Photo by Samuel Zeller on Unsplash

How to stop training a neural-network using callback?

An useful hack with Tensorflow and Keras

Supratim Haldar Mar 18, 2019 · 2 min read

Introduction

Often, when training a very deep neural network, we want to stop training once the training accuracy reaches a certain desired threshold. Thus, we can achieve what we want (optimal model weights) and avoid wastage of resources (time and computation power).

In this brief tutorial, let's learn how to achieve this in Tensorflow and Keras, using the **callback** approach, **in 4 simple steps**.

Deep-dive

```
# Import tensorflow
import tensorflow as tf
```

1. First, set the accuracy threshold till which you want to train your model.

```
ACCURACY_THRESHOLD = 0.95
```

2. Now, implement callback class and function to stop training when accuracy reaches ACCURACY_THRESHOLD.

```
# Implement callback function to stop training
# when accuracy reaches e.g. ACCURACY_THRESHOLD = 0.95

class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('acc') > ACCURACY_THRESHOLD):
        print("\nReached %2.2f%% accuracy, so stopping training!!" %
(ACCURACY_THRESHOLD*100))
        self.model.stop_training = True
```

What exactly is going on here? We are creating new class by extending *tf.keras.callbacks.Callback*, and implementing the *on_epoch_end()* method. This is invoked at the end of each epoch. Next, we are fetching the value of accuracy at the end of that epoch, and if it is greater than our threshold, we are setting the stop_training of model to True.

3. Instantiate an object of *myCallback* class.

```
callbacks = myCallback()
```

Next, build a DNN or Conv-Net model following the normal steps of TensorFlow or Keras. The callback that we have built above will be used while training the model using *fit()* method.

4. Simply pass an argument as **callbacks=[<the newly instantiated object of myCallback class>]** to fit() method.

```
model.fit(x_train, y_train, epochs=20, callbacks=[callbacks])
```

And that's all! While training, as soon as accuracy reaches the value set in ACCURACY_THRESHOLD, training will be stopped.

To tie it all together, here's a complete code snippet.

```
import tensorflow as tf

# Implement callback function to stop training
# when accuracy reaches ACCURACY_THRESHOLD
ACCURACY_THRESHOLD = 0.95

class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
                if(logs.get('acc') > ACCURACY_THRESHOLD):
                        print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_TH
                        self.model.stop_training = True

# Instantiate a callback object
callbacks = myCallback()

# Load fashion mninst dataset
mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
# Scale data
```

```
x_train, x_test = x_train / 255.0, x_test / 255.0

# Build a conv dnn model
model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        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_train, y_train, epochs=20, callbacks=[callbacks])
```

tf_callback_to_stop_training.py hosted with ♡ by GitHub view raw

Conclusion

With our imagination this approach can be used in varied creative ways, especially when we want to run quick PoCs to test and validate multiple DNN architectures. What other interesting usages can you think of? Please share your thoughts in the comments section below.

Machine Learning Deep Learning Neural Networks TensorFlow Keras