Gurjus Singh

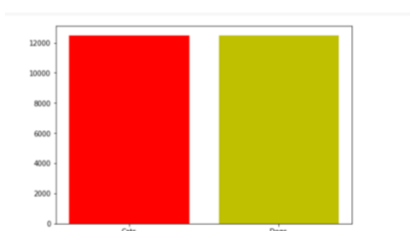
11/01/2020

MSDS 422 – Practical Machine Learning

ASSIGNMENT #7 IMAGE PROCESSING WITH CNN

Data preparation, exploration, visualization

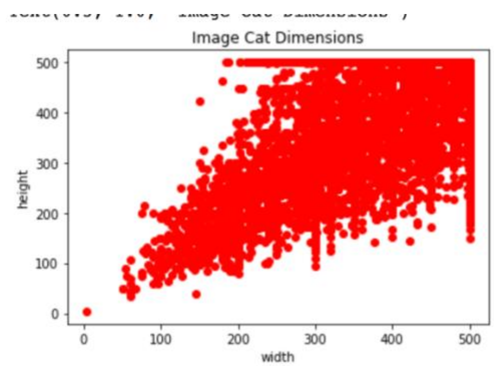
In this week's analysis, I used Convolution Neural Networks to classify Dogs and Cats. Particularly, I did **Binary Classification** on the Dogs and Cats Images from a data set of pixels obtained from Microsoft that were fed into a Convolution Neural Network. The Dataset contained a target variable which helped with the Binary Classification of Dogs or Cats. To start the analysis, I first had to import the packages. The most important for implementing Neural Networks involved Tensorflow and Keras. I then checked the versions to make sure the packages were 2.0.0 or above for Tensorflow. Since I was Google Colab, I had to mount the drive to access space to write files. I then downloaded my Dogs and Cats dataset to the temp directory in Google Drive. It was download as a .zip file and unzipped. I then saw how many Cat and Dog Images there were which was even at 12,501 each. I then did some EDA making a barplot representing number of Cat and Dog Images seen in 1-1.



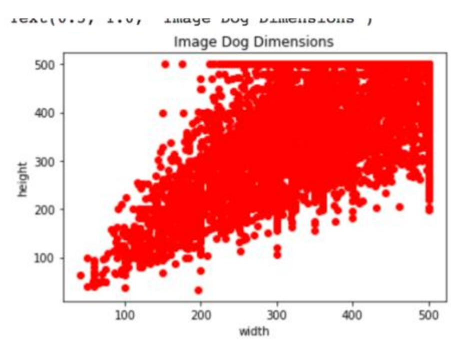
Dogs and Cat Dataset Size Barplot 1-1

I then wanted to find the size of each image or the dimensions of each image in Cats and Dogs and I found some interesting EDA using a dot plot below in 1-2 and 1-3. After doing initial

EDA I then setup directories for train and test sets. For this I set up a Training Set directory for Dogs and Cats, and a Test Set directory for the Dogs and Cats as well. After I made the directories I then made a function to split the data shuffling the data randomly along the 4 directories Train, Test Dogs and Cats. I found there were 22,500 in Dogs and Cats Train Data and 1,500 in the Test Data. I then create a subplot of 9 images of Cat Images and 9 images of the Dog images represented below in 1-4 and 1-5. After doing all the Data Prep and EDA it was time to train the model.

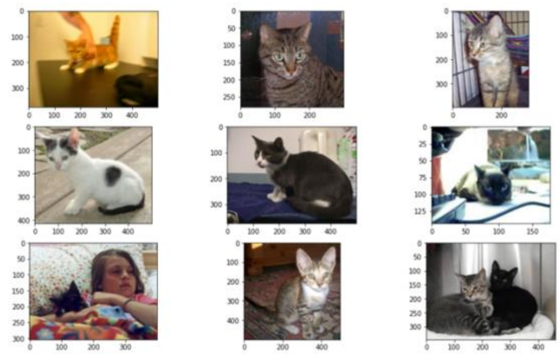


Cat Image Dimensions in a Plot 1-2



,

Dog Image Dimensions in a Plot 1-3



9 images of Cat Data 1-4

Dog Images 1-5

Review research design and modeling methods

As I mentioned above in the Data Prep section, before training the model, I had to set up four directories to split the data in. I then shuffled the data using a custom function to get an even number of Dogs and Cat images in Train folders and Test folders. After splitting the model, it was time to make and train the model. I used 3 2D Convolution Layers, 3 Max Pooling Layers, 1 Dropout layer, 1 Dense Layer, and an output layer using Sigmoid Function since the Sigmoid Function is used for Binary Classification. I also ReLu activation function in Networks as computes fast and is the most common activation functions used.

To describe the Convolution Neural Network, I created above one needs to understand key components such as the Convolution Layer and Max Pooling and what makes CNNs unique compared to Dense Neural Networks. The way the Convolution Layering works is the 1st layer represents a receptive field like in the eye, where it analyzes a small portion of the image at a time [1]. Other Convolution Layers after the 1st Convolution Layers try to analyze other details about the image [1]. The receptive field is also called a small rectangular portion of the image [1]. Each Convolution Layer also has a set of filters which analyzes different features of the receptive field such as lines/edges involved in the image which are either horizontal or vertical lines [1].

Filters can be used for each color RGB in this case. The filters create a feature map which outputs which features are represent in the rectangular receptive field [1]. The next layer after the Convolution Layers above are the Max Pooling Layers. The Max Pooling Layer is a summary of each Convolution Layer [1]. This means the feature map is summarized using a aggregation method such as max, or mean [1]. This is more like a shrinkage type layer, so less memory and resources are used by the machine [1]. CNNs are unique in that they can be either 1D which represents predicting audio data, 2D image data in the analysis I worked on and 3D which represents the prediction of Video Data [3]. They are also important in that they can consider that neighboring pixels are correlated and that they spatial features are similar, which basically means that feature that are far in space are also similar such as an image of a face where eyes and ears are similar [2]. With this information once can see why it is better to use Convolution Neural Networks on Dogs and Cat Binary Prediction as this is a powerful tool to use when similar features are far apart such as the face example, or that there are correlated pixels such as color of fur on the Cats and Dogs.

