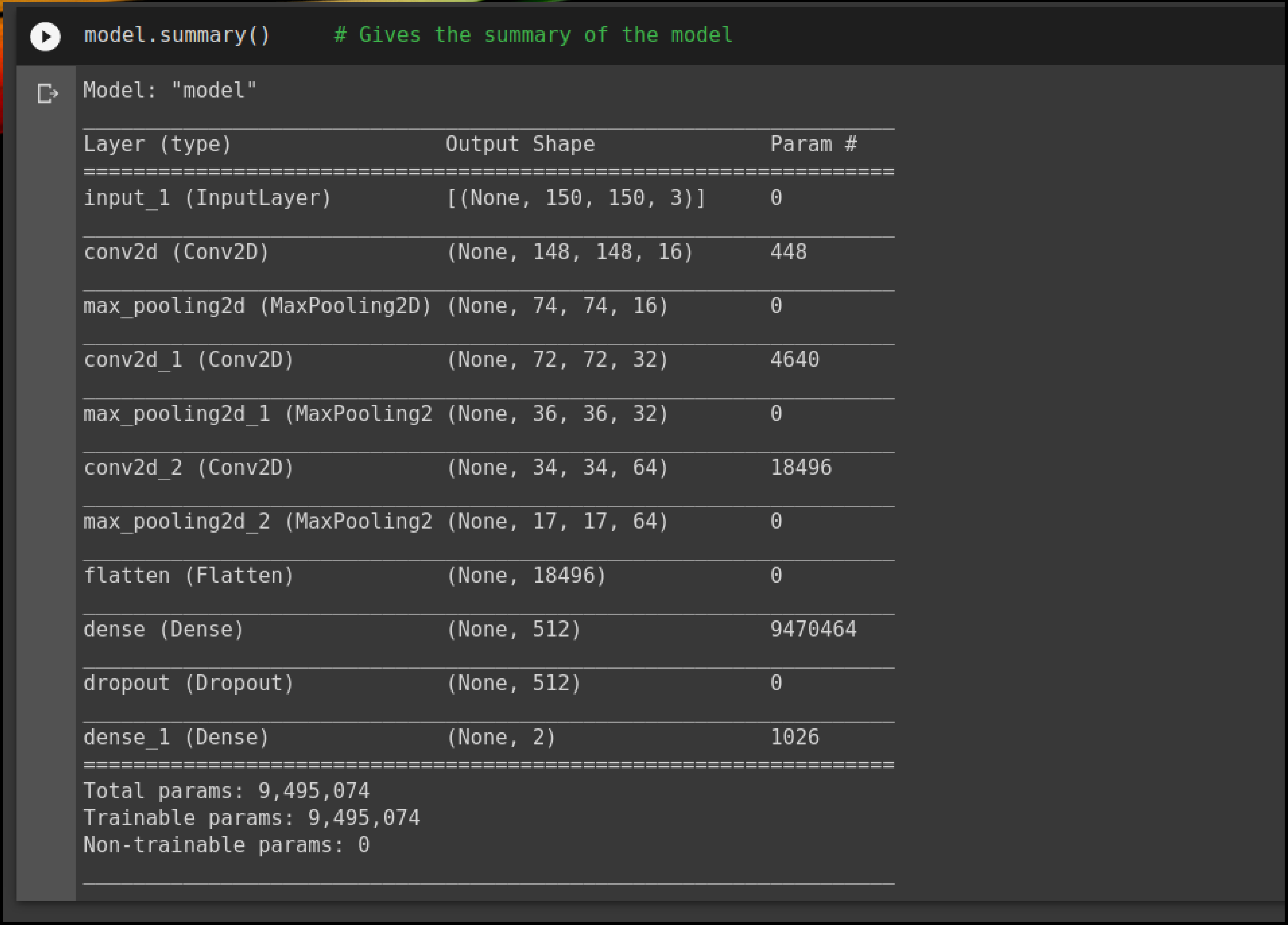
Image Classification

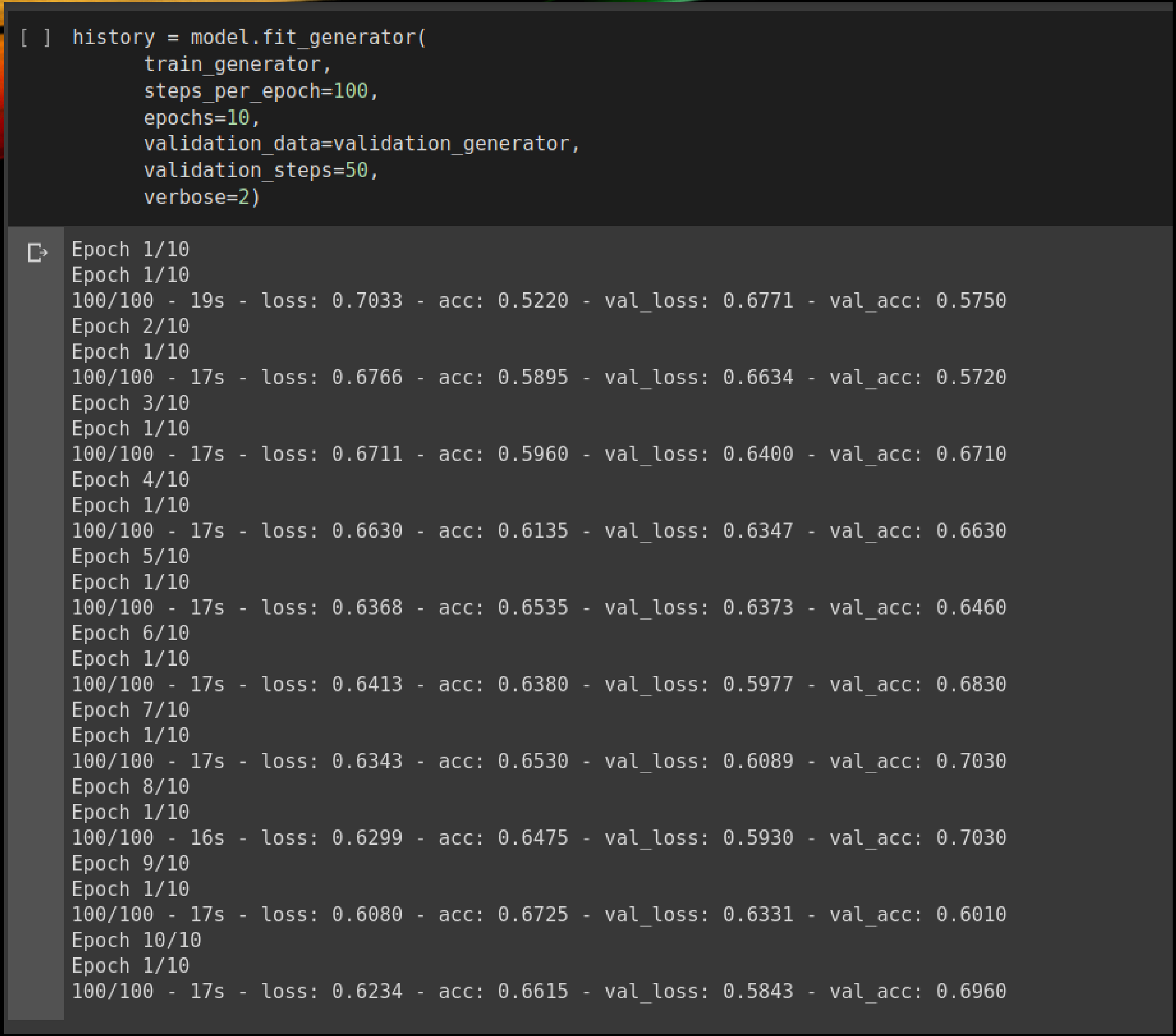
WALLPAPERSWIDE.COM

Multilayer Perceptron



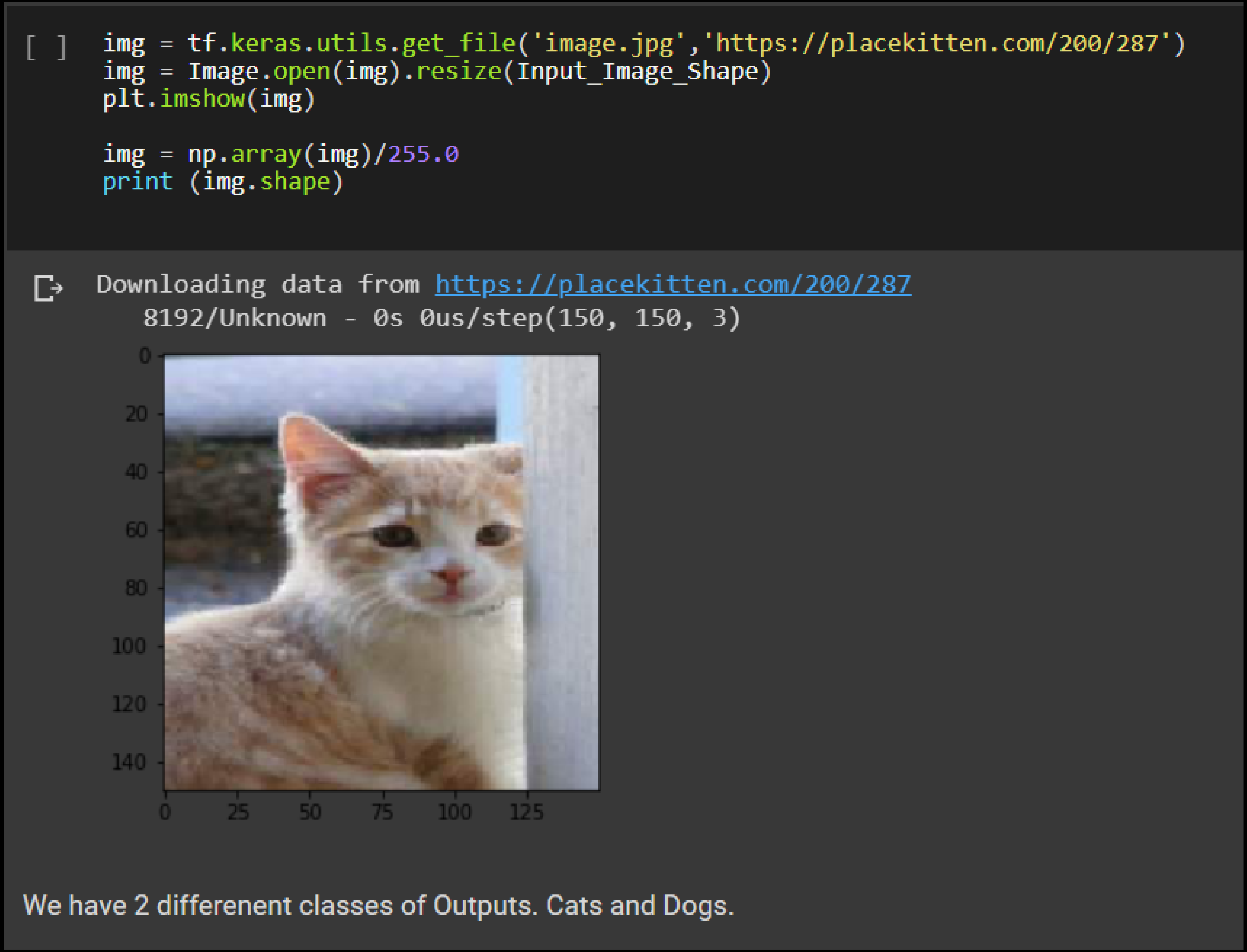
```
model.summary()     # Gives the summary of the model
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 148, 148, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 18496)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9470464
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 9,495,074
Trainable params: 9,495,074
Non-trainable params: 0
