.mtrand import choice
from numpy.random.mtrand import gamma
from numpy.random.mtrand import beta
from numpy.random.mtrand import chisquare
from numpy.random.mtrand import exponential
from numpy.random.mtrand import f
from numpy.random.mtrand import gumbel
from numpy.random.mtrand import laplace
from numpy.random.mtrand import logistic
from numpy.random.mtrand import lognormal
from numpy.random.mtrand import logseries
from numpy.random.mtrand import multinomial
from numpy.random.mtrand import multivariate_normal
from numpy.random.mtrand import negative_binomial
from numpy.random.mtrand import noncentral_chisquare
from numpy.random.mtrand import noncentral_f
from numpy.random.mtrand import pareto
from numpy.random.mtrand import poisson
from numpy.random.mtrand import power
from numpy.random.mtrand import rayleigh
from numpy.random.mtrand import standard_cauchy
from numpy.random.mtrand import standard_exponential
from numpy.random.mtrand import standard_gamma
from numpy.random.mtrand import standard_normal
from numpy.random.mtrand import standard_t
from numpy.random.mtrand import triangular
from numpy.random.mtrand import uniform
from numpy.random.mtrand import vonmises
from numpy.random.mtrand import wald
from numpy.random.mtrand import weibull
from numpy.random.mtrand import zipf
from numpy import *
from numpy import ma
from numpy.ma.core import *
from numpy.ma.extras import *
from numpy.lib.arraysetops import ediff1d
from numpy.lib.arraysetops import intersect1d
from numpy.lib.arraysetops import setdiff1d
from numpy.lib.arraysetops import setxor1d
from numpy.lib.arraysetops import union1d
from numpy.lib.arraysetops import unique
from numpy.lib.arraysetops import in1d
from numpy.lib.arraysetops import isin
from numpy.lib.function_base import select
from numpy.lib.function_base import piecewise
from numpy.lib.function_base import trim_zeros
from numpy.lib.function_base import angle
from numpy.lib.function_base import unwrap
from numpy.lib.function_base import sinc
from numpy.lib.function_base import array_pad
from numpy.lib.twodim_base import compress
from numpy.lib.twodim_base import diagflat
from numpy.lib.twodim_base import diagonal
from numpy.lib.twodim_base import tri
from numpy.lib.twodim_base import tril
from numpy.lib.twodim_base import triu
from numpy.lib.twodim_base import vandermonde
from numpy.lib.twodim_base import mask_indices
from numpy.lib.twodim_base import tril_indices
from numpy.lib.twodim_base import triu_indices
from numpy.lib.twodim_base import tril_indices_from
from numpy.lib.twodim_base import triu_indices_from
from numpy.linalg.linalg import matrix_power
from numpy.linalg.linalg import solve
from numpy.linalg.linalg import tensorsolve
from numpy.linalg.linalg import tensorinv
from numpy.linalg.linalg import inv
from numpy.linalg.linalg import cholesky
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import pinv
from numpy.linalg.linalg import lstsq
from numpy.linalg.linalg import linalg_norm
from numpy.linalg.linalg import cond
from numpy.linalg.linalg import matrix_rank
from numpy.linalg.linalg import slogdet
from numpy.linalg.linalg import det
from numpy.linalg.linalg import svd
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import eig
from numpy.linalg.linalg import eigh
from numpy.linalg.linalg import multi_dot
from numpy.linalg.linalg import matrix_power
from numpy.linalg.linalg import solve
from numpy.linalg.linalg import tensorsolve
from numpy.linalg.linalg import tensorinv
from numpy.linalg.linalg import inv
from numpy.linalg.linalg import cholesky
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import pinv
from numpy.linalg.linalg import lstsq
from numpy.linalg.linalg import linalg_norm
from numpy.linalg.linalg import cond
from numpy.linalg.linalg import matrix_rank
from numpy.linalg.linalg import slogdet
from numpy.linalg.linalg import det
from numpy.linalg.linalg import svd
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import eig
from numpy.linalg.linalg import eigh
from numpy.linalg.linalg import multi_dot
```

```
epoch [ 1/5]
Image Shape: [2, 5,0,0,3, 3] obtained 0.5-0,3
```

Image of the combined tensor corresponding to concatenation.


