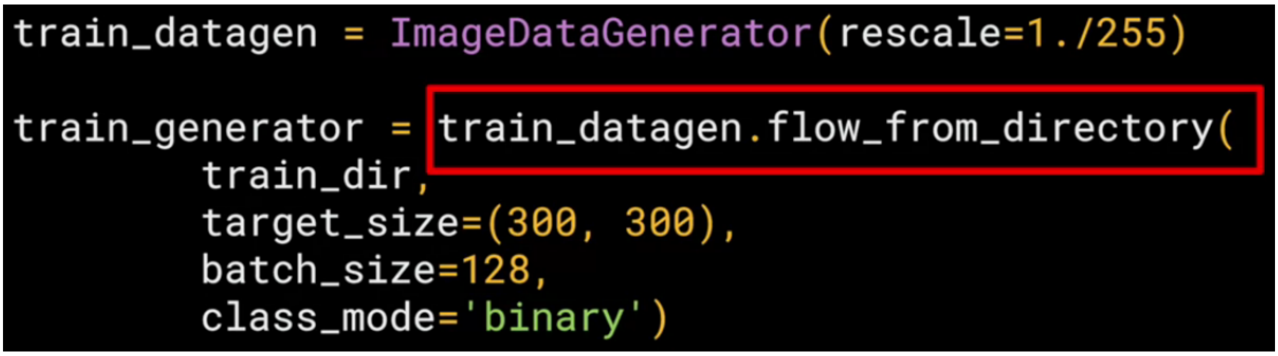
Introduction to TF for AI — Week 04 (Using real world images)

aakashgoel12.medium.com/introduction-to-tf-for-ai-week-04-using-real-world-images-a806bd68cbc2

February 4, 2021



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

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(300, 300),
        batch_size=128,
        class_mode='binary')
```

[Aakash Goel](#)

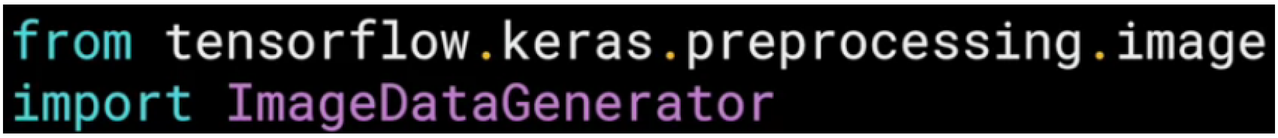
·Just now

Mobile App → Nuru (Detects disease in plants)

List of Cassava disease (5000 Images), single shot detector Model, TensorFlow Mobile net architecture

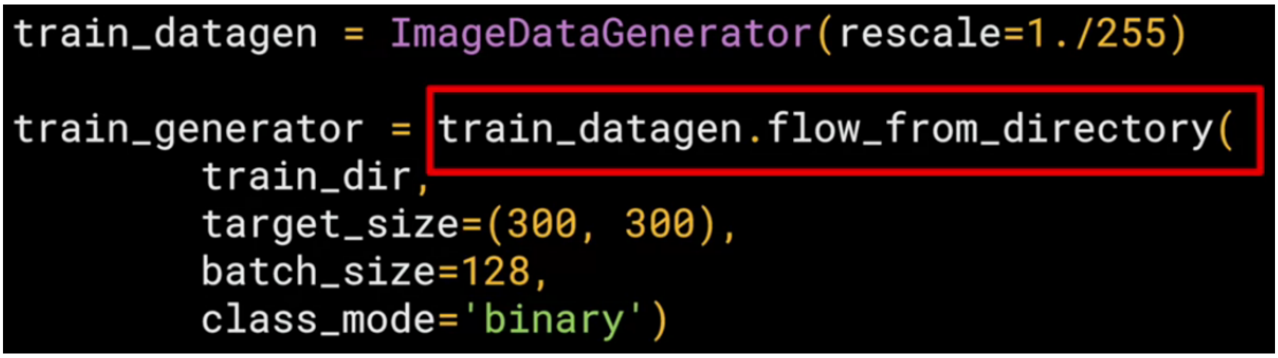
Horses Vs Humans

Image Generator in TensorFlow