_________________________________________________________________
```



```
[ ] history = model.fit_generator(
          train_generator,
          steps_per_epoch=100,
          epochs=10,
          validation_data=validation_generator,
          validation_steps=50,
          verbose=2)
```

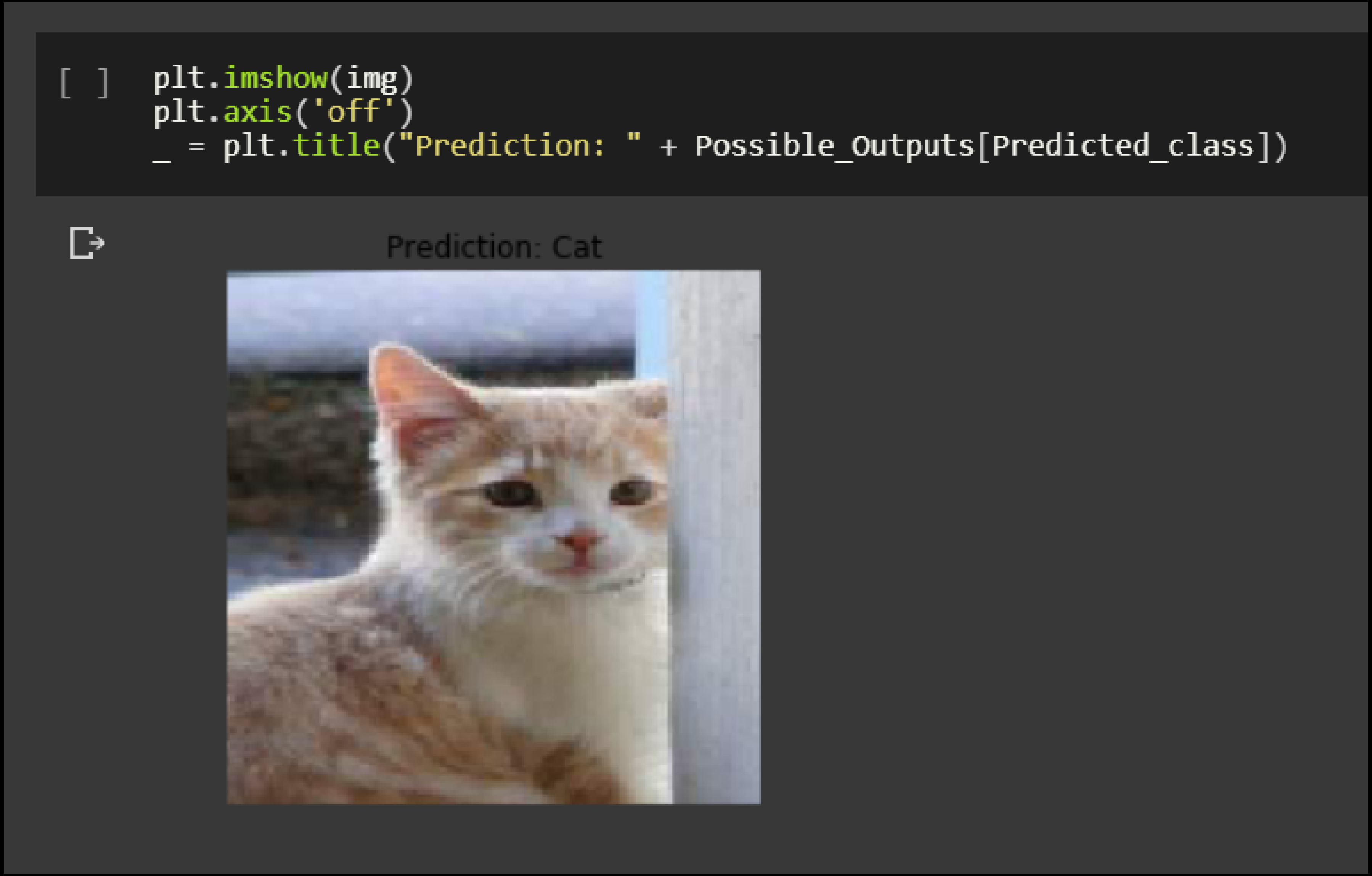
```
Epoch 1/10
Epoch 1/10
100/100 - 19s - loss: 0.7033 - acc: 0.5220 - val_loss: 0.6771 - val_acc: 0.5750
Epoch 2/10
Epoch 1/10
100/100 - 17s - loss: 0.6766 - acc: 0.5895 - val_loss: 0.6634 - val_acc: 0.5720
Epoch 3/10
Epoch 1/10
100/100 - 17s - loss: 0.6711 - acc: 0.5960 - val_loss: 0.6400 - val_acc: 0.6710