```
epoch [ 2/5]
Image Shape: [2, 5,0,0,3, 3] obtained 0.5-0,3
```

Image of the other combined tensor corresponding to concatenation.

a.3a) (3)

```
epoch [ 3/5] [ 1/ 1] Generatorloss: 1.2662
Image : Shape[batch,w,h,c]: [batch=2,128,128,3] [c1,c2,c3,c4,c5,c6,c7,c8]: [128,64,32,16,8,4,2,1]
epoch [ 4/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 5/5] [ 1/ 1] Generatorloss: 1.2662
```


a.3a) (4)

```
epoch [ 1/5] [ 1/ 1] Generatorloss: 1.2662
Image : Shape[batch,w,h,c]: [batch=2,128,128,3] [c1,c2,c3,c4,c5,c6,c7,c8]: [128,64,32,16,8,4,2,1]
epoch [ 2/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 3/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 4/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 5/5] [ 1/ 1] Generatorloss: 1.2662
```


The visual patterns are much more evident compared to the other image.

Often the number of features c1, c2, c3 ... c8 are, especially for new runs, a little 2logp.

2. Research other pretrained networks in tensorflow and run DeepDream on their layers/filters.

I used a mixed5b layer out of the inceptionv3 model.

```
!pip install keras
```

```
pip install -q -U keras-preprocessing
Collecting keras-preprocessing
  Downloading https://files.pythonhosted.org/packages/ed/ca/a8a7ebb4a0e690cb9a56621d8a2b2f1e9ca0a4d764a6616f8778a2a6b0d2/Keras_Preprocessing-1.1.0-py2.py3-none-any.whl (42kB)
     |100%| 51kB, 2.8MB/s, ETA 0:00:00
Installing collected packages: keras-preprocessing
Successfully installed keras-preprocessing-1.1.0
```

```
import numpy
from keras.applications import InceptionV3
from keras.preprocessing.image import load_img, save_img
model = InceptionV3(weights='imagenet',
                    include_top=False)
layer_contributions = {
    'mixed2': 0.2,
    'mixed3': 0.5,
    'mixed4': 2.,
    'mixed5': 1.5,
}
```

```
Using TensorFlow backend.
Downloading data from https://github.com/fchollet/keras-models/release/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) in layer-9. Expected shape (None, 299, 299, 3) in layer-9. Things may perform oddly.
layer_name, coeff, shape: ('mixed2', 0.2, (None, None, None, 288))
layer_name, coeff, shape: ('mixed3', 0.5, (None, None, None, 768))
layer_name, coeff, shape: ('mixed4', 2.0, (None, None, None, 768))
layer_name, coeff, shape: ('mixed5', 1.5, (None, None, None, 768))
```

```
layer_dict = dict(layer.name, layer)
print('Layer contributions: ', layer_contributions)
print('Layer dict: ', layer_dict)
```

```
Layer contributions:  {'mixed2': 0.2, 'mixed3': 0.5, 'mixed4': 2.0, 'mixed5': 1.5}
Layer dict:  {'mixed2': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed3': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed4': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed5': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>}
```

```
loss = K.variable(0.)
for layer_name in layer_contributions:
    coeff = layer_contributions[layer_name]
    activation = layer_dict[layer_name].output
    scaling = K.prod(K.cast(K.shape(activation), 'float32'))
    loss = loss + coeff * K.sum(K.square(activation[:, 2: -2, 2: -2, :])) / scaling
```