```
from tensorflow.keras.preprocessing.image
import ImageDataGenerator
```

Training Generator

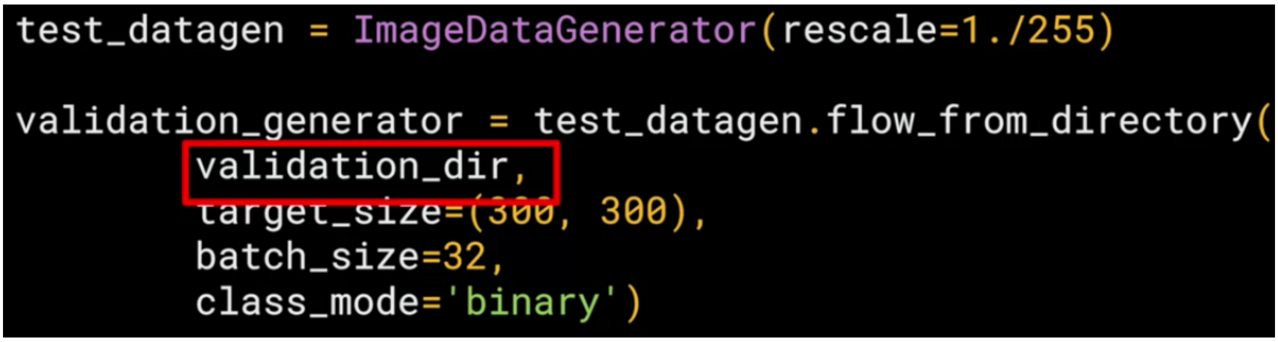