Review Results and Evaluate Model

I trained the 3 2D Convolution Layers, 3 Max Pooling Layers, 1 Dropout layer, 1 Dense Layer, and an output layer using Sigmoid Function first to see if the model was overfitting at the beginning. What I noticed from the Train and Validation Set, which is also the Test Set in our case since the Validation Set is pointing to the Testing Directory, that it was overfitting. Particularly the accuracy scores were 0.9015 for the Train Set and the Validation Set had an accuracy score of 0.8327. This is seen in the output 1-6. I then tried implementing a different model using more three dropout layers instead of one, so my model was 3 2D Convolution Layers, 3 Max Pooling Layers, 3 Dropout layers, 1 Dense Layer, and an output layer using Sigmoid Function. What I saw was there was a major difference as the training set accuracy was 0.8508 and Validation Accuracy was 0.8247 as seen in 1-7. This model was overfitting less, but I also tried several other models, but they were obviously overfitting compared to model 1-7.

```
Epoch 1/10
90/90 [==============================] - 70s 778ms/step - loss: 0.7195 - acc: 0.5683 - val_loss: 0.6097 - val_acc: 0.6627
Epoch 2/10
90/90 [==============================] - 70s 778ms/step - loss: 0.5611 - acc: 0.7085 - val_loss: 0.5146 - val_acc: 0.7500
Epoch 3/10
90/90 [==============================] - 70s 777ms/step - loss: 0.5007 - acc: 0.7515 - val_loss: 0.5026 - val_acc: 0.7420
Epoch 4/10
90/90 [==============================] - 70s 775ms/step - loss: 0.4482 - acc: 0.7883 - val_loss: 0.4244 - val_acc: 0.7913
Epoch 5/10
90/90 [==============================] - 72s 796ms/step - loss: 0.4129 - acc: 0.8108 - val_loss: 0.4309 - val_acc: 0.7907
Epoch 6/10
90/90 [==============================] - 71s 789ms/step - loss: 0.3743 - acc: 0.8305 - val_loss: 0.3893 - val_acc: 0.8173
Epoch 7/10
90/90 [==============================] - 71s 788ms/step - loss: 0.3380 - acc: 0.8530 - val_loss: 0.4027 - val_acc: 0.8180
Epoch 8/10
90/90 [==============================] - 72s 795ms/step - loss: 0.3018 - acc: 0.8684 - val_loss: 0.3954 - val_acc: 0.8180
Epoch 9/10
90/90 [==============================] - 71s 786ms/step - loss: 0.2792 - acc: 0.8832 - val_loss: 0.3864 - val_acc: 0.8340
Epoch 10/10
90/90 [==============================] - 71s 791ms/step - loss: 0.2376 - acc: 0.9015 - val_loss: 0.4170 - val_acc: 0.8327
```

Accuracy score comparisons for 10 epochs; with 1 dropout layer 1-6

```
Epoch 1/10
90/90 [==============================] - 71s 793ms/step - loss: 0.7834 - acc: 0.5521 - val_loss: 0.6478 - val_acc: 0.6273
Epoch 2/10
90/90 [==============================] - 71s 785ms/step - loss: 0.6078 - acc: 0.6637 - val_loss: 0.5994 - val_acc: 0.6793
Epoch 3/10
90/90 [==============================] - 70s 781ms/step - loss: 0.5270 - acc: 0.7362 - val_loss: 0.5046 - val_acc: 0.7527
Epoch 4/10
90/90 [==============================] - 71s 789ms/step - loss: 0.5024 - acc: 0.7528 - val_loss: 0.4840 - val_acc: 0.7660
Epoch 5/10
90/90 [==============================] - 71s 793ms/step - loss: 0.4690 - acc: 0.7745 - val_loss: 0.4531 - val_acc: 0.7807
Epoch 6/10
90/90 [==============================] - 70s 775ms/step - loss: 0.4374 - acc: 0.7964 - val_loss: 0.5071 - val_acc: 0.7527
Epoch 7/10
90/90 [==============================] - 70s 778ms/step - loss: 0.4133 - acc: 0.8118 - val_loss: 0.4293 - val_acc: 0.8060
Epoch 8/10
90/90 [==============================] - 70s 776ms/step - loss: 0.3875 - acc: 0.8272 - val_loss: 0.4329 - val_acc: 0.8147
Epoch 9/10
90/90 [==============================] - 70s 781ms/step - loss: 0.3614 - acc: 0.8401 - val_loss: 0.4178 - val_acc: 0.8100
Epoch 10/10
90/90 [==============================] - 70s 779ms/step - loss: 0.3403 - acc: 0.8508 - val_loss: 0.3820 - val_acc: 0.8247
```

Accuracy score comparisons for 10 epochs; with 3 dropout layers 1-7

After training the models I wanted to see how the model 1-6 was doing on a data instance. It was predicting for a picture of three cats an 87 percent probability that it was a picture of a cat 1-8 while for a dog picture it was predicting a 12.8 percent probability that the dog was a cat as seem in 1-9. I then tried model 1-7 where it was getting a 0.012 percent probability cat for an image of a cat while for dog it was getting a 0.044 percent probability cat for an image of a dog. I think part of the discrepancy was that the model only had around an 85% accuracy and 82% for test/val data so it was not going to get a perfect prediction as it was not around 100 percent, the predictions are seen in 1-10, 1-11. For this analysis then, it was better to choose the model with 3 dropout layers.


```
predictions[2021]
array([0.87], dtype=float32)
```