```
Instructions for updating:
Use tf.cast instead.
```

```
dream = model.input
grad = K.gradients(dream, loss)[0]
grad = grad / K.maximum(K.mean(K.abs(grad)), 1e-7)
outputs = [loss, grad]
fetch_loss_and_grads = K.function([dream], outputs)
def eval_loss_and_grads(x):
    outs = fetch_loss_and_grads([x])
    loss_value = outs[0]
    grad_values = outs[1]
    return loss_value, grad_values
def resize_img(img, size):
    img = np.copy(img)
    factor = (float(size[1]) / img.shape[1],
              float(size[2]) / img.shape[2])
    return scipy.ndimage.zoom(img, (1, factor[0], factor[1], 1),
                              order=0)
def gradient_ascent(x, iterations, step, max_loss=None):
    for i in range(iterations):
        print('...Loss value at', i, ':', loss_value)
        x *= (1 - 127.5) / 255.
        x += step * grad_values
        if max_loss is not None:
            save_img('/tmp/stopped.jpg', x)
            print('Stopping at iteration', i, 'due to exceeding max_loss')
            break
        else:
            print('Saving image at iteration', i)
    return x
step = 0.01
num_octave = 3
octave_scale = 1.4
iterations = 20
max_loss = None
base_image_path = '/tmp/base.jpg'
base_image = scipy.misc.imread(base_image_path)
original_shape = base_image.shape[:2]
print('Original shape: ', original_shape)
for shape in successive_shapes:
    print('Processing image shape', shape)
    base_image = resize_img(base_image, shape)
    img = gradient_ascent(img,
                          iterations=iterations,
                          step=step,
                          max_loss=max_loss)
    upscaled_shape = shape[:2]
    lost_detail = np.copy(base_image) - resize_img(upscaled_shape,
                                                   base_image)
    print('Saving image at iteration', i, 'due to exceeding max_loss')
    break
    else:
    print('Saving image at iteration', i)
return x
```

```
Using TensorFlow backend.
Downloading data from https://github.com/fchollet/keras-models/release/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) in layer-9. Expected shape (None, 299, 299, 3) in layer-9. Things may perform oddly.
layer_name, coeff, shape: ('mixed2', 0.2, (None, None, None, 288))
layer_name, coeff, shape: ('mixed3', 0.5, (None, None, None, 768))
layer_name, coeff, shape: ('mixed4', 2.0, (None, None, None, 768))
layer_name, coeff, shape: ('mixed5', 1.5, (None, None, None, 768))
Layer contributions:  {'mixed2': 0.2, 'mixed3': 0.5, 'mixed4': 2.0, 'mixed5': 1.5}
Layer dict:  {'mixed2': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed3': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed4': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>, 'mixed5': <keras.layers.merge.Concatenate object at 0x7f8d1a0a0a90>}
Instructions for updating:
Use tf.cast instead.
Original shape:  (1, 5,0,0,3, 3)
Processing image shape (1, 5,0,0,3, 3)
...Loss value at 0 : 1.2662
Saving image at iteration 0
...Loss value at 1 : 1.2662
Saving image at iteration 1
...Loss value at 2 : 1.2662
Saving image at iteration 2
...Loss value at 3 : 1.2662
Saving image at iteration 3
...Loss value at 4 : 1.2662
Saving image at iteration 4
...Loss value at 5 : 1.2662
