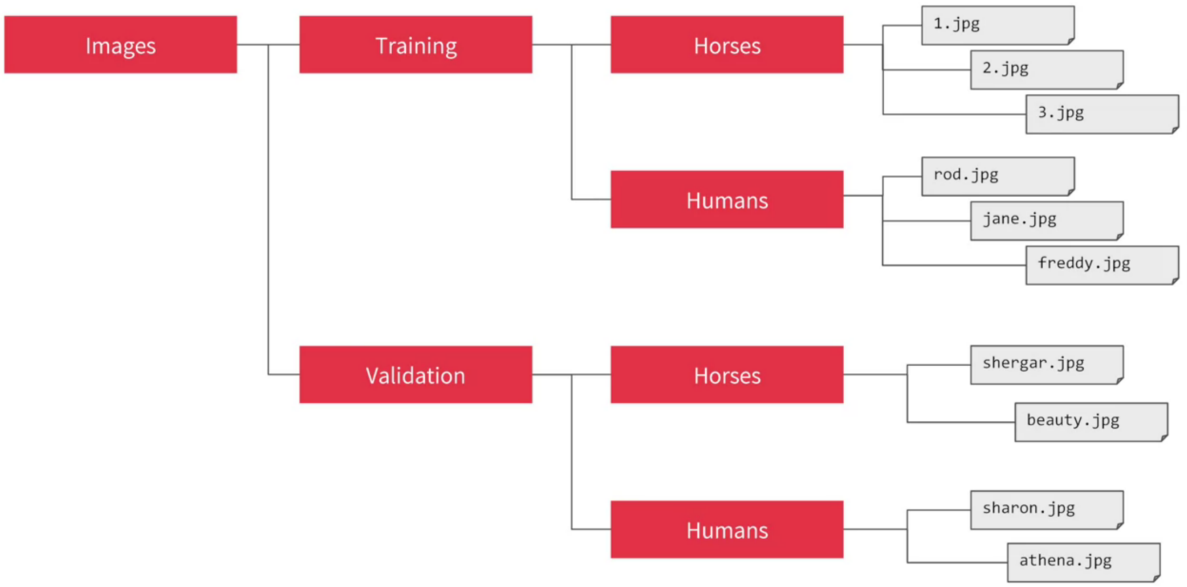
Week 4

Understanding Image Generator

- Previous image samples are all the same size and at the same location
- These images we will be using would be in different aspect ratios, different location and different sizes and even might have multiple subjects

- all you have to do to generate image is to point it at a directory



Defining a ConvNet to use complex images

- The layers of the algorithm

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu',
                           input_shape=(300, 300, 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(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

- summary of the method above

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 78400)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               40141312
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 40,165,409
Trainable params: 40,165,409
Non-trainable params: 0
```

- Compiling the model

```
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['acc'])
```

- because dealing with binary here, so this is using binary crossentropy instead
- RMSprop adjusts the learning rate
- Fitting generator

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=8,
      epochs=15,
      validation_data=validation_generator,
      validation_steps=8,
      verbose=2)
```

- using fit_generator because data using is generated with generator instead of dataset
- First parameter is the training generator set up earlier
- In order to load all the data in we need to use 8 epochs
- Epochs to train for is 15
- validation set is using the validation generator created earlier
- 8 steps so handle it by 32
- verbose specifies how much to display while training

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():

  # predicting images
  path = '/content/' + fn
  img = image.load_img(path, target_size=(300, 300))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)
  print(classes[0])
  if classes[0]>0.5:
    print(fn + " is a human")
  else:
    print(fn + " is a horse")
```

- The code above is using the trained data
- Some specific for Google Colab

Horse and Human identifier

```
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

class MyCallBack(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, log):
        if log.get('accuracy') > 0.999:
            print("\nReached 99.9 accuracy so cancelling training!")
            self.model.stop_training = True

callback = MyCallBack()

train_hourse_dir = os.path.join('C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/horse-
train_human_dir = os.path.join('C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/horse-o

train_hourse_labels = os.listdir(train_hourse_dir)
train_human_labels = os.listdir(train_human_dir)
# print(train_hourse_labels[0:10])
# print(train_human_labels[0:10])
# print('Total hourse Images', len(train_hourse_labels))
# print('Total human Images', len(train_human_labels))

# ---------------------------------------------------------- #
# Plotting plt images
# nrows = 4
# ncols = 4
# pic_index = 0
# fig = plt.gcf()
# fig.set_size_inches(ncols*4, nrows*4)
# pic_index += 28
# next_horse_pix = [os.path.join(train_hourse_dir, fname)
#                 for fname in train_hourse_labels[pic_index-8:pic_index]]
# next_human_pix = [os.path.join(train_human_dir, fname)
#                 for fname in train_human_labels[pic_index-8:pic_index]]
```