Cat Prediction for Model in 1-6 (1-8)


```
predictions[2020]
array([0.128], dtype=float32)
```


```
array([0.012], dtype=float32)
```

Dog Prediction for Model in 1-6 (1-9) *Cat Prediction for Model in 1-7 (1-10)*


```
array([0.044], dtype=float32)
```

Dog Prediction for Model in 1-7 (1-11)

Implementation and Programming

For this assignment the **Keras and Tensorflow Packages** 2.0.0 or above was really important. I first imported these packages as seen below and made sure my version was correct for the packages as seen in 1-12. My Tensorflow version was at 2.3.0. and my Keras version was at 2.4.0. To get rid of warning I used defined a function which said to ignore warnings by writing **pass** keyword. After explicitly ignoring the warnings, to use google drive workspace I had to call **drive.mount()** and pass in the path to where I wanted to save the data set. After mounting the workspace to the drive, I downloaded the datset using **!wget** , a linux command, and gave the download link which was through the Microsoft website. I saved the zip file in the /tmp directory under the name cats-and-dogs.zip. I used the method **zip.ref.extractall()** to unzip the file. Zipref was a variable defined to explicitly tell Python the type of file and where it was and to open it using the zipfile.ZipFile() statement. After unzipping, I had to use .**close()** to close the reference to the zip file. I then used **os.listdir()** to list what was in the unzipped Cats and Dog dataset. I used **len()** function to get the number of files in the directory or folder. I wanted to create a barplot representing the number of items. I used **ax.bar()** to plot what len() function called back. I used the **PIL package, and os package** to plot the image dimensions of each image as seen in 1-2 and 1-3 above. I got the dimensions using **im.size** after using **open()** function to open each image and save the opened image via **im** variable. I used a **for loop** to get each file and use a **try and error statement to catch errors.**

After examining the dimensions, I then made a training and testing directories which resulted in cat and dog directories. Particularly **os.mkdir from os package** was used to make the directories where first the training and testing directories were first created and then the Cats and Dogs directories were created in both training and testing. **A try except block** was used to catch OS errors. I then split the data using a custom made function. I used **random.sample ()** to shuffle the data at random. This sample method was imported using **random package** as seen below. The method resulted in 22,500 images for train directory and 2,500 images in testing directory. The split was .91 to .9 split which was designed as **SPLIT_SIZE** in the code.

I used **len() and os package listdir()** to get sizes of each split. I then wanted to see images to get an idea what was in the dogs part of the data and cats part of the data. The way I did this was created a variable which defined the path to the images. I used **plt.figure()** to get the size of how big to make matrix to plot images. I then use **for range loop** to get a total of 9 images from cats and 9 images from dog. I used **subplot()** to create a subplot which is in **matplotlib package.** I then reference a file name, use the **display packag**e to read in the file. And used **matplotlib package** to show the image in the subplot. After looking at the images it was time to utilize **tensorflow and keras packages**.

```
# Helper libraries
import os
import zipfile
import random
import datetime
from packaging import version
import numpy as np
from IPython.display import Image, display
import matplotlib.pyplot as plt
from matplotlib import pyplot
from matplotlib.image import imread

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from shutil import copyfile
#from plot_keras_history import plot_history
```

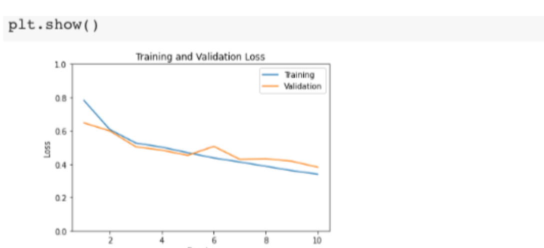
Import Statements 1-12

The Convolution Layers had 16 filters in them with a size of 3x3 representing the receptor field size. The Relu function was used. The way to call the Convolution Layer was **tf.keras.layers.Conv2D(), where** the first layer had the input shape of 150x150 X 3. There were 2 other convolution layers involved which involved 32 and 64 filter respectively. For MaxPooling, one would have to call **tf.keras.layers.MaxPooling2D(),** and the size it would summarize the convolution layer to is 2x2. The Dropout Layer would be called by **tf.keras.layers.Dropout() where 0.2 would be passed in representing 20% dropout.** Other layers that were called were Dense which was used for the output layer with Sigmoid Function explicitly stated representing Binary Classification. After defining the models one could see the model diagram by **model.summary() and plot_model() from keras package.** I then compiled the model using **optimizer - “adam” and loss function “binary crossentropy” and Accuarcy as the metric.** I used **model.compile()** to compile the model and used **model.fit() to train the model with 10 epochs**. After training a dictionary was created with metrics such as loss, and accuracy. The test set was used to validate on. After getting the results I plotted them using **matplotlib package and plot() function.** I then used model.predict() to get an array of probability predictions. I also plotted the associated image that the prediction was linked to.

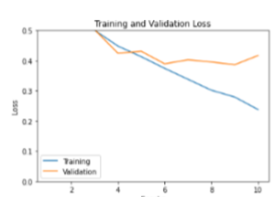
Exposition, problem description, Management recommendation

The goal of this analysis was to use **Convolution Neural Networks** to do **Binary Classification** on the Dogs and Cats dataset taken directly from Microsoft. Another goal that was involved was finding if I could find other models that explained the data better. As explained above Convolution Neural Networks are used as an alternative to Dense Neural Networks as they a good at distinguishing distance spatial features [2]. They are also used for 2D data such as the image data involved here and finding correlations between neighboring pixels [2].

