DeepLearning.AI TensorFlow Developer

Contents

1 Introduction to TensorFlow for AI, ML, and DL

1.0.1 Callbacks

We can use **callbacks** in order to stop training when we reach a certain accuracy we desire. This is to stop the loss from beginning to increase again if we start to overfit the model. Click here to see the TensorFlow Callbacks documentation.

```
import tensorflow as tf
print(tf.__version__)

class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('accuracy')>0.6): # might need to use 'acc' instead
      print("\nReached 60% accuracy so cancelling training!")
      self.model.stop_training = True

callbacks = myCallback()

mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_images, y_labels, epochs=10, callbacks=[callbacks])
```

1.0.2 Upload Custom Images

We can use the below code to upload a custom image and use it on a trained model.

```
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")
```

1.0.3 ImageDataGenerator

```
import tensorflow as tf
import os
import zipfile
from os import path, getcwd, chdir
from tensorflow.keras.optimizers import RMSprop
  from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Import and extract zip file containing images
path = f"{getcwd()}/../tmp2/happy-or-sad.zip"
zip_ref = zipfile.ZipFile(path, 'r')
zip_ref.extractall("/tmp/h-or-s")
zip_ref.close()

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('acc')>DESIRED_ACCURACY):
        print('\nReached 100% accuracy so stopping training.')
        self.model.stop_training = True

  callbacks = myCallback()

  # Define and Compile the Model.
  model = tf.keras.models.Sequential([
      tf.keras.layers.Conv2D(64, (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(16, (3,3), activation='relu'),
      tf.keras.layers.MaxPooling2D(2,2),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(512),
      tf.keras.layers.Dense(1, activation='sigmoid')
  ])

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

  # Create an instance of an ImageDataGenerator
  train_datagen = ImageDataGenerator(rescale=1./255)

  train_generator = train_datagen.flow_from_directory(
      '/tmp/h-or-s',
      target_size=(150,150),
      class_mode='binary'
  )

  history = model.fit(
      train_generator,
      epochs=50,
      callbacks=[callbacks],
      verbose=1
  )

  return history.history['acc'][-1]
```

2 CNN's in TensorFlow