Saving image at iteration 5
...Loss value at 6 : 1.2662
Saving image at iteration 6
...Loss value at 7 : 1.2662
Saving image at iteration 7
...Loss value at 8 : 1.2662
Saving image at iteration 8
...Loss value at 9 : 1.2662
Saving image at iteration 9
...Loss value at 10 : 1.2662
Saving image at iteration 10
...Loss value at 11 : 1.2662
Saving image at iteration 11
...Loss value at 12 : 1.2662
Saving image at iteration 12
...Loss value at 13 : 1.2662
Saving image at iteration 13
...Loss value at 14 : 1.2662
Saving image at iteration 14
...Loss value at 15 : 1.2662
Saving image at iteration 15
...Loss value at 16 : 1.2662
Saving image at iteration 16
...Loss value at 17 : 1.2662
Saving image at iteration 17
...Loss value at 18 : 1.2662
Saving image at iteration 18
...Loss value at 19 : 1.2662
Saving image at iteration 19
Processing image shape (1, 7,0,0,3, 3)
...Loss value at 0 : 1.2662
Saving image at iteration 0
...Loss value at 1 : 1.2662
Saving image at iteration 1
...Loss value at 2 : 1.2662
Saving image at iteration 2
...Loss value at 3 : 1.2662
Saving image at iteration 3
...Loss value at 4 : 1.2662
Saving image at iteration 4
...Loss value at 5 : 1.2662
Saving image at iteration 5
...Loss value at 6 : 1.2662
Saving image at iteration 6
...Loss value at 7 : 1.2662
Saving image at iteration 7
...Loss value at 8 : 1.2662
Saving image at iteration 8
...Loss value at 9 : 1.2662
Saving image at iteration 9
...Loss value at 10 : 1.2662
Saving image at iteration 10
...Loss value at 11 : 1.2662
Saving image at iteration 11
...Loss value at 12 : 1.2662
Saving image at iteration 12
...Loss value at 13 : 1.2662
Saving image at iteration 13
...Loss value at 14 : 1.2662
Saving image at iteration 14
...Loss value at 15 : 1.2662
Saving image at iteration 15
...Loss value at 16 : 1.2662
Saving image at iteration 16
...Loss value at 17 : 1.2662
Saving image at iteration 17
...Loss value at 18 : 1.2662
Saving image at iteration 18
...Loss value at 19 : 1.2662
Saving image at iteration 19
Processing image shape (1, 9,0,0,3, 3)
...Loss value at 0 : 1.2662
Saving image at iteration 0
...Loss value at 1 : 1.2662
Saving image at iteration 1
...Loss value at 2 : 1.2662
Saving image at iteration 2
...Loss value at 3 : 1.2662
Saving image at iteration 3
...Loss value at 4 : 1.2662
Saving image at iteration 4
...Loss value at 5 : 1.2662
Saving image at iteration 5
...Loss value at 6 : 1.2662
Saving image at iteration 6
...Loss value at 7 : 1.2662
Saving image at iteration 7
...Loss value at 8 : 1.2662
Saving image at iteration 8
...Loss value at 9 : 1.2662
Saving image at iteration 9
...Loss value at 10 : 1.2662
Saving image at iteration 10
...Loss value at 11 : 1.2662
Saving image at iteration 11
...Loss value at 12 : 1.2662
Saving image at iteration 12
...Loss value at 13 : 1.2662
Saving image at iteration 13
...Loss value at 14 : 1.2662
Saving image at iteration 14
...Loss value at 15 : 1.2662
Saving image at iteration 15
...Loss value at 16 : 1.2662
Saving image at iteration 16
...Loss value at 17 : 1.2662
Saving image at iteration 17
...Loss value at 18 : 1.2662
Saving image at iteration 18
...Loss value at 19 : 1.2662
Saving image at iteration 19
```