After finding the results in 1-6 and 1-7 I noticed that 1-6 was overfitting more than 1-7 which involved 3 dropout layers. With this information I was able to determine that 1-7 was the best model as it had a margin smaller than model 1-6. The accuracy scores for Train data was 0.9015 and test/val Accuracy was 0.8327 for model 1-6 which involved 1 dropout layer of 20 percent. This was huge in comparison to 1-7 which had an accuracy for train data of 0.8508 and a test/val accuracy 0.8247 which involved 3 dropout layer of 20 percent. One can also see this by the gaps in the losses between the two models below in 1-13 and 1-14. **Therefore, to Management I recommend Model 1-7 which involved 3 2D Convolution Layers with 16, 32, and 64 filters, 3 Max Pooling Layers, 3 Dropout layers, 1 Dense Layer, and an output layer using Sigmoid Function.**



Loss for Model in 1-7 (1-13)



Loss for Model in 1-6 (1-14)

References

[1] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media.

[2] Srinivasan, S. (2020). *Week 7 Lecture CNN* [Slides]. Canvas. https://canvas.northwestern.edu/courses/125893/files/9362508/download?wrap=1

[3] Srinivasan, S. (2020b, October 27). *Sync Session CNN* [Slides]. Canvas. https://canvas.northwestern.edu/courses/125893/modules/items/1692263

Appendix

```
# Helper libraries
import os
import zipfile
import random
import datetime
from packaging import version
import numpy as np
from IPython.display import Image, display
import matplotlib.pyplot as plt
from matplotlib import pyplot
from matplotlib.image import imread

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from shutil import copyfile
#from plot_keras_history import plot_history
```

```
%matplotlib inline
np.set_printoptions(precision=3, suppress=True)
```

```
print("This notebook requires TensorFlow 2.0 or above")
print("TensorFlow version: ", tf.__version__)
assert version.parse(tf.__version__).release[0] >=2
```

```
This notebook requires TensorFlow 2.0 or above
TensorFlow version:  2.3.0
```

```
print("Keras version: ", keras.__version__)
```

```
Keras version:  2.4.0
```

```
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
# download dataset
!wget --no-check-certificate \
    "https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kag
    -O "/tmp/cats-and-dogs.zip"
local_zip = '/tmp/cats-and-dogs.zip'
zip_ref   = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp')
zip_ref.close()
```

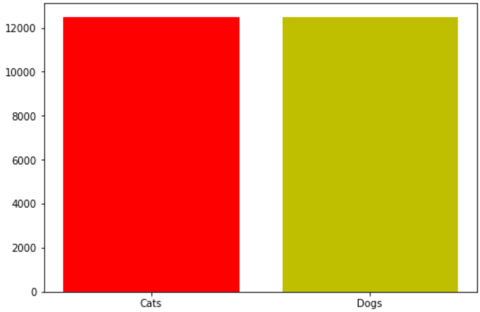
```
--2020-10-31 00:55:51--  https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs
Resolving download.microsoft.com (download.microsoft.com)... 23.196.32.25, 2600:1408:5c00:39b::e59, 2600:1408:5c00:3ad::e59,
Connecting to download.microsoft.com (download.microsoft.com)|23.196.32.25|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 824894548 (787M) [application/octet-stream]
Saving to: '/tmp/cats-and-dogs.zip'

/tmp/cats-and-dogs. 100%[===================>] 786.68M  88.7MB/s    in 8.9s
```

```
2020-10-31 00:56:00 (88.1 MB/s) - '/tmp/cats-and-dogs.zip' saved [824894548/824894548]
```

```
# print quantity of dogs and cats
print(len(os.listdir('/tmp/PetImages/Cat/')))
print(len(os.listdir('/tmp/PetImages/Dog/')))
```

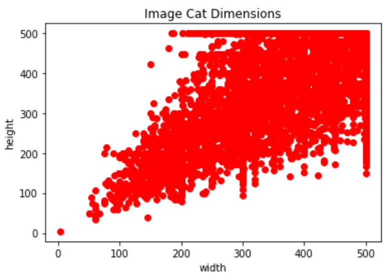
```
12501
12501
```

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
CATSNUM = len(os.listdir('/tmp/PetImages/Cat/'))
DOGSNUM = len(os.listdir('/tmp/PetImages/Dog/'))
barlist = ax.bar(['Cats', 'Dogs'], [CATSNUM, DOGSNUM])
barlist[0].set_color('r')
barlist[1].set_color('y')
plt.show()
```




```
from PIL import *
for filename in os.listdir('/tmp/PetImages/Cat/'):
  file = '/tmp/PetImages/Cat/' + filename
  try:
    if os.path.getsize(file) > 0:
      im = Image.open(file)
  except UnidentifiedImageError:
      pass
  except:
      print(filename + " is zero length, so ignoring.")

  width, height = im.size
  plt.plot(width, height, 'ro')
plt.xlabel('width')
plt.ylabel('height')
plt.title('Image Cat Dimensions')
```

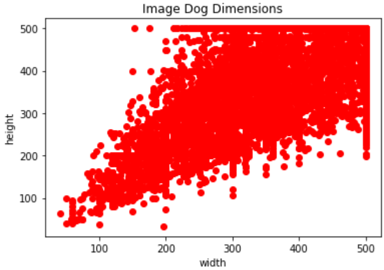
```
Text(0.5, 1.0, 'Image Cat Dimensions')
```




```
for filename in os.listdir('/tmp/PetImages/Dog/'):
  file = '/tmp/PetImages/Dog/' + filename
```

```
  try:
    if os.path.getsize(file) > 0:
      im = Image.open(file)
  except UnidentifiedImageError:
      pass
  except:
      print(filename + " is zero length, so ignoring.")