```
train_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(300, 300),
        batch_size=128,
        class_mode='binary')
```

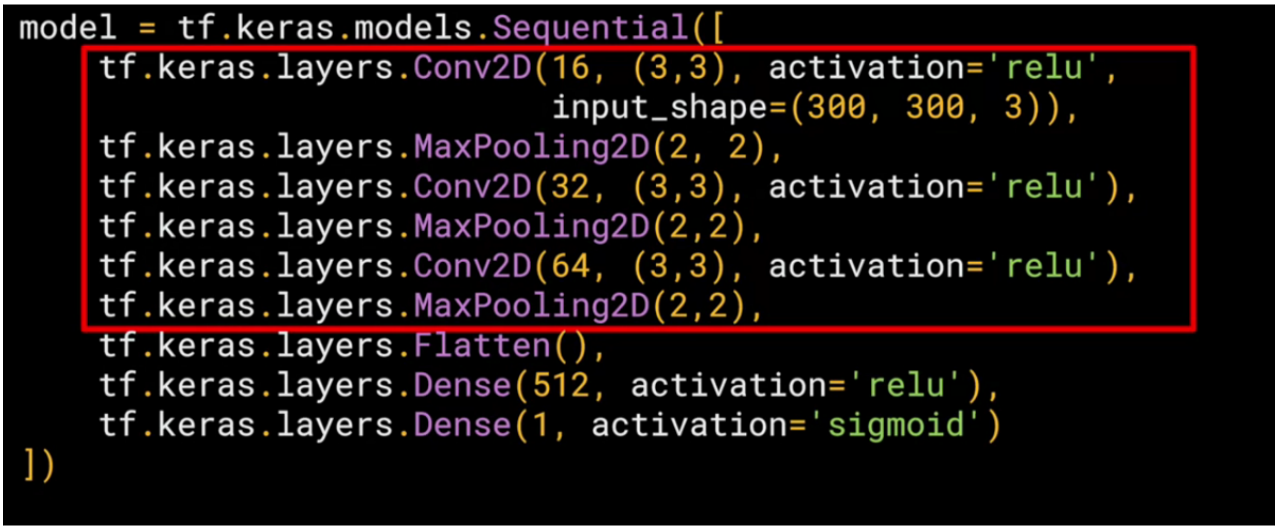
Target Size should be of same size to train Neural Network.