```
# for i, img_path in enumerate(next_horse_pix+next_human_pix):
#   # Set up subplot; subplot indices start at 1
#   sp = plt.subplot(nrows, ncols, i + 1)
#   sp.axis('Off') # Don't show axes (or gridlines)

#   img = mpimg.imread(img_path)
#   plt.imshow(img)

# plt.show()
# ------------------------------------------------------------ #

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy', optimizer=RMSprop(lr=0.001),metrics=['accuracy'])
train_datagen = ImageDataGenerator(rescale=1/255)
train_generator = train_datagen.flow_from_directory(
        'C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/horse-or-human/',
        target_size=(300, 300), batch_size=128, class_mode='binary')
history = model.fit(train_generator, steps_per_epoch=8, epochs=15, verbose=1, callbacks=[callback])
```

Horse and Human identifier with validation

```
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

class MyCallBack(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, log):
        if log.get('accuracy') > 0.999:
            print("\nReached 99.9 accuracy so cancelling training!")
            self.model.stop_training = True

callback = MyCallBack()

train_hourse_dir = os.path.join('C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/horse-
train_human_dir = os.path.join('C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/horse-o
val_hourse_dir = os.path.join('C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/validati
val_human_dir = os.path.join('C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/validatio


train_hourse_labels = os.listdir(train_hourse_dir)
train_human_labels = os.listdir(train_human_dir)
val_hourse_labels = os.listdir(val_hourse_dir)
val_human_labels = os.listdir(val_human_dir)


model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy', optimizer=RMSprop(lr=0.001),metrics=['accuracy'])
train_datagen = ImageDataGenerator(rescale=1/255)
train_generator = train_datagen.flow_from_directory(
        'C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/horse-or-human/',
```

```
        target_size=(300, 300), batch_size=128, class_mode='binary')
val_generator = train_datagen.flow_from_directory(
        'C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/horse or human/validation-horse-or-human/',
        target_size=(300, 300), batch_size=32, class_mode='binary')
history = model.fit(train_generator, steps_per_epoch=8, epochs=15, verbose=1, callbacks=[callback], validation_data=val_generat
```

Weekly test

```
import tensorflow as tf
import os
from os import path, getcwd, chdir
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import RMSprop


# GRADED FUNCTION: train_happy_sad_model
def train_happy_sad_model():
    # Please write your code only where you are indicated.
    # please do not remove # model fitting inline comments.

    DESIRED_ACCURACY = 0.999

    class myCallback(tf.keras.callbacks.Callback):
         def on_epoch_end(self, epoch, log={}):
             if log.get('accuracy') > DESIRED_ACCURACY:
                print('Reached 99.9% accuracy so cancelling training!')
                model.stop_training = True
    callbacks = myCallback()

    # This Code Block should Define and Compile the Model. Please assume the images are 150 X 150 in your implementation.
    model = tf.keras.models.Sequential([
       tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
       tf.keras.layers.MaxPool2D(2, 2),
       tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
       tf.keras.layers.MaxPool2D(2, 2),
       tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
       tf.keras.layers.MaxPool2D(2, 2),
       tf.keras.layers.Flatten(),
       tf.keras.layers.Dense(40, activation='relu'),
       tf.keras.layers.Dense(1, 'sigmoid')
    ])


    model.compile(loss='binary_crossentropy', optimizer=RMSprop(lr=0.001), metrics=['accuracy'])


    # This code block should create an instance of an ImageDataGenerator called train_datagen
    # And a train_generator by calling train_datagen.flow_from_directory



    train_datagen = ImageDataGenerator(rescale=1./255)

    # Please use a target_size of 150 X 150.
    train_generator = train_datagen.flow_from_directory(
        'C:/Users/Alex Lai.DESKTOP-AJOHRHM/Desktop/Deep Learning AI course/Week 4/test/happy-or-sad',
        target_size=(150, 150), batch_size=10, class_mode='binary'
    )
    # Expected output: 'Found 80 images belonging to 2 classes'

    # This code block should call model.fit_generator and train for
    # a number of epochs.
    # model fitting
    history = model.fit_generator(train_generator, steps_per_epoch=8, epochs=15, callbacks=[callbacks], verbose=1)
    # model fitting
    return history.history['accuracy'][-1]

# The Expected output: "Reached 99.9% accuracy so cancelling training!""
train_happy_sad_model()
```