  width, height = im.size
  plt.plot(width, height, 'ro')
plt.xlabel('width')
plt.ylabel('height')
plt.title('Image Dog Dimensions')
```

```
Text(0.5, 1.0, 'Image Dog Dimensions')
```




```
try:
    os.mkdir('/tmp/cats_v_dogs')
    os.mkdir('/tmp/cats_v_dogs/training')
    os.mkdir('/tmp/cats_v_dogs/testing')
    os.mkdir('/tmp/cats_v_dogs/training/cats')
    os.mkdir('/tmp/cats_v_dogs/training/dogs')
    os.mkdir('/tmp/cats_v_dogs/testing/cats')
    os.mkdir('/tmp/cats_v_dogs/testing/dogs')
except OSError:
    pass
```

```
print(os.listdir('/tmp/cats_v_dogs/training/cats'))
```

```
[]
```

```
def split_data(SOURCE, TRAINING, TESTING, SPLIT_SIZE):
    files = []
    for filename in os.listdir(SOURCE):
        file = SOURCE + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " is zero length, so ignoring.")

    training_length = int(len(files) * SPLIT_SIZE)
    testing_length = int(len(files) - training_length)
    shuffled_set = random.sample(files, len(files))
    training_set = shuffled_set[0:training_length]
    testing_set = shuffled_set[-testing_length:]

    for filename in training_set:
        this_file = SOURCE + filename
        destination = TRAINING + filename
        copyfile(this_file, destination)
```

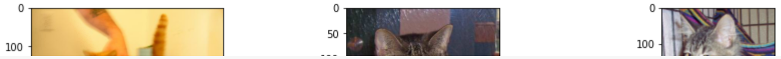
```
        copyfile(this_file, destination)

    for filename in testing_set:
        this_file = SOURCE + filename
        destination = TESTING + filename
        copyfile(this_file, destination)


CAT_SOURCE_DIR = '/tmp/PetImages/Cat/'
TRAINING_CATS_DIR = '/tmp/cats_v_dogs/training/cats/'
TESTING_CATS_DIR = '/tmp/cats_v_dogs/testing/cats/'
DOG_SOURCE_DIR = '/tmp/PetImages/Dog/'
TRAINING_DOGS_DIR = '/tmp/cats_v_dogs/training/dogs/'
TESTING_DOGS_DIR = '/tmp/cats_v_dogs/testing/dogs/'

split_size = .9
split_data(CAT_SOURCE_DIR, TRAINING_CATS_DIR, TESTING_CATS_DIR, split_size)
split_data(DOG_SOURCE_DIR, TRAINING_DOGS_DIR, TESTING_DOGS_DIR, split_size)
```

```
666.jpg is zero length, so ignoring.
11702.jpg is zero length, so ignoring.
```

```
print(len(os.listdir('/tmp/cats_v_dogs/training/cats/')))
print(len(os.listdir('/tmp/cats_v_dogs/training/dogs/')))
print(len(os.listdir('/tmp/cats_v_dogs/testing/cats/')))
print(len(os.listdir('/tmp/cats_v_dogs/testing/dogs/')))
```

```
11250
11250
1250
1250
```

```
fig = plt.figure(figsize = (15, 9))
Catfolder = '/tmp/PetImages/Cat/'
for i in range(9):
# define subplot
    pyplot.subplot(330 + 1 + i)
# define filename
    filename = Catfolder + str(i) + '.jpg'
# load image pixels
    image = imread(filename)
# plot raw pixel data
    pyplot.imshow(image)
# show the figure
pyplot.show()
```



```
fig = plt.figure(figsize = (15, 9))
Dogfolder = '/tmp/PetImages/Dog/'
for i in range(9):
# define subplot
    pyplot.subplot(330 + 1 + i)
# define filename
    filename = Dogfolder + str(i) + '.jpg'
# load image pixels
    image = imread(filename)
# plot raw pixel data
    pyplot.imshow(image)
# show the figure
pyplot.show()
```

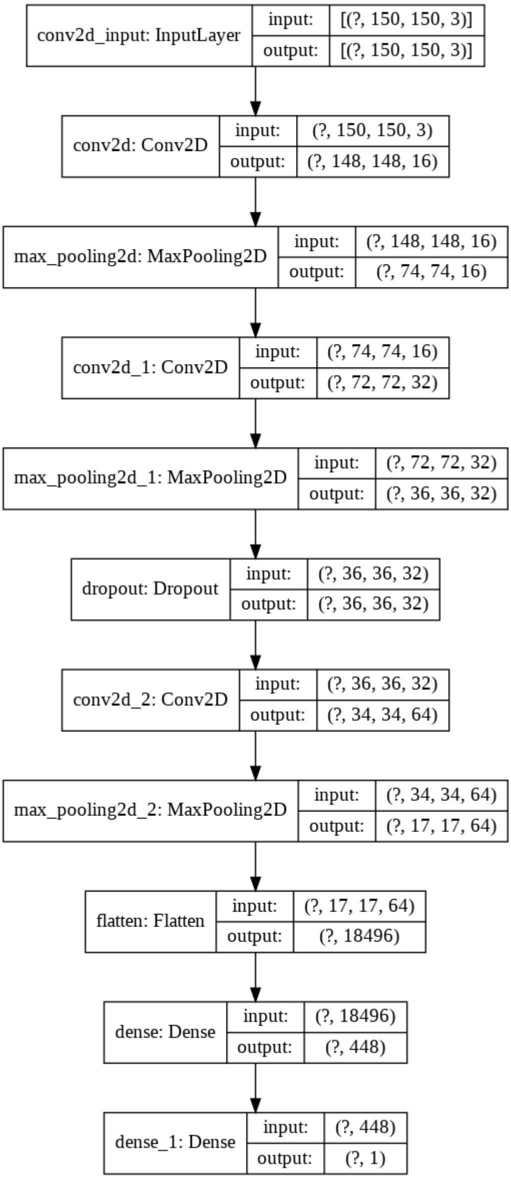
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
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
_________________________________________________________________
dropout (Dropout)            (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, 448)               8286656
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 449
=================================================================
Total params: 8,310,689
Trainable params: 8,310,689
Non-trainable params: 0
_________________________________________________________________
```

```
keras.utils.plot_model(model, "MCVD_model.png", show_shapes=True)
```




```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
TRAINING_DIR = "/tmp/cats_v_dogs/training/"
train_datagen = ImageDataGenerator(rescale=1.0/255.)
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=250,
                                                    class_mode='binary',
                                                    target_size=(150, 150))

VALIDATION_DIR = "/tmp/cats_v_dogs/testing/"
validation_datagen = ImageDataGenerator(rescale=1.0/255.)
validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                              batch_size=250,
                                                              class_mode='binary',
                                                              target_size=(150, 150))
```

```
Found 22498 images belonging to 2 classes.
Found 2500 images belonging to 2 classes.
```

```
history = model.fit(train_generator, epochs=10
                    ,validation_data=validation_generator
                    ,validation_steps=6
                    #,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patien
                   )
```