Validation Generator

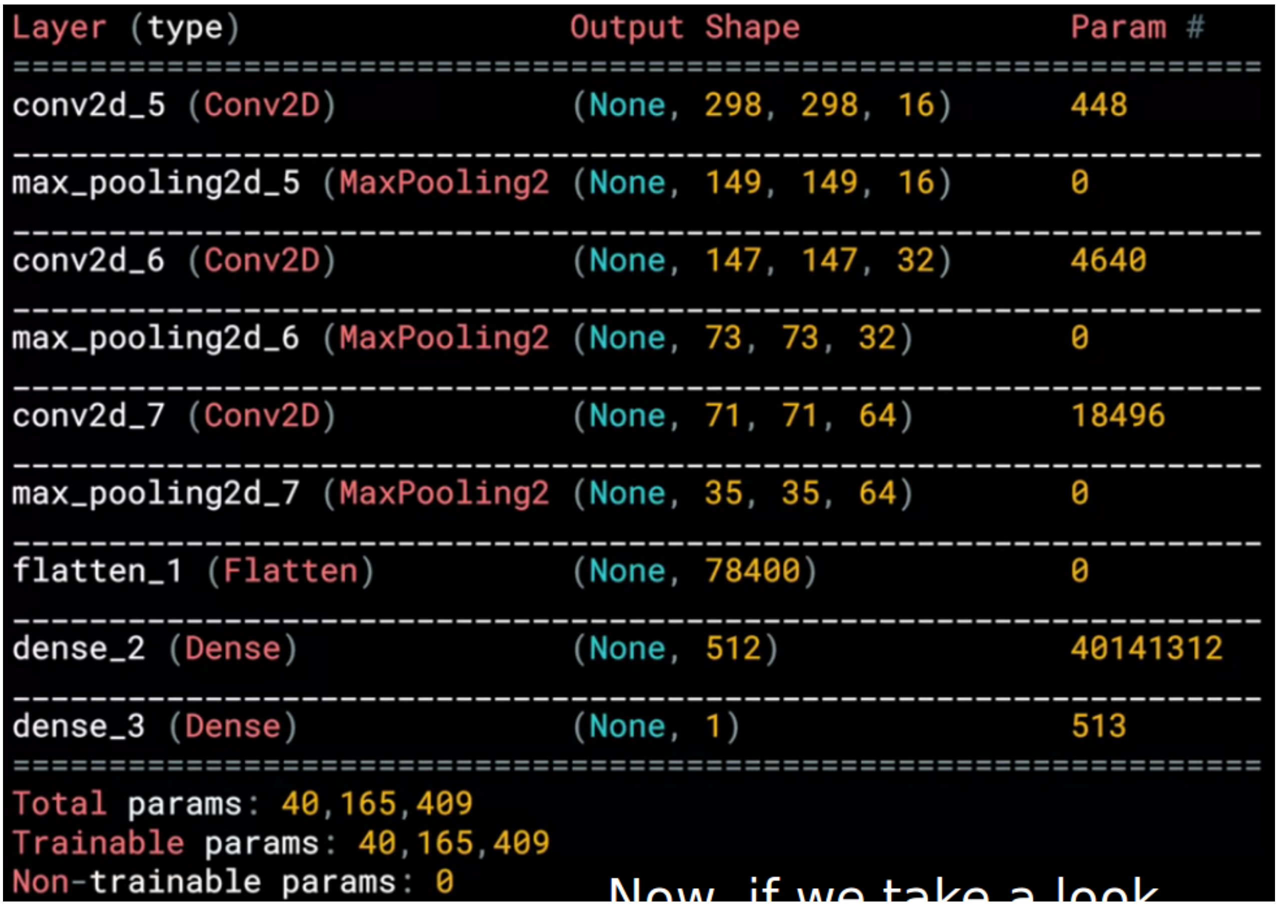


```
test_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(300, 300),
        batch_size=32,
        class_mode='binary')
```

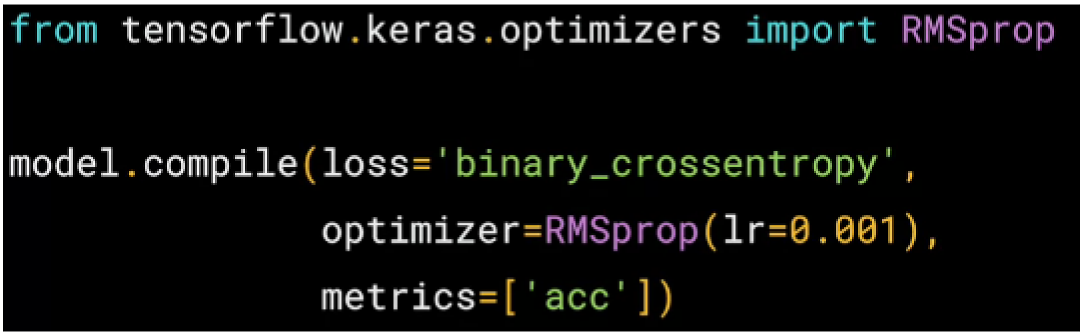