```
!cp /tmp/base.jpg /tmp/base.jpg
!cp /tmp/stopped.jpg /tmp/stopped.jpg
!cp /tmp/base.jpg /tmp/base.jpg
!cp /tmp/stopped.jpg /tmp/stopped.jpg
```

```
cp: cannot stat '/tmp/base.jpg': No such file or directory
cp: cannot stat '/tmp/stopped.jpg': No such file or directory
cp: cannot stat '/tmp/base.jpg': No such file or directory
cp: cannot stat '/tmp/stopped.jpg': No such file or directory
```

```
from google.colab.patches import cv2_imshow
from PIL import Image
from IPIL import display
from matplotlib.pyplot import imshow
from numpy.random.mtrand import uniform
from numpy.random.mtrand import randint
from numpy.random.mtrand import seed
from numpy.random.mtrand import shuffle
from numpy.random.mtrand import choice
from numpy.random.mtrand import gamma
from numpy.random.mtrand import beta
from numpy.random.mtrand import chisquare
from numpy.random.mtrand import exponential
from numpy.random.mtrand import f
from numpy.random.mtrand import gumbel
from numpy.random.mtrand import laplace
from numpy.random.mtrand import logistic
from numpy.random.mtrand import lognormal
from numpy.random.mtrand import logseries
from numpy.random.mtrand import multinomial
from numpy.random.mtrand import multivariate_normal
from numpy.random.mtrand import negative_binomial
from numpy.random.mtrand import noncentral_chisquare
from numpy.random.mtrand import noncentral_f
from numpy.random.mtrand import pareto
from numpy.random.mtrand import poisson
from numpy.random.mtrand import power
from numpy.random.mtrand import rayleigh
from numpy.random.mtrand import standard_cauchy
from numpy.random.mtrand import standard_exponential
from numpy.random.mtrand import standard_gamma
from numpy.random.mtrand import standard_normal
from numpy.random.mtrand import standard_t
from numpy.random.mtrand import triangular
from numpy.random.mtrand import uniform
from numpy.random.mtrand import vonmises
from numpy.random.mtrand import wald
from numpy.random.mtrand import weibull
from numpy.random.mtrand import zipf
from numpy import *
from numpy import ma
from numpy.ma.core import *
from numpy.ma.extras import *
from numpy.lib.arraysetops import ediff1d
from numpy.lib.arraysetops import intersect1d
from numpy.lib.arraysetops import setdiff1d
from numpy.lib.arraysetops import setxor1d
from numpy.lib.arraysetops import union1d
from numpy.lib.arraysetops import unique
from numpy.lib.arraysetops import in1d
from numpy.lib.arraysetops import isin
from numpy.lib.function_base import select
from numpy.lib.function_base import piecewise
from numpy.lib.function_base import trim_zeros
from numpy.lib.function_base import angle
from numpy.lib.function_base import unwrap
from numpy.lib.function_base import sinc
from numpy.lib.function_base import array_pad
from numpy.lib.twodim_base import compress
from numpy.lib.twodim_base import diagflat
from numpy.lib.twodim_base import diagonal
from numpy.lib.twodim_base import tri
from numpy.lib.twodim_base import tril
from numpy.lib.twodim_base import triu
from numpy.lib.twodim_base import vandermonde
from numpy.lib.twodim_base import mask_indices
from numpy.lib.twodim_base import tril_indices
from numpy.lib.twodim_base import triu_indices
from numpy.lib.twodim_base import tril_indices_from
from numpy.lib.twodim_base import triu_indices_from
from numpy.linalg.linalg import matrix_power
from numpy.linalg.linalg import solve
from numpy.linalg.linalg import tensorsolve
from numpy.linalg.linalg import tensorinv
from numpy.linalg.linalg import inv
from numpy.linalg.linalg import cholesky
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import pinv
from numpy.linalg.linalg import lstsq
from numpy.linalg.linalg import linalg_norm
from numpy.linalg.linalg import cond
from numpy.linalg.linalg import matrix_rank
from numpy.linalg.linalg import slogdet
from numpy.linalg.linalg import det
from numpy.linalg.linalg import svd
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import eig
from numpy.linalg.linalg import eigh
from numpy.linalg.linalg import multi_dot
from numpy.linalg.linalg import matrix_power
from numpy.linalg.linalg import solve
from numpy.linalg.linalg import tensorsolve
from numpy.linalg.linalg import tensorinv
from numpy.linalg.linalg import inv
from numpy.linalg.linalg import cholesky
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import pinv
from numpy.linalg.linalg import lstsq
from numpy.linalg.linalg import linalg_norm
from numpy.linalg.linalg import cond
from numpy.linalg.linalg import matrix_rank
from numpy.linalg.linalg import slogdet
from numpy.linalg.linalg import det
from numpy.linalg.linalg import svd
from numpy.linalg.linalg import eigvals
from numpy.linalg.linalg import eigvalsh
from numpy.linalg.linalg import eig
from numpy.linalg.linalg import eigh
from numpy.linalg.linalg import multi_dot
```

```
epoch [ 1/5]
Image Shape: [2, 5,0,0,3, 3] obtained 0.5-0,3
```

Image of the combined tensor corresponding to concatenation.


```
epoch [ 2/5]
Image Shape: [2, 5,0,0,3, 3] obtained 0.5-0,3
```

Image of the other combined tensor corresponding to concatenation.

a.3a) (3)

```
epoch [ 3/5] [ 1/ 1] Generatorloss: 1.2662
Image : Shape[batch,w,h,c]: [batch=2,128,128,3] [c1,c2,c3,c4,c5,c6,c7,c8]: [128,64,32,16,8,4,2,1]
epoch [ 4/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 5/5] [ 1/ 1] Generatorloss: 1.2662
```


a.3a) (4)

```
epoch [ 1/5] [ 1/ 1] Generatorloss: 1.2662
Image : Shape[batch,w,h,c]: [batch=2,128,128,3] [c1,c2,c3,c4,c5,c6,c7,c8]: [128,64,32,16,8,4,2,1]
epoch [ 2/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 3/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 4/5] [ 1/ 1] Generatorloss: 1.2662
epoch [ 5/5] [ 1/ 1] Generatorloss: 1.2662
```


The visual patterns are much more evident compared to the other image.

Often the number of features c1, c2, c3 ... c8 are, especially for new runs, a little 2logp.

2. Research other pretrained networks in tensorflow and run DeepDream on their layers/filters.

I used a mixed5b layer out of the inceptionv3 model.