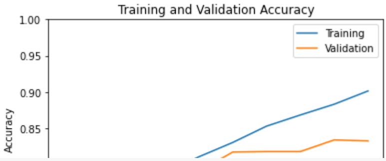
```
Epoch 1/10
90/90 [==============================] - 70s 778ms/step - loss: 0.7195 - acc: 0.5683 - val_loss: 0.6097 - val_acc: 0.6627
Epoch 2/10
90/90 [==============================] - 70s 778ms/step - loss: 0.5611 - acc: 0.7085 - val_loss: 0.5146 - val_acc: 0.7500
Epoch 3/10
90/90 [==============================] - 70s 777ms/step - loss: 0.5007 - acc: 0.7515 - val_loss: 0.5026 - val_acc: 0.7420
Epoch 4/10
90/90 [==============================] - 70s 775ms/step - loss: 0.4482 - acc: 0.7883 - val_loss: 0.4244 - val_acc: 0.7913
Epoch 5/10
90/90 [==============================] - 72s 796ms/step - loss: 0.4129 - acc: 0.8108 - val_loss: 0.4309 - val_acc: 0.7907
Epoch 6/10
90/90 [==============================] - 71s 789ms/step - loss: 0.3743 - acc: 0.8305 - val_loss: 0.3893 - val_acc: 0.8173
Epoch 7/10
90/90 [==============================] - 71s 788ms/step - loss: 0.3380 - acc: 0.8530 - val_loss: 0.4027 - val_acc: 0.8180
Epoch 8/10
90/90 [==============================] - 72s 795ms/step - loss: 0.3018 - acc: 0.8684 - val_loss: 0.3954 - val_acc: 0.8180
Epoch 9/10
90/90 [==============================] - 71s 786ms/step - loss: 0.2792 - acc: 0.8832 - val_loss: 0.3864 - val_acc: 0.8340
Epoch 10/10
90/90 [==============================] - 71s 791ms/step - loss: 0.2376 - acc: 0.9015 - val_loss: 0.4170 - val_acc: 0.8327
```

```
history_dict = history.history
history_dict.keys()
```

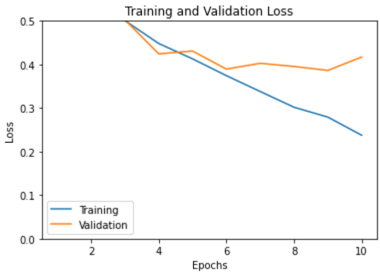
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
plt.plot(range(1, len(loss) + 1), history.history['loss'], label = 'Training')
plt.plot(range(1, len(val_loss) + 1), history.history['val_loss'], label = 'Validation')
plt.ylim([0.0, 0.5])
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
predictions = model.predict(validation_generator, validation_generator)
```

```
print('shape of preds: ', predictions.shape)
```

```
shape of preds:  (2500, 1)
```

```
listOfImageNames = ["/tmp/PetImages/Cat/2020.jpg"]
for imageName in listOfImageNames:
    display(Image.open(imageName))
    print(imageName)
```

/tmp/PetImages/Cat/2020.jpg

```
predictions[2021]
```

```
array([0.87], dtype=float32)
```

```
listOfImageNames = ["/tmp/PetImages/Dog/2020.jpg"]
for imageName in listOfImageNames:
    display(Image.open(imageName))
```

```
    print(imageName)
```

```
/tmp/PetImages/Dog/2020.jpg
```

```
predictions[2020]
```

```
array([0.128], dtype=float32)
```

MODEL 2

```
model2 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model2.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