```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu',
                           input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (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')
])
```



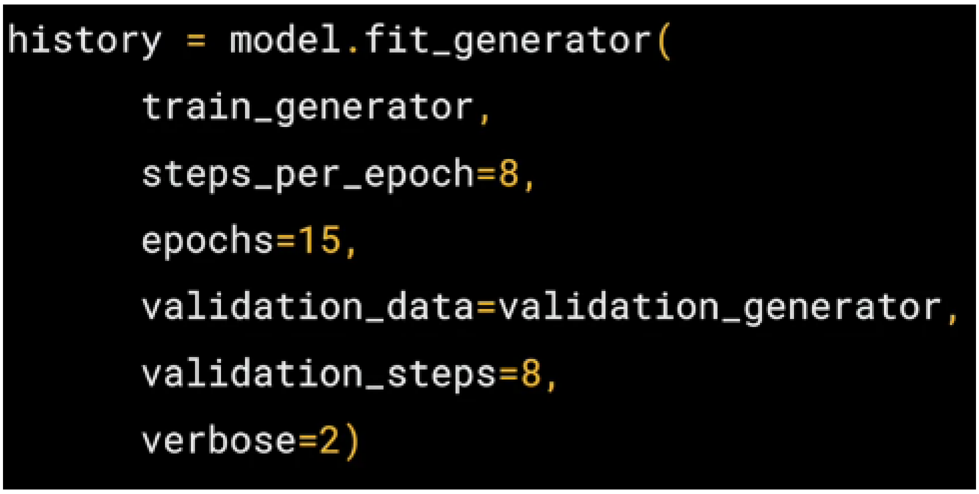
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 149, 149, 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
```

Now, if we take a look



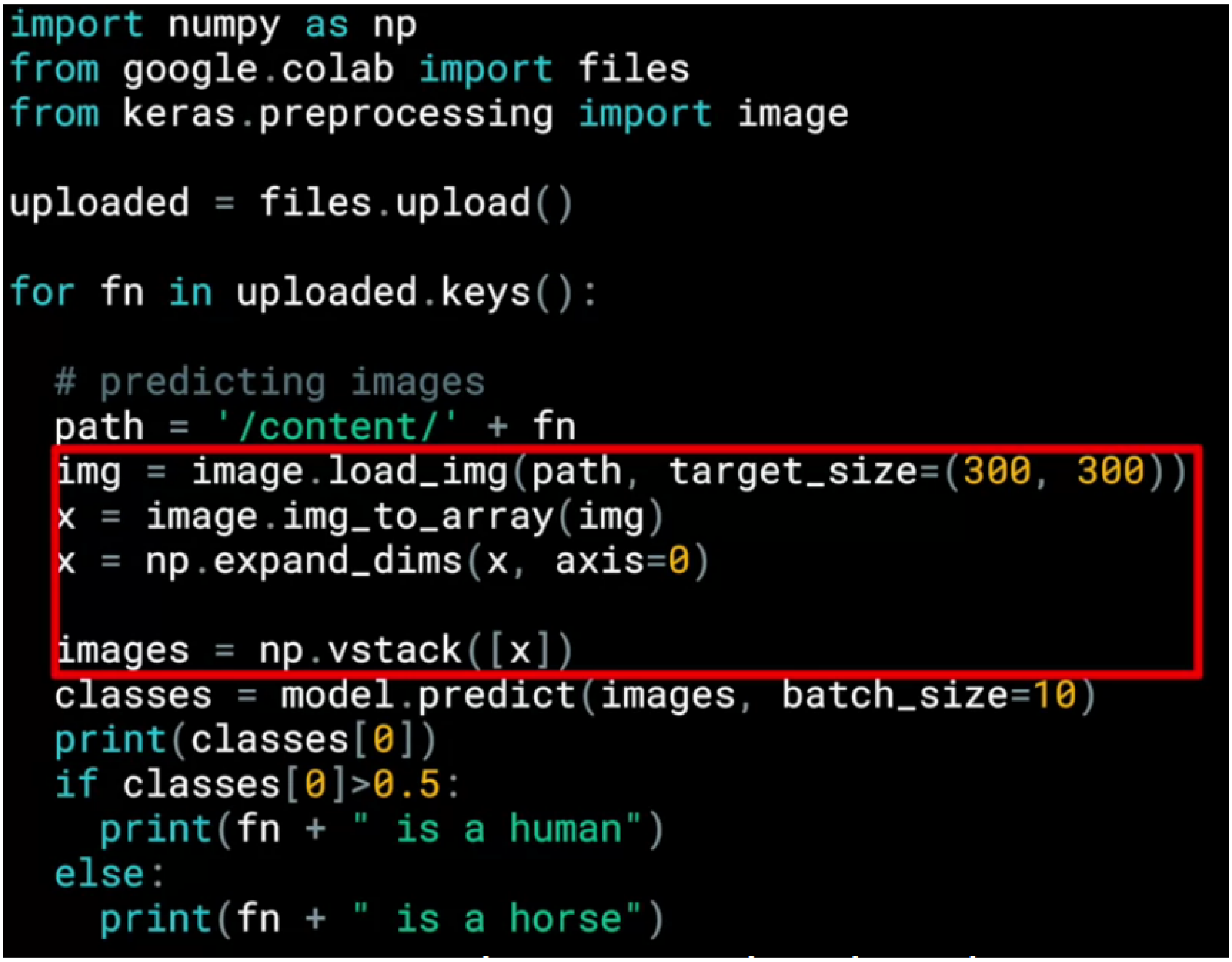
```
from tensorflow.keras.optimizers import RMSprop

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



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

steps_per_epoch = 8 (1024 images in training directory and 128 batch size used → Loading 128 images @ time. So 1024/128 = 8)