Epoch 4/10
Epoch 1/10
100/100 - 17s - loss: 0.6630 - acc: 0.6135 - val_loss: 0.6347 - val_acc: 0.6630
Epoch 5/10
Epoch 1/10
100/100 - 17s - loss: 0.6368 - acc: 0.6535 - val_loss: 0.6373 - val_acc: 0.6460
Epoch 6/10
Epoch 1/10
100/100 - 17s - loss: 0.6413 - acc: 0.6380 - val_loss: 0.5977 - val_acc: 0.6830
Epoch 7/10
Epoch 1/10
100/100 - 17s - loss: 0.6343 - acc: 0.6530 - val_loss: 0.6089 - val_acc: 0.7030
Epoch 8/10
Epoch 1/10
100/100 - 16s - loss: 0.6299 - acc: 0.6475 - val_loss: 0.5930 - val_acc: 0.7030
Epoch 9/10
Epoch 1/10
100/100 - 17s - loss: 0.6080 - acc: 0.6725 - val_loss: 0.6331 - val_acc: 0.6010
Epoch 10/10
Epoch 1/10
100/100 - 17s - loss: 0.6234 - acc: 0.6615 - val_loss: 0.5843 - val_acc: 0.6960
```



```
img = tf.keras.utils.get_file('image.jpg','https://placekitten.com/200/287')
img = Image.open(img).resize(Input_Image_Shape)
plt.imshow(img)

img = np.array(img)/255.0
print (img.shape)
```

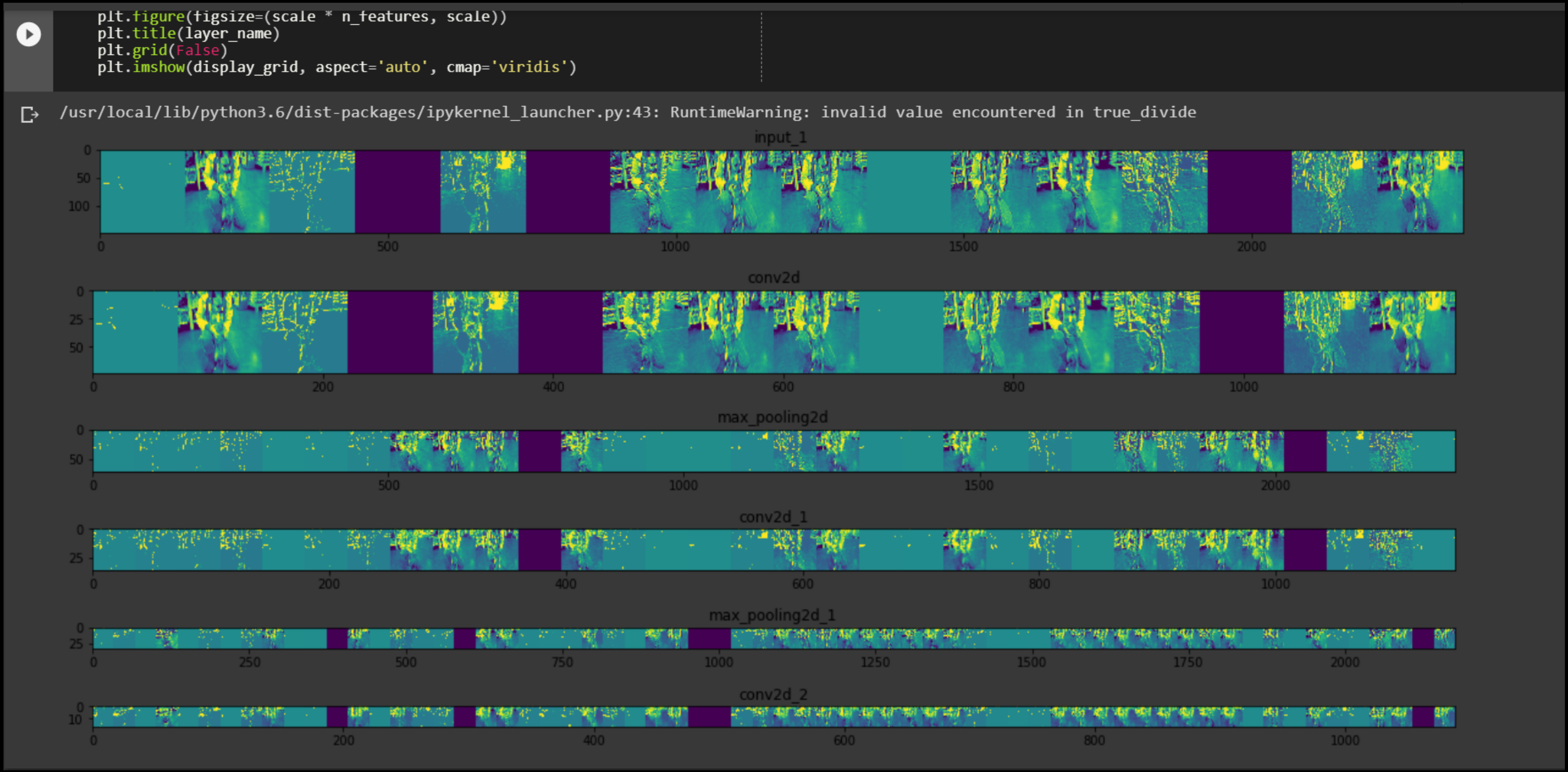
```
Downloading data from https://placekitten.com/200/287
   8192/Unknown - 0s 0us/step(150, 150, 3)
```

We have 2 differenent classes of Outputs. Cats and Dogs.