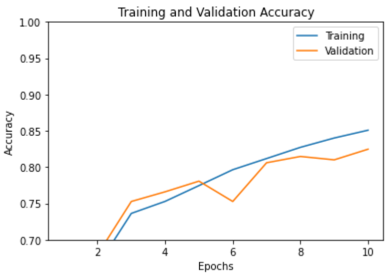
```
history = model2.fit(train_generator, epochs=10
                    ,validation_data=validation_generator
                    ,validation_steps=6
                    #,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patien
                    )
```

```
Epoch 1/10
90/90 [==============================] - 71s 793ms/step - loss: 0.7834 - acc: 0.5521 - val_loss: 0.6478 - val_acc: 0.6273
Epoch 2/10
90/90 [==============================] - 71s 785ms/step - loss: 0.6078 - acc: 0.6637 - val_loss: 0.5994 - val_acc: 0.6793
Epoch 3/10
90/90 [==============================] - 70s 781ms/step - loss: 0.5270 - acc: 0.7362 - val_loss: 0.5046 - val_acc: 0.7527
Epoch 4/10
90/90 [==============================] - 71s 789ms/step - loss: 0.5024 - acc: 0.7528 - val_loss: 0.4840 - val_acc: 0.7660
Epoch 5/10
90/90 [==============================] - 71s 793ms/step - loss: 0.4690 - acc: 0.7745 - val_loss: 0.4531 - val_acc: 0.7807
Epoch 6/10
90/90 [==============================] - 70s 775ms/step - loss: 0.4374 - acc: 0.7964 - val_loss: 0.5071 - val_acc: 0.7527
Epoch 7/10
90/90 [==============================] - 70s 778ms/step - loss: 0.4133 - acc: 0.8118 - val_loss: 0.4293 - val_acc: 0.8060
```

```
Epoch 8/10
90/90 [==============================] - 70s 776ms/step - loss: 0.3875 - acc: 0.8272 - val_loss: 0.4329 - val_acc: 0.8147
Epoch 9/10
90/90 [==============================] - 70s 781ms/step - loss: 0.3614 - acc: 0.8401 - val_loss: 0.4178 - val_acc: 0.8100
Epoch 10/10
90/90 [==============================] - 70s 779ms/step - loss: 0.3403 - acc: 0.8508 - val_loss: 0.3820 - val_acc: 0.8247
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
plt.plot(range(1, len(loss) + 1), history.history['loss'], label = 'Training')
plt.plot(range(1, len(val_loss) + 1), history.history['val_loss'], label = 'Validation')
plt.ylim([0.0, 1.0])
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

MODEL 3

```
model3 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
```

```
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model3.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
history = model3.fit(train_generator, epochs=10
                    ,validation_data=validation_generator
                    ,validation_steps=6
                    ,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patienc
                   )
```

```
Epoch 1/10
90/90 [==============================] - 70s 780ms/step - loss: 0.7780 - acc: 0.5683 - val_loss: 0.6803 - val_acc: 0.5640
Epoch 2/10
90/90 [==============================] - 69s 769ms/step - loss: 0.5674 - acc: 0.7053 - val_loss: 0.5764 - val_acc: 0.6940
Epoch 3/10
90/90 [==============================] - 70s 775ms/step - loss: 0.5046 - acc: 0.7522 - val_loss: 0.5544 - val_acc: 0.7053
Epoch 4/10
90/90 [==============================] - 69s 771ms/step - loss: 0.4650 - acc: 0.7802 - val_loss: 0.5028 - val_acc: 0.7467
Epoch 5/10
90/90 [==============================] - 69s 766ms/step - loss: 0.4328 - acc: 0.7963 - val_loss: 0.5622 - val_acc: 0.7253
Epoch 6/10
90/90 [==============================] - 69s 763ms/step - loss: 0.3968 - acc: 0.8213 - val_loss: 0.4453 - val_acc: 0.7807
Epoch 7/10
90/90 [==============================] - 69s 765ms/step - loss: 0.3704 - acc: 0.8345 - val_loss: 0.4478 - val_acc: 0.7987
Epoch 8/10
90/90 [==============================] - 69s 767ms/step - loss: 0.3442 - acc: 0.8483 - val_loss: 0.4445 - val_acc: 0.8013
Epoch 9/10
90/90 [==============================] - 69s 762ms/step - loss: 0.3055 - acc: 0.8666 - val_loss: 0.4186 - val_acc: 0.8127
Epoch 10/10
90/90 [==============================] - 69s 763ms/step - loss: 0.2781 - acc: 0.8843 - val_loss: 0.4927 - val_acc: 0.7947
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

MODEL 4

```
model4 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
```

```
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

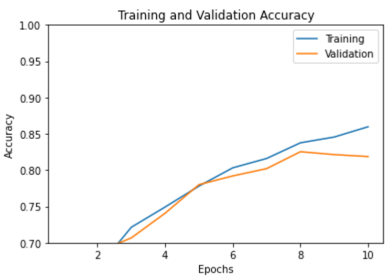
```
model4.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
history = model4.fit(train_generator, epochs=10
                    ,validation_data=validation_generator
                    ,validation_steps=6
                    ,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patienc
                   )
```

```
Epoch 1/10
90/90 [==============================] - 69s 771ms/step - loss: 0.7885 - acc: 0.5428 - val_loss: 0.6628 - val_acc: 0.6100
Epoch 2/10
90/90 [==============================] - 69s 764ms/step - loss: 0.6097 - acc: 0.6647 - val_loss: 0.5838 - val_acc: 0.6873
Epoch 3/10
90/90 [==============================] - 66s 731ms/step - loss: 0.5474 - acc: 0.7213 - val_loss: 0.5459 - val_acc: 0.7067
Epoch 4/10
90/90 [==============================] - 68s 757ms/step - loss: 0.5117 - acc: 0.7492 - val_loss: 0.5130 - val_acc: 0.7407
Epoch 5/10
90/90 [==============================] - 67s 749ms/step - loss: 0.4676 - acc: 0.7780 - val_loss: 0.4743 - val_acc: 0.7800
Epoch 6/10
90/90 [==============================] - 68s 751ms/step - loss: 0.4296 - acc: 0.8031 - val_loss: 0.4318 - val_acc: 0.7920
Epoch 7/10
90/90 [==============================] - 68s 751ms/step - loss: 0.4042 - acc: 0.8160 - val_loss: 0.4237 - val_acc: 0.8020
Epoch 8/10
90/90 [==============================] - 67s 748ms/step - loss: 0.3667 - acc: 0.8376 - val_loss: 0.4002 - val_acc: 0.8253
Epoch 9/10
90/90 [==============================] - 67s 745ms/step - loss: 0.3446 - acc: 0.8455 - val_loss: 0.3957 - val_acc: 0.8213
Epoch 10/10
90/90 [==============================] - 67s 742ms/step - loss: 0.3227 - acc: 0.8597 - val_loss: 0.4089 - val_acc: 0.8187
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



