```
import numpy as np
import matplotlib.pyplot as plt

import os
import shutil
```

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

We will split cat and dog images and give them their own directories training_set_cats
look for files that start with dog. This process will take a few hours as we need to proc

```
source = os.listdir('/content/drive/My Drive/Colab Notebooks/train/'

dst_1='/content/drive/My Drive/Colab Notebooks/training_dogs/'
dst_2='/content/drive/My Drive/Colab Notebooks/training_cats/'
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/train')
```

```
for file in source:
  if file.startswith('dog.'):
    shutil.copy(file, dst_1)
  else:
    shutil.copy(file, dst_2)
```

```
list = os.listdir('/content/drive/My Drive/Colab Notebooks/training_c
number_files = len(list)
print(number_files)
```

No we will set aside our training set of 1000 dogs and 1000 cats and store them in trai

```
source_2 = os.listdir('/content/drive/My Drive/Colab Notebooks/train
source_3 = os.listdir('/content/drive/My Drive/Colab Notebooks/train
dst_3='/content/drive/My Drive/Colab Notebooks/training_2000/trainin
dst_4='/content/drive/My Drive/Colab Notebooks/training_2000/trainin
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_dogs')
```

```
for file in source_2:
  shutil.copy(file, dst_3)
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_cats')
```

```
for file in source_3:
  shutil.copy(file, dst_4)
```

No we will set aside our validation set of 500 dogs and 500 cats and store them in val_

```
source_4 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
source_5 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
dst_5='/content/drive/My Drive/Colab Notebooks/val_1000/val_dogs_500/
dst_6='/content/drive/My Drive/Colab Notebooks/val_1000/val_cats_500/
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_dogs')
```

```
for file in source_4:
  shutil.copy(file, dst_5)
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_cats')
```

```
for file in source_5:
  shutil.copy(file, dst_6)
```

```
list = os.listdir('/content/drive/My Drive/Colab Notebooks/training_2
number_files = len(list)
print(number_files)
```

No we will set aside our test set of 500 dogs and 500 cats in test_set_dogs_500 and te

```
source_6 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
source_7 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
dst_7='/content/drive/My Drive/Colab Notebooks/test_1000/test_dogs_50
dst_8='/content/drive/My Drive/Colab Notebooks/test_1000/test_cats_50
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_dogs/')
```

```
for file in source_6:
  shutil.copy(file, dst_7)
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_cats/')
```

```
for file in source_7:
  shutil.copy(file, dst_8)
```

```
list = os.listdir('/content/drive/My Drive/Colab Notebooks/training_2
number_files = len(list)
print(number_files)
```

```
from keras import models
from keras import layers

model=models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(15
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Flatten())
#model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
from keras import optimizers

model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSpro
```

Generator allows you to loop over the training data in pieces(batches), avoiding large n

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255)
validation_datagen=ImageDataGenerator(rescale=1./255)

train_generator=train_datagen.flow_from_directory('/content/drive/My
validation_generator=validation_datagen.flow_from_directory('/content
```

Generators are objects that act as an iterator (for loop). However the generator yields b
the train and val generators will yield batches of inputs and targets indefinitely. Howeve
declaring the epoch is over. The steps_per_epoch argument takes care of this. In our c

```
history = model.fit_generator(train_generator, steps_per_epoch=100, e
acc=history.history['acc']
val_acc=history.history['val_acc']
loss=history.history['loss']
val_loss=history.history['val_loss']
```

```
epochs=range(1, len(acc)+1)
```

```
plt.plot(epochs, acc, 'bo', label='training acc')
plt.plot(epochs, val_acc, 'b', label='validation acc')
plt.title('training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'bo', label='training loss')
plt.plot(epochs, val_loss, 'b', label='validation loss')
plt.title('training and validation loss')
plt.legend()
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt

import os
import shutil
```

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

We will split cat and dog images and give them their own directories training_set_cats a
look for files that start with dog. This process will take a few hours as we need to proce

```
source = os.listdir('/content/drive/My Drive/Colab Notebooks/train/')

dst_1='/content/drive/My Drive/Colab Notebooks/training_dogs/'
dst_2='/content/drive/My Drive/Colab Notebooks/training_cats/'
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/train')
```

```
for file in source:
  if file.startswith('dog.'):
    shutil.copy(file, dst_1)
  else:
    shutil.copy(file, dst_2)
```

```
list = os.listdir('/content/drive/My Drive/Colab Notebooks/training_c
number_files = len(list)
print(number_files)
```

No we will set aside our training set of 1000 dogs and 1000 cats and store them in trair

```
source_2 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
source_3 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
dst_3='/content/drive/My Drive/Colab Notebooks/training_2000/training
dst_4='/content/drive/My Drive/Colab Notebooks/training_2000/training
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_dogs')
```

```
for file in source_2:
  shutil.copy(file, dst_3)
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_cats')
```

```
for file in source_3:
  shutil.copy(file, dst_4)
```

No we will set aside our validation set of 500 dogs and 500 cats and store them in val_s

```
source_4 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
source_5 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
dst_5='/content/drive/My Drive/Colab Notebooks/val_1000/val_dogs_500/
dst_6='/content/drive/My Drive/Colab Notebooks/val_1000/val_cats_500/
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_dogs')
```

```
for file in source_4:
  shutil.copy(file, dst_5)
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_cats')
```

```
for file in source_5:
  shutil.copy(file, dst_6)
```

```
list = os.listdir('/content/drive/My Drive/Colab Notebooks/training_2
number_files = len(list)
print(number_files)
```

No we will set aside our test set of 500 dogs and 500 cats in test_set_dogs_500 and te

```
source_6 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
source_7 = os.listdir('/content/drive/My Drive/Colab Notebooks/traini
dst_7='/content/drive/My Drive/Colab Notebooks/test_1000/test_dogs_50
```

```
dst_8='/content/drive/My Drive/Colab Notebooks/test_1000/test_cats_50
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_dogs/')
```

```
for file in source_6:
  shutil.copy(file, dst_7)
```

```
os.chdir('/content/drive/My Drive/Colab Notebooks/training_cats/')
```

```
for file in source_7:
  shutil.copy(file, dst_8)
```

```
list = os.listdir('/content/drive/My Drive/Colab Notebooks/training_2
number_files = len(list)
print(number_files)
```

```
from keras import models
from keras import layers

model=models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(15
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Flatten())
#model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
from keras import optimizers

model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSpro
```

Generator allows you to loop over the training data in pieces(batches), avoiding large m

Generator allows you to loop over the training data in pieces(batches), avoiding large m

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255)
validation_datagen=ImageDataGenerator(rescale=1./255)

train_generator=train_datagen.flow_from_directory('/content/drive/My
validation_generator=validation_datagen.flow_from_directory('/content
```

Generators are objects that act as an iterator (for loop). However the generator yields b
the train and val generators will yield batches of inputs and targets indefinitely. Howeve
declaring the epoch is over. The steps_per_epoch argument takes care of this. In our ca

```
history = model.fit_generator(train_generator, steps_per_epoch=100, e
acc=history.history['acc']
val_acc=history.history['val_acc']
loss=history.history['loss']
val_loss=history.history['val_loss']
```

```
epochs=range(1, len(acc)+1)
```

```
plt.plot(epochs, acc, 'bo', label='training acc')
plt.plot(epochs, val_acc, 'b', label='validation acc')
plt.title('training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='training loss')
plt.plot(epochs, val_loss, 'b', label='validation loss')
plt.title('training and validation loss')
plt.legend()
plt.show()
```