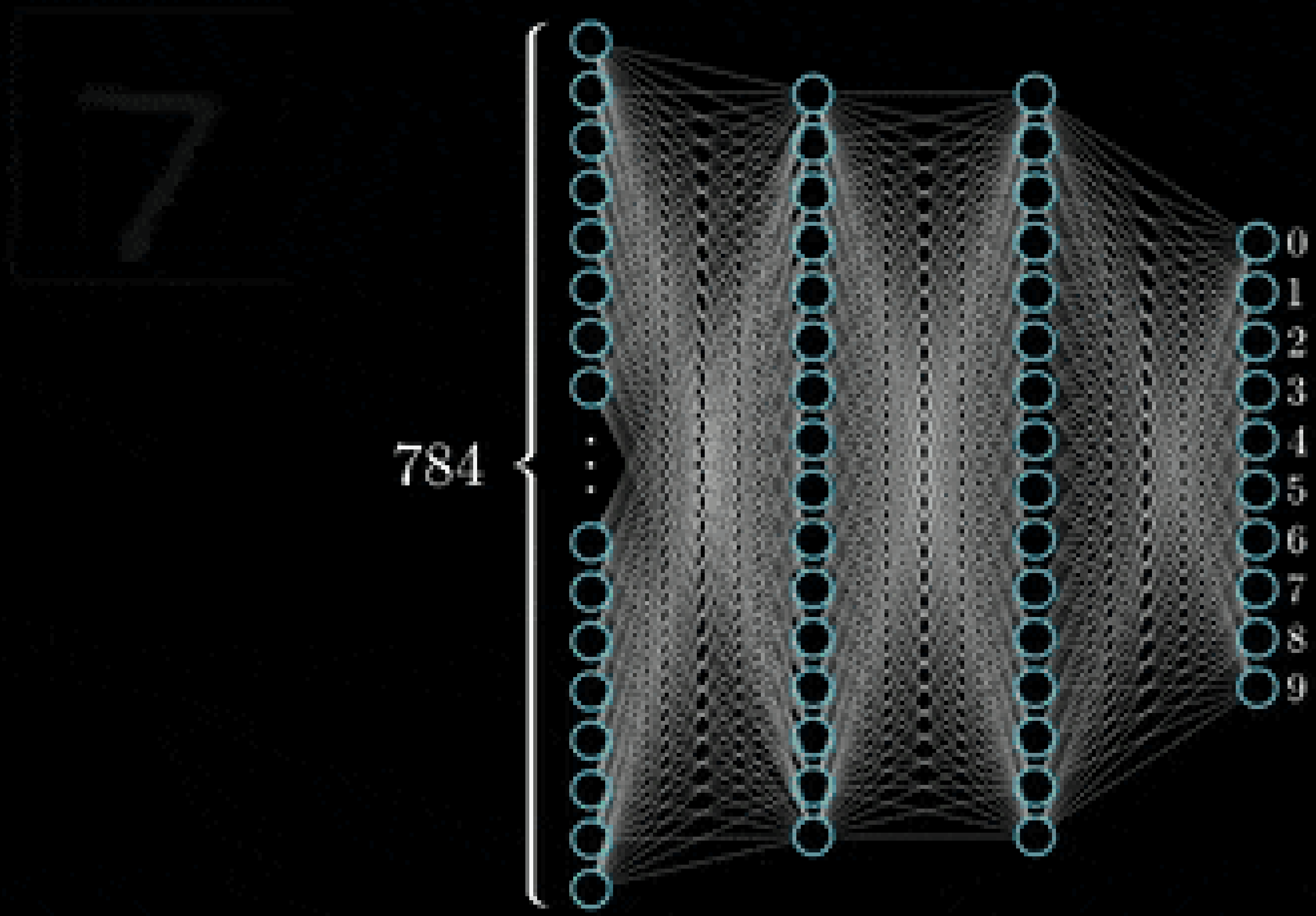
```
plt.imshow(img)
plt.axis('off')
_ = plt.title("Prediction: " + Possible_Outputs[Predicted_class])
```

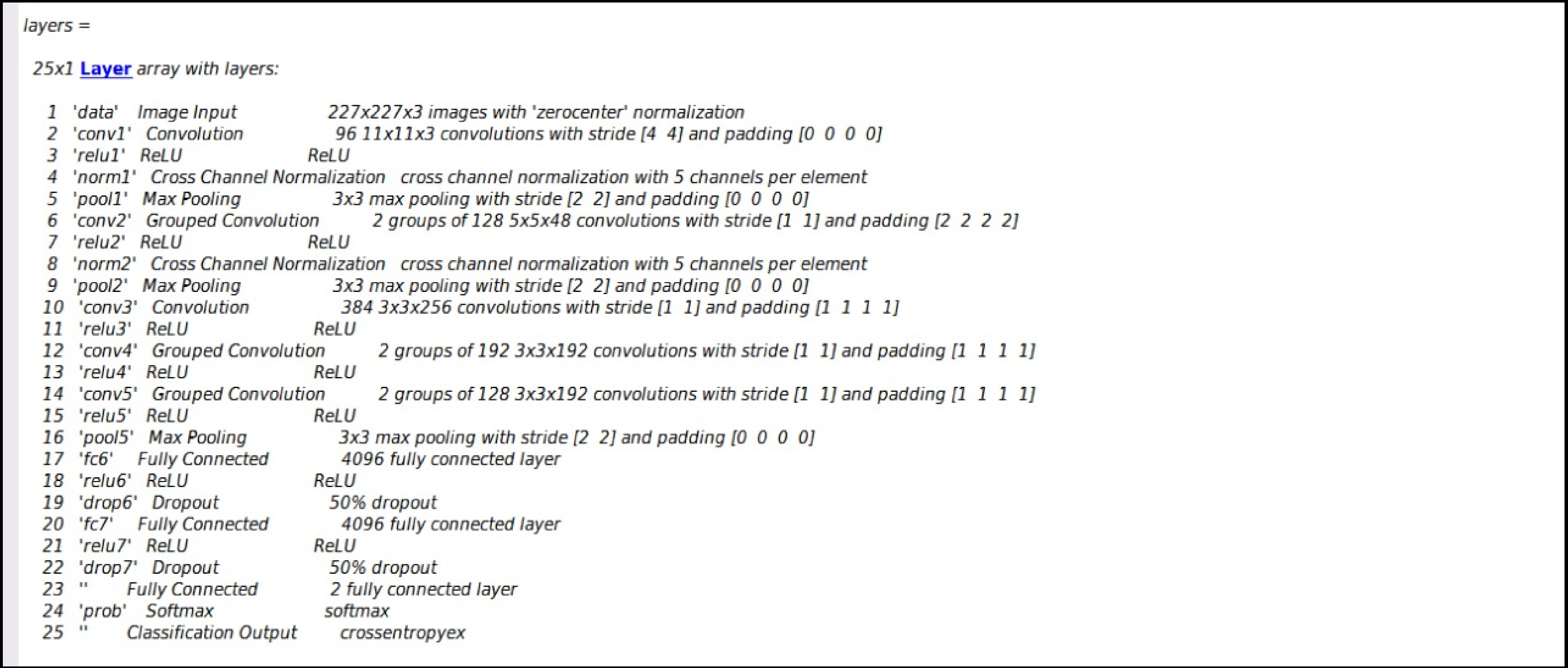
```
Prediction: Cat
```



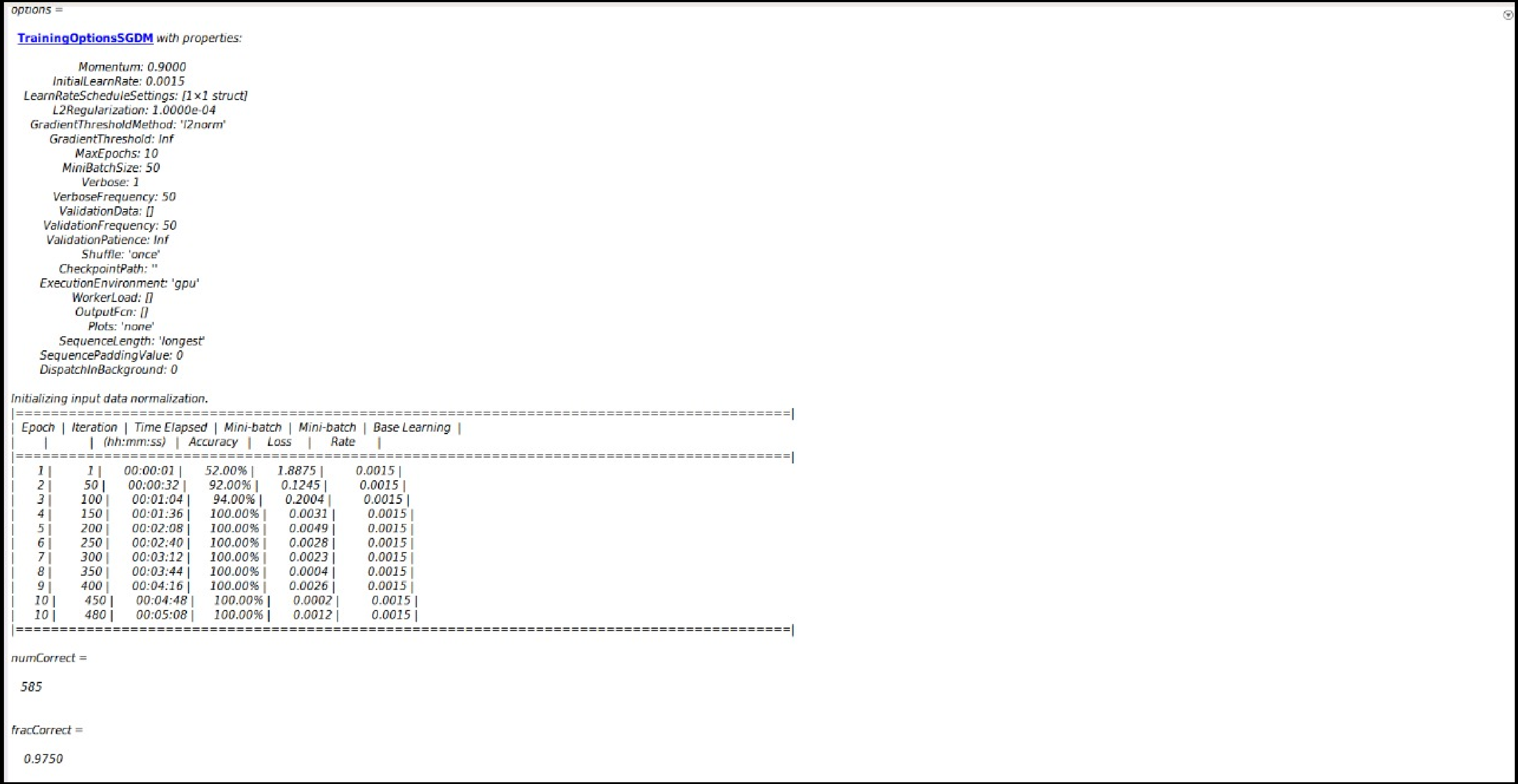
```
    plt.figure(figsize=(scale * n_features, scale))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:43: RuntimeWarning: invalid value encountered in true_divide
```

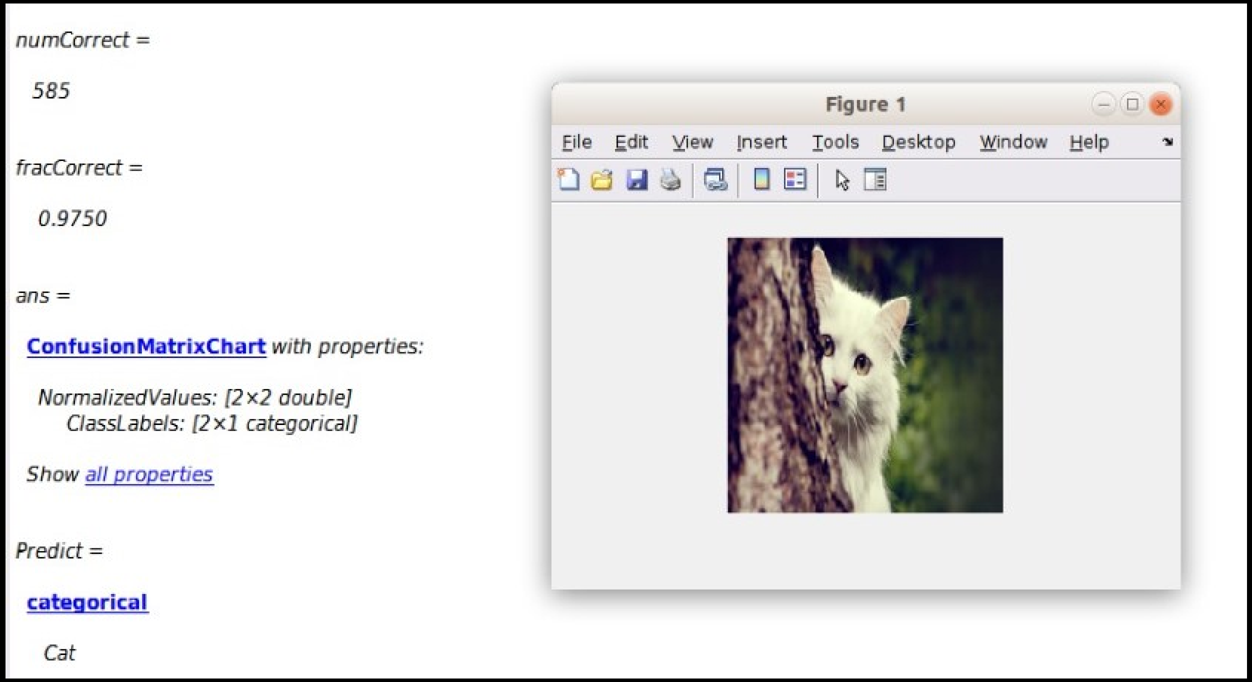

784
0
1
2
3
4
5
6
7
8
9



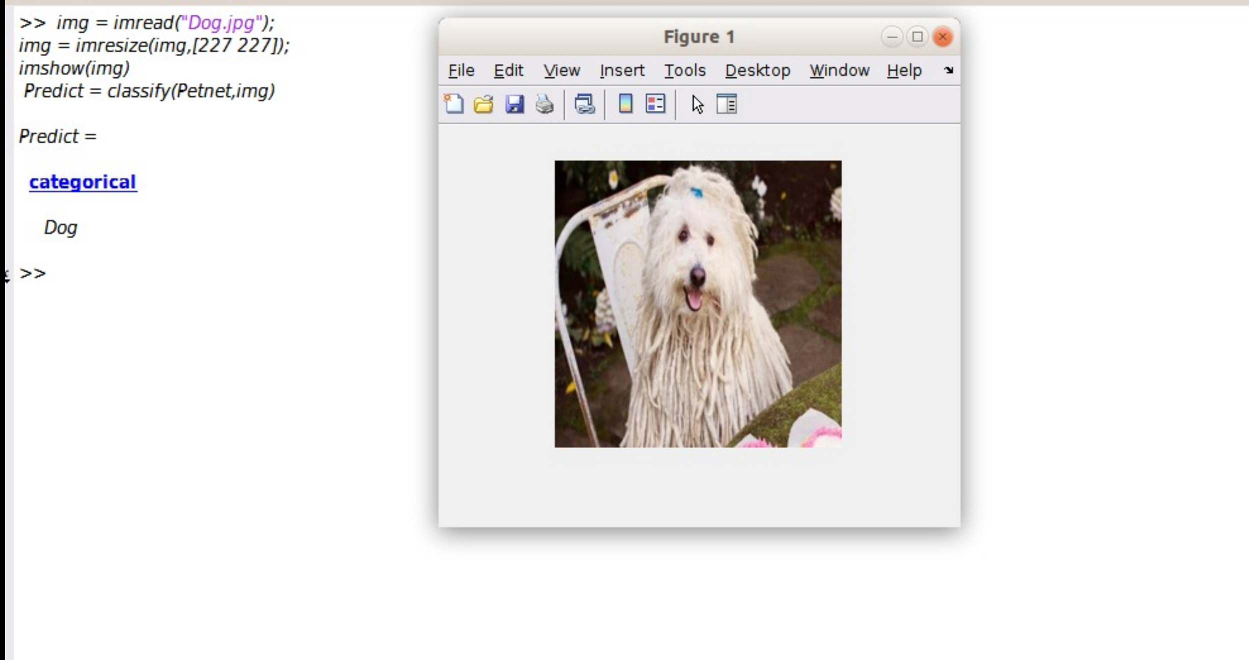
```
layers =

  25x1 Layer array with layers:

     1   'data'    Image Input                   227x227x3 images with 'zerocenter' normalization
     2   'conv1'   Convolution                   96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
     3   'relu1'   ReLU                          ReLU
     4   'norm1'   Cross Channel Normalization   cross channel normalization with 5 channels per element
     5   'pool1'   Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv2'   Grouped Convolution           2 groups of 128 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'   ReLU                          ReLU
     8   'norm2'   Cross Channel Normalization   cross channel normalization with 5 channels per element