MODEL 5

```
model5 = tf.keras.models.Sequential([
```

```
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

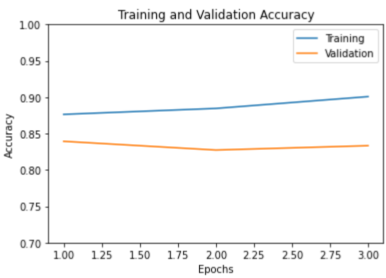
```
model5.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
history = model4.fit(train_generator, epochs=10
                    ,validation_data=validation_generator
                    ,validation_steps=6
                    ,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patienc
                   )
```

```
Epoch 1/10
90/90 [==============================] - 68s 754ms/step - loss: 0.2933 - acc: 0.8764 - val_loss: 0.3799 - val_acc: 0.8393
Epoch 2/10
90/90 [==============================] - 67s 744ms/step - loss: 0.2697 - acc: 0.8846 - val_loss: 0.3912 - val_acc: 0.8273
Epoch 3/10
90/90 [==============================] - 67s 742ms/step - loss: 0.2367 - acc: 0.9008 - val_loss: 0.3954 - val_acc: 0.8333
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



MODEL 6

```
model6 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
```

```
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model6.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
history = model6.fit(train_generator, epochs=10
                    ,validation_data=validation_generator
                    ,validation_steps=6
                    ,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patienc
                   )
```

```
Epoch 1/10
90/90 [==============================] - 68s 757ms/step - loss: 0.7234 - acc: 0.5751 - val_loss: 0.6573 - val_acc: 0.6293
Epoch 2/10
90/90 [==============================] - 68s 759ms/step - loss: 0.5798 - acc: 0.6962 - val_loss: 0.6251 - val_acc: 0.6720
Epoch 3/10
90/90 [==============================] - 68s 758ms/step - loss: 0.5045 - acc: 0.7538 - val_loss: 0.5191 - val_acc: 0.7467
Epoch 4/10
90/90 [==============================] - 67s 747ms/step - loss: 0.4620 - acc: 0.7797 - val_loss: 0.4931 - val_acc: 0.7513
Epoch 5/10
90/90 [==============================] - 68s 751ms/step - loss: 0.4171 - acc: 0.8069 - val_loss: 0.4653 - val_acc: 0.7747
Epoch 6/10
90/90 [==============================] - 68s 756ms/step - loss: 0.3756 - acc: 0.8314 - val_loss: 0.4682 - val_acc: 0.7753
Epoch 7/10
90/90 [==============================] - 68s 754ms/step - loss: 0.3481 - acc: 0.8451 - val_loss: 0.4159 - val_acc: 0.8100
Epoch 8/10
90/90 [==============================] - 67s 749ms/step - loss: 0.3001 - acc: 0.8688 - val_loss: 0.4344 - val_acc: 0.7900
Epoch 9/10
90/90 [==============================] - 67s 745ms/step - loss: 0.2585 - acc: 0.8922 - val_loss: 0.4633 - val_acc: 0.7887
Epoch 10/10
90/90 [==============================] - 67s 747ms/step - loss: 0.2214 - acc: 0.9087 - val_loss: 0.4368 - val_acc: 0.8153
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

MODEL 7

```
model7 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
```

```
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(448, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model7.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
history = model7.fit(train_generator, epochs=10
                     ,validation_data=validation_generator
                     ,validation_steps=6
                     ,callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_acc', patienc
                    )
```

```
Epoch 1/10
90/90 [==============================] - 68s 757ms/step - loss: 0.6747 - acc: 0.6076 - val_loss: 0.5598 - val_acc: 0.7133
Epoch 2/10
90/90 [==============================] - 68s 759ms/step - loss: 0.5283 - acc: 0.7362 - val_loss: 0.4873 - val_acc: 0.7813
Epoch 3/10
90/90 [==============================] - 67s 743ms/step - loss: 0.4543 - acc: 0.7855 - val_loss: 0.4501 - val_acc: 0.7787
Epoch 4/10
90/90 [==============================] - 66s 736ms/step - loss: 0.4039 - acc: 0.8165 - val_loss: 0.4236 - val_acc: 0.8027
Epoch 5/10
90/90 [==============================] - 66s 735ms/step - loss: 0.3666 - acc: 0.8377 - val_loss: 0.4011 - val_acc: 0.8100
Epoch 6/10
90/90 [==============================] - 66s 738ms/step - loss: 0.3217 - acc: 0.8595 - val_loss: 0.4012 - val_acc: 0.8227
Epoch 7/10
90/90 [==============================] - 66s 734ms/step - loss: 0.2759 - acc: 0.8826 - val_loss: 0.4162 - val_acc: 0.8193
Epoch 8/10
90/90 [==============================] - 66s 737ms/step - loss: 0.2309 - acc: 0.9043 - val_loss: 0.4454 - val_acc: 0.8147
Epoch 9/10
90/90 [==============================] - 67s 741ms/step - loss: 0.1901 - acc: 0.9222 - val_loss: 0.4980 - val_acc: 0.7933
Epoch 10/10
90/90 [==============================] - 67s 748ms/step - loss: 0.1480 - acc: 0.9441 - val_loss: 0.4949 - val_acc: 0.8160
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(range(1, len(acc) + 1), history.history['acc'], label = 'Training')
plt.plot(range(1, len(val_acc) + 1), history.history['val_acc'], label = 'Validation')
plt.ylim([0.7, 1.0])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```