```
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")
```

Understand Binary Cross Entropy →
https://gombru.github.io/2018/05/23/cross_entropy_loss/

http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pd

https://pixabay.com → Free Images

Google Colaboratory

Edit description

colab.research.google.com

Google Colaboratory

Edit description

colab.research.google.com

Google Colaboratory

Edit description

colab.research.google.com

QUIZ

1. Using Image Generator, how do you label images?

○ TensorFlow figures it out from the contents

● It's based on the file name

○ You have to manually do it

○ It's based on the directory the image is contained in

! **Incorrect**

2. What method on the Image Generator is used to normalize the image?

○ normalize

● rescale

○ Rescale_image

○ normalize_image

3. How did we specify the training size for the images?

● The target_size parameter on the training generator

○ The target_size parameter on the validation generator

○ The training_size parameter on the training generator

○ The training_size parameter on the validation generator

✓ **Correct**

4. When we specify the input_shape to be (300, 300, 3), what does that mean?

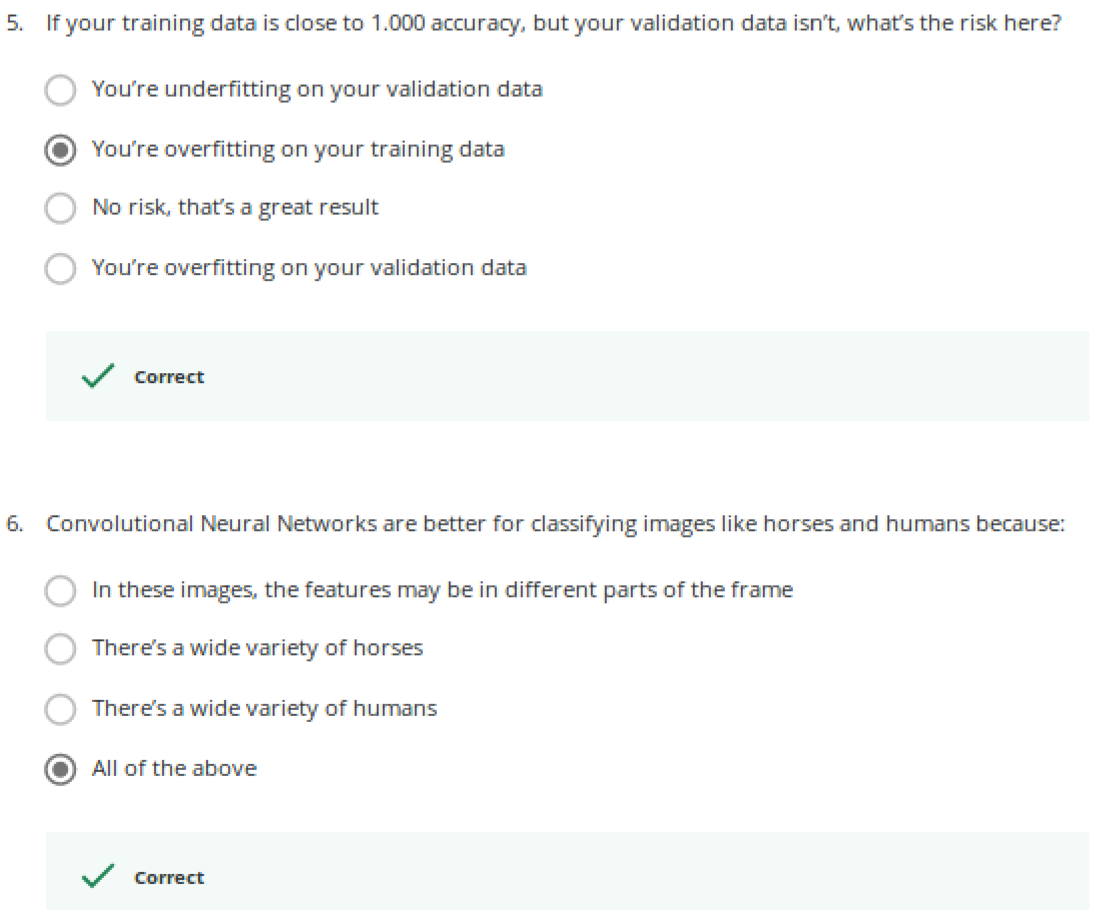
○ There will be 300 images, each size 300, loaded in batches of 3

○ Every Image will be 300x300 pixels, and there should be 3 Convolutional Layers

● Every Image will be 300x300 pixels, with 3 bytes to define color

○ There will be 300 horses and 300 humans, loaded in batches of 3

✓ **Correct**



5. If your training data is close to 1.000 accuracy, but your validation data isn't, what's the risk here?

○ You're underfitting on your validation data

● You're overfitting on your training data

○ No risk, that's a great result

○ You're overfitting on your validation data

✓ Correct

6. Convolutional Neural Networks are better for classifying images like horses and humans because:

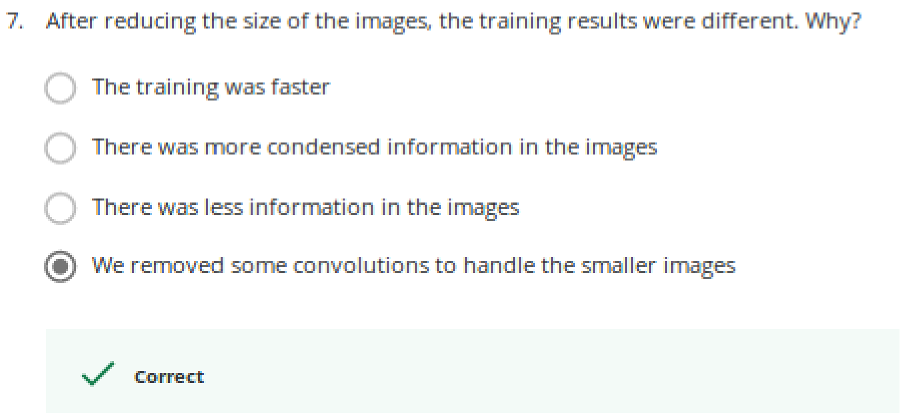
○ In these images, the features may be in different parts of the frame

○ There's a wide variety of horses

○ There's a wide variety of humans

● All of the above

✓ Correct



7. After reducing the size of the images, the training results were different. Why?

○ The training was faster

○ There was more condensed information in the images

○ There was less information in the images

● We removed some convolutions to handle the smaller images

✓ Correct

Assignment Solution

Below is code with a link to a happy or sad dataset which contains 80 images, 40 happy and 40 sad. Create a convolutional neural network that trains to 100% accuracy on these images, which cancels training upon hitting training accuracy of >.999

Hint -- it will work best with 3 convolutional layers.