     9   'pool2'   Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    10   'conv3'   Convolution                   384 3x3x256 convolutions with stride [1  1] and padding [1  1  1  1]
    11   'relu3'   ReLU                          ReLU
    12   'conv4'   Grouped Convolution           2 groups of 192 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    13   'relu4'   ReLU                          ReLU
    14   'conv5'   Grouped Convolution           2 groups of 128 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    15   'relu5'   ReLU                          ReLU
    16   'pool5'   Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    17   'fc6'     Fully Connected               4096 fully connected layer
    18   'relu6'   ReLU                          ReLU
    19   'drop6'   Dropout                       50% dropout
    20   'fc7'     Fully Connected               4096 fully connected layer
    21   'relu7'   ReLU                          ReLU
    22   'drop7'   Dropout                       50% dropout
    23   ''        Fully Connected               2 fully connected layer
    24   'prob'    Softmax                       softmax
    25   ''        Classification Output         crossentropyex
```



```
options =

  TrainingOptionsSGDM with properties:

                    Momentum: 0.9000
              InitialLearnRate: 0.0015
     LearnRateScheduleSettings: [1×1 struct]
              L2Regularization: 1.0000e-04
       GradientThresholdMethod: 'l2norm'
             GradientThreshold: Inf
                     MaxEpochs: 10
                 MiniBatchSize: 50
                       Verbose: 1
              VerboseFrequency: 50
                ValidationData: []
           ValidationFrequency: 50
            ValidationPatience: Inf
                       Shuffle: 'once'
                CheckpointPath: ''
          ExecutionEnvironment: 'gpu'
                    WorkerLoad: []
                     OutputFcn: []
                         Plots: 'none'
                SequenceLength: 'longest'
          SequencePaddingValue: 0
          DispatchInBackground: 0

Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:01 |       52.00% |       1.8875 |          0.0015 |
|       2 |          50 |       00:00:32 |       92.00% |       0.1245 |          0.0015 |
|       3 |         100 |       00:01:04 |       94.00% |       0.2004 |          0.0015 |
