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 [1]:

```
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') > 0.999):
                print("\nReached 100% accuracy so cancelling training!")
                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(learning_rate=0.001),
                  metrics=['accuracy'])


    # This code block should create an instance of an ImageDataGenerator called train_d
atagen
    # 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=128,
                                                        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()
```

```
Found 80 images belonging to 2 classes.
Epoch 1/15
8/8 [==============================] - 4s 476ms/step - loss: 1.7617 - acc:
0.6484
Epoch 2/15
8/8 [==============================] - 2s 271ms/step - loss: 0.4729 - acc:
0.8422
Epoch 3/15
8/8 [==============================] - 2s 264ms/step - loss: 0.2821 - acc:
0.8797
Epoch 4/15
8/8 [==============================] - 2s 262ms/step - loss: 0.1281 - acc:
0.9688
Epoch 5/15
8/8 [==============================] - 2s 259ms/step - loss: 0.0996 - acc:
0.9719
Epoch 6/15
8/8 [==============================] - 2s 252ms/step - loss: 0.0714 - acc:
0.9750
Epoch 7/15
8/8 [==============================] - 2s 260ms/step - loss: 0.0366 - acc:
0.9891
Epoch 8/15
7/8 [=========================>....] - ETA: 0s - loss: 0.0133 - acc: 1.000
0
Reached 100% accuracy so cancelling training!
8/8 [==============================] - 2s 262ms/step - loss: 0.0127 - 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 and close
the notebook
```

In [6]:

```
%%javascript
<!-- Save the notebook -->
IPython.notebook.save_checkpoint();
```

In [ ]:

```
%%javascript
IPython.notebook.session.delete();
window.onbeforeunload = null
setTimeout(function() { window.close(); }, 1000);
```