```
import tensorflow as tf
import os
import zipfile
from os import path, getcwd, chdir

# DO NOT CHANGE THE LINE BELOW. If you are developing in a local
# environment, then grab happy-or-sad.zip from the Coursera Jupyter Notebook
# and place it inside a local folder and edit the path to that location
path = f"{getcwd()}/../tmp2/happy-or-sad.zip"

zip_ref = zipfile.ZipFile(path, 'r')
zip_ref.extractall("/tmp/h-or-s")
zip_ref.close()
```

```
train_dir = os.path.join("/tmp/h-or-s")
```

```python
# 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, logs = {}):
            if(logs.get('acc')>0.999):
                print("Hit training accuracy of 0.999")
                self.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(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(512,activation = 'relu'),
        tf.keras.layers.Dense(1,activation = 'sigmoid')])

    from tensorflow.keras.optimizers import RMSprop

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

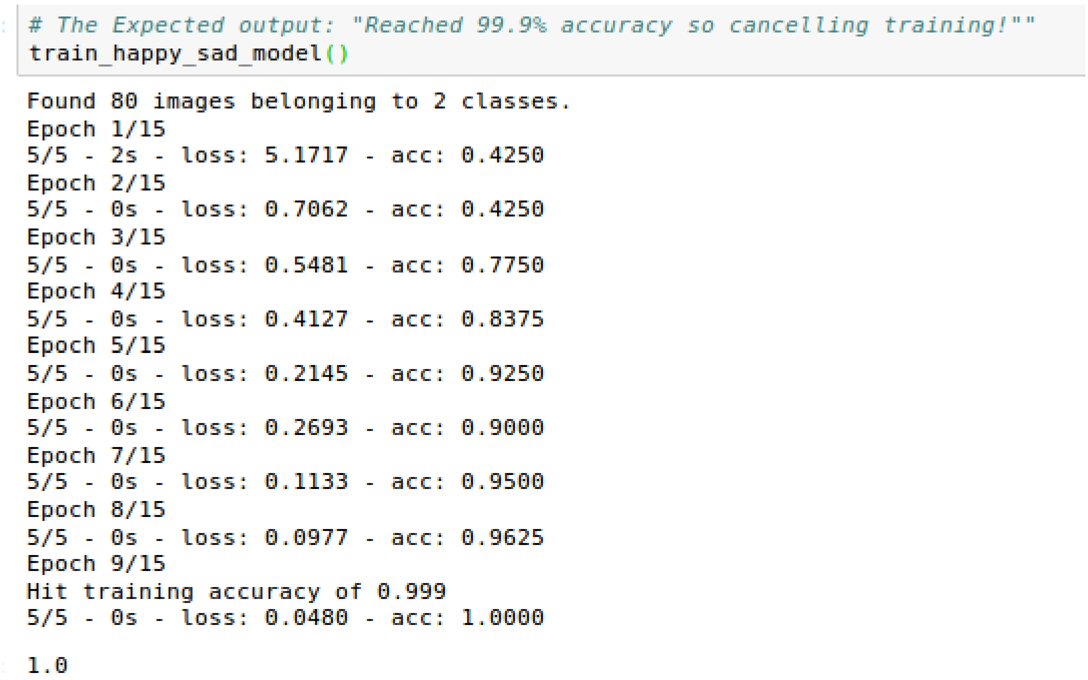
```python
# This code block should create an instance of an ImageDataGenerator called train_datagen
# And a train_generator by calling train_datagen.flow_from_directory

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255)

# Please use a target_size of 150 X 150.
train_generator = train_datagen.flow_from_directory(train_dir,\
                                                   target_size = (150,150),\
                                                  batch_size = 16,\
                                                  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 = 5, epochs = 15,\
                             callbacks = [callbacks], verbose = 2)
# model fitting
return history.history['acc'][-1]
```



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

```
Found 80 images belonging to 2 classes.
Epoch 1/15
5/5 - 2s - loss: 5.1717 - acc: 0.4250
Epoch 2/15
5/5 - 0s - loss: 0.7062 - acc: 0.4250
Epoch 3/15
5/5 - 0s - loss: 0.5481 - acc: 0.7750
Epoch 4/15
5/5 - 0s - loss: 0.4127 - acc: 0.8375
Epoch 5/15
5/5 - 0s - loss: 0.2145 - acc: 0.9250
Epoch 6/15
5/5 - 0s - loss: 0.2693 - acc: 0.9000
Epoch 7/15
5/5 - 0s - loss: 0.1133 - acc: 0.9500
Epoch 8/15
5/5 - 0s - loss: 0.0977 - acc: 0.9625
Epoch 9/15
Hit training accuracy of 0.999
5/5 - 0s - loss: 0.0480 - acc: 1.0000

1.0
```

Links

Horses or Humans Convnet

Horses or Humans with Validation

Horses or Humans with Compacting of Images