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 [20]:
```
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 loca
tion
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 [25]:
```
# 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 99% accuracy so cancelling trainin
g!")
                self.model.stop_training = True

    callbacks = myCallback()

    # This Code Block should Define and Compile the Model. Please a
ssume the images are 150 X 150 in your implementation.
    model = tf.keras.models.Sequential([
        # Note the input shape is the desired size of the image 150
X 150 with 3 bytes color
    # This is the first convolution
        tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_
shape=(150, 150, 3)),
        tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
        tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
```

```
        tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
        tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
        tf.keras.layers.Dense(512, activation='relu'),
    # Only 1 output neuron. It will contain a value from 0-1 where
0 for 1 class ('horses') and 1 for the other ('humans')
        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 ImageDataGene
rator called train_datagen
    # And a train_generator by calling train_datagen.flow_from_dire
ctory

    from tensorflow.keras.preprocessing.image import ImageDataGener
ator

    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/',  # This is the source directory for trainin
g images
        target_size=(150, 150),
        batch_size=128,
        # Since we use binary_crossentropy loss, we need binary lab
els
        class_mode='binary')

        # Your Code Here)
    # 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(
      train_generator,
      steps_per_epoch=8,
      epochs=15,
      verbose=1,callbacks=[callbacks])
    # model fitting
    return history.history['acc'][-1]
```

In [26]:
```
# The Expected output: "Reached 99.9% accuracy so cancelling traini
ng!""
train_happy_sad_model()
```

```
Found 80 images belonging to 2 classes.
Epoch 1/15
8/8 [==============================] - 3s 432ms/step - loss: 0.773
0 - acc: 0.5719
Epoch 2/15
8/8 [==============================] - 2s 265ms/step - loss: 0.605
6 - acc: 0.7266
Epoch 3/15
8/8 [==============================] - 2s 249ms/step - loss: 0.364
0 - acc: 0.8641
Epoch 4/15
8/8 [==============================] - 2s 260ms/step - loss: 0.204
1 - acc: 0.9250
Epoch 5/15
8/8 [==============================] - 2s 252ms/step - loss: 0.146
8 - acc: 0.9484
Epoch 6/15
8/8 [==============================] - 2s 248ms/step - loss: 0.102
6 - acc: 0.9672
Epoch 7/15
8/8 [==============================] - 2s 251ms/step - loss: 0.038
6 - acc: 0.9922
Epoch 8/15
7/8 [=========================>....] - ETA: 0s - loss: 0.0074 - ac
c: 1.0000
Reached 99% accuracy so cancelling training!
8/8 [==============================] - 2s 261ms/step - loss: 0.006
8 - acc: 1.0000
```

Out[26]:
```
1.0
```

In [ ]:
```
# 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 [ ]:
```
%%javascript
<!-- Save the notebook -->
IPython.notebook.save_checkpoint();
```

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