|       4 |         150 |       00:01:36 |      100.00% |       0.0031 |          0.0015 |
|       5 |         200 |       00:02:08 |      100.00% |       0.0049 |          0.0015 |
|       6 |         250 |       00:02:40 |      100.00% |       0.0028 |          0.0015 |
|       7 |         300 |       00:03:12 |      100.00% |       0.0023 |          0.0015 |
|       8 |         350 |       00:03:44 |      100.00% |       0.0004 |          0.0015 |
|       9 |         400 |       00:04:16 |      100.00% |       0.0026 |          0.0015 |
|      10 |         450 |       00:04:48 |      100.00% |       0.0002 |          0.0015 |
|      10 |         480 |       00:05:08 |      100.00% |       0.0012 |          0.0015 |
|========================================================================================|

numCorrect =

   585


fracCorrect =

    0.9750
```



```
numCorrect =

   585


fracCorrect =

    0.9750


ans =

  ConfusionMatrixChart with properties:

    NormalizedValues: [2×2 double]
         ClassLabels: [2×1 categorical]

  Show all properties


Predict =

  categorical

     Cat
```



```
>> img = imread("Dog.jpg");
img = imresize(img,[227 227]);
imshow(img)
 Predict = classify(Petnet,img)

Predict =

  categorical

    Dog

>>
```