```
!pip install keras
```

```
pip install -q -U keras-preprocessing
Collecting keras-preprocessing
  Downloading https://files.pythonhosted.org/packages/ed/ca/a8a7ebb4a0e690cb9a56621d8a2b2f1e9ca0a4d764a6616f8778a2a6b0d2/Keras_Preprocessing-1.1.0-py2.py3-none-any.whl (42kB)
     |100%| 51kB, 2.8MB/s, ETA 0:00:00
Installing collected packages: keras-preprocessing
Successfully installed keras-preprocessing-1.1.0
```

```
import numpy
from keras.applications import InceptionV3
from keras.preprocessing.image import load_img, save_img
model = InceptionV3(weights='imagenet',
                    include_top=False)
layer_contributions = {
    'mixed2': 0.2,
    'mixed3': 0.5,
    'mixed4': 2.,
    'mixed5': 1.5,
}
```

```
Using TensorFlow backend.
Downloading data from https://github.com/fchollet/keras-models/release/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) in layer-9. Expected shape (None, 299, 299, 3) in layer-9. Things may perform oddly.
layer_name, coeff, shape: ('mixed2', 0.2, (None, None, None, 288))
layer_name, coeff, shape: ('mixed3', 0.5, (None, None, None, 768))
layer_name, coeff, shape: ('mixed4', 2.0, (None, None, None, 768))
layer_name, coeff, shape: ('mixed5', 1.5, (None, None, None, 768))
```

```
layer_dict = dict(layer.name, layer)
print('Layer contributions: ', layer_contributions)
print('Layer dict: ', layer_dict)
```

```
Layer contributions:  {'mixed2': 0.2, 'mixed3': 0.5, 'mixed4': 2
```

output type	stream	name	stdout

output type	display_data		
mime bundle	{'text/plain': [] }		

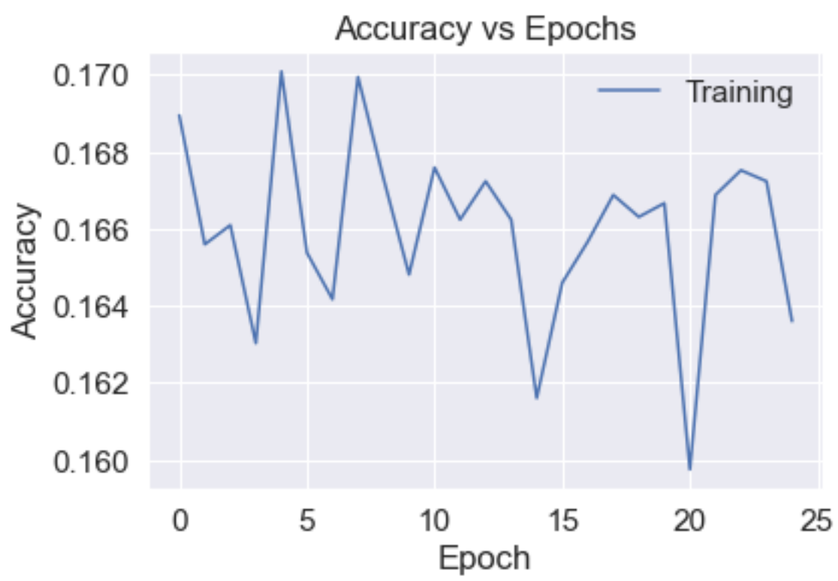
```
Epoch 25/25
188/188 - 112s - loss: 3.6259 - accuracy: 0.1636
```

Check the test accuracy and loss after training

In [30]:
```
test_loss, test_acc = aug_model.evaluate(test_images, test_labels, verbose=0)
print(f'Test loss: {test_loss}, accuracy: {test_acc}')
```

```
Test loss: 3.7419943809509277, accuracy: 0.15833333134651184
```

In [31]:
```
plt.plot(hist.history["accuracy"])
plt.title("Accuracy vs Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Training"])
plt.show()
```



In [32]:
```
plt.plot(hist.history["loss"])
plt.title("Loss vs Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["Training"])
plt.show()
```

Based on the result of the test set, the accuracy of the test set is increased by around 5 percents, which is good, but maybe not enough. Given the huge amount of data, we might attribute this small failure as insufficient training epochs and large batch size. In general, the result of this transfer learning using MobileNetV2 is not very ideal. To improve this, we can try spend more time on training or simply change to use another pre-trained net.