Open in Colab

```
# this is used in downloading data from the google drive
#!pip install gdown
```

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

!gdown --id 1NvV6VhmrfU8JDZNoEbwJxwx_6dhyN5bf

zip_ref = zipfile.ZipFile("./happy-or-sad.zip", 'r')
zip_ref.extractall("./h-or-s")
zip_ref.close()
```

```
Downloading...
From: https://drive.google.com/uc?id=1NvV6VhmrfU8JDZNoEbwJxwx_6dhyN5bf
To: /content/happy-or-sad.zip
100% 2.67M/2.67M [00:00<00:00, 85.2MB/s]
```

```
# GRADED FUNCTION: train_happy_sad_model
def train_happy_sad_model():

    DESIRED_ACCURACY = 0.999

    class myCallback(tf.keras.callbacks.Callback):

        def on_epoch_end(self, epoch, logs={}):
            if logs.get('accuracy') is not None and logs.get('accuracy') > 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 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(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(learning_rate=0.001),
              metrics=['accuracy']) # YOUR CODE HERE)

# 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( "./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(train_generator,
                    steps_per_epoch=8,
                    epochs=15,
                    verbose=1,
                    callbacks=[callbacks]
```

```
                       )

    return history.history['accuracy'][-1]



train_happy_sad_model()
```

```
Found 80 images belonging to 2 classes.
Epoch 1/15
8/8 [==============================] - 31s 26ms/step - loss: 1.2697 - accuracy: 0.6500
Epoch 2/15
8/8 [==============================] - 0s 26ms/step - loss: 0.6432 - accuracy: 0.5875
Epoch 3/15
8/8 [==============================] - 0s 26ms/step - loss: 0.2854 - accuracy: 0.9125
Epoch 4/15
8/8 [==============================] - 0s 28ms/step - loss: 0.3268 - accuracy: 0.8875
Epoch 5/15
8/8 [==============================] - 0s 26ms/step - loss: 0.1608 - accuracy: 0.9500
Epoch 6/15
8/8 [==============================] - 0s 26ms/step - loss: 0.0979 - accuracy: 0.9625
Epoch 7/15
8/8 [==============================] - 0s 26ms/step - loss: 0.1513 - accuracy: 0.9250
Epoch 8/15
8/8 [==============================] - 0s 26ms/step - loss: 0.0291 - accuracy: 1.0000

Reached 99.9% accuracy so cancelling training!
1.0
```

✓ 38s completed at 3:36 PM ● ✕