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.

In [2]:

```
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()
```

In [4]:

```
# 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')>DESIRED_ACCURACY):
                print("\nReached 99.9% accuracy so cancelling training!")
                self.model.stop_training = True

    callbacks = myCallback()

    # This Code Block should Define and Compile the Model. Please assume the ima
ges are 150 X 150 in your implementation.
    model = tf.keras.models.Sequential([
        # Your Code Here
    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(32, (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 = ['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

    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(
           "/tmp/h-or-s",
            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,
        verbose=1,
        callbacks=[callbacks])

    # model fitting
    return history.history['acc'][-1]
```

In [5]:

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

```
WARNING: Logging before flag parsing goes to stderr.
W0422 07:14:13.570796 140280201516864 deprecation.py:506] From /usr/
local/lib/python3.6/dist-packages/tensorflow/python/ops/init_ops.py:
1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.i
nit_ops) with dtype is deprecated and will be removed in a future ve
rsion.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing
it to the constructor
W0422 07:14:13.967475 140280201516864 deprecation.py:323] From /usr/
local/lib/python3.6/dist-packages/tensorflow/python/ops/nn_impl.py:1
80: add_dispatch_support.<locals>.wrapper (from tensorflow.python.op
s.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where

Found 80 images belonging to 2 classes.
Epoch 1/15
8/8 [==============================] - 6s 699ms/step - loss: 1.0374
- acc: 0.6000
Epoch 2/15
8/8 [==============================] - 0s 62ms/step - loss: 0.6159 -
acc: 0.6750
Epoch 3/15
8/8 [==============================] - 0s 62ms/step - loss: 0.2063 -
acc: 0.9375
Epoch 4/15
8/8 [==============================] - 1s 63ms/step - loss: 0.2023 -
acc: 0.9125
Epoch 5/15
8/8 [==============================] - 0s 62ms/step - loss: 0.3431 -
acc: 0.8875
Epoch 6/15
8/8 [==============================] - 1s 63ms/step - loss: 0.0859 -
acc: 0.9750
Epoch 7/15
8/8 [==============================] - 0s 62ms/step - loss: 0.0636 -
acc: 0.9750
Epoch 8/15
8/8 [==============================] - 0s 53ms/step - loss: 0.2807 -
acc: 0.9125
Epoch 9/15
8/8 [==============================] - 0s 54ms/step - loss: 0.0661 -
acc: 0.9750
Epoch 10/15
8/8 [==============================] - 0s 53ms/step - loss: 0.0291 -
acc: 0.9875
Epoch 11/15
8/8 [==============================] - 0s 53ms/step - loss: 0.0617 -
acc: 0.9875
Epoch 12/15
8/8 [==============================] - 0s 53ms/step - loss: 0.0370 -
acc: 0.9875
Epoch 13/15
7/8 [=========================>....] - ETA: 0s - loss: 0.0105 - acc:
1.0000
Reached 99.9% accuracy so cancelling training!
8/8 [==============================] - 0s 53ms/step - loss: 0.0096 -
acc: 1.0000
```

Out[5]:

```
1.0
```

In [4]:

```
# Now click the 'Submit Assignment' button above.
# Once that is complete, please run the following two cells to save your work an
d close the notebook
```

In [ ]:

```

```

In [ ]:

```

```