Tensorflow and Keras 2

Callbacks

A callback **defines a set of functions** that are **executed at different stages of the training** procedure.

They can be **used to view the internal states of the model** during training.

- For example, we may **want to print loss, accuracy or lr every 2000th epoch**.

Examples:

1. **on_epoch_begin** : function will execute before every epoch
2. **on_epoch_end** : function will execute after every epoch
3. **verbose**=1, model training prints associated data after every epoch
4. **verbose**=0, model training prints nothing

Customized callback example

```
class VerboseCallback(tf.keras.callbacks.Callback):
  # runs only before the training starts
  def on_train_begin(self, logs=None):
        print("Starting training...")

  # runs after every epoch
  def on_epoch_end(self, epoch, logs = None):
    if epoch % 50 == 0:
       print(f'Epoch {str(epoch).zfill(3)}', '- loss : ', logs['loss'], '- Acc : ', logs['accuracy'])

  # runs once training is finished
  def on_train_end(self, logs=None):
        print("...Finished training")
```

- The custom class will inherit from tf.keras.callbacks.Callback.
- All methods in keras.callbacks.callback class will be available for our customized class, and we can also override them.

We will have to **pass a list of callback objects** to callback argument of the fit method

```
history = model.fit(X_train, y_train, epochs=500, batch_size=256,
                    validation_split=0.1, verbose=0, callbacks=[VerboseCallback()])
```

```
Starting training...
Epoch 000 - loss :  0.5516670346260071 - Acc :  0.7737144231796265
Epoch 050 - loss :  0.5456981658935547 - Acc :  0.7760704755783081
Epoch 100 - loss :  0.5408483743667603 - Acc :  0.776172935962677
Epoch 150 - loss :  0.5364638566970825 - Acc :  0.7808850407600403
Epoch 200 - loss :  0.5334064364433289 - Acc :  0.7805777788162231
Epoch 250 - loss :  0.530765175819397 - Acc :  0.7842655181884766
Epoch 300 - loss :  0.5280439853668213 - Acc :  0.7856996655464172
Epoch 350 - loss :  0.524202823638916 - Acc :  0.7886703610420227
Epoch 400 - loss :  0.5229038000106812 - Acc :  0.7881581783294678
Epoch 450 - loss :  0.5198482275009155 - Acc :  0.7865191698074341
...Finished training
```

Note: We **can pass callback objects to evaluate and predict** method as well.

The parent class tf.keras.callbacks.Callback supports various kinds of methods that we can override

- Global methods
  at the beginning/ending of training
- Batch-level method
  at the beginning/ending of a batch
- Epoch-level method
  at the beginning/ending of an epoch

Other examples include:

1. CSVLogger - save history object in a csv file csv_logger = keras.callbacks.CSVLogger("file_name.csv")
2. EarlyStopping - stop the training when model starts to overfit
3. ModelCheckpoint - saves the intermediate model weights
4. LearningRateScheduler - control/change Learning Rate in between epoch

Tensorboard

- It is used to **closely monitor the training process**
- It can be used to visualize information regarding the training process like
  - ❖ Metrics - loss, accuracy
  - ❖ Visualize the model graphs
  - ❖ Histograms of W, b, or other tensors as they change during training - distributions
  - ❖ Displaying images, text, and audio data

Ways to install

pip install tensorboard

conda install -c conda-forge tensorboard

To load tensorboard in the notebook

%load_ext tensorboard

TensorBoard will **store all the logs in this log directory**.

log_folder = 'logs'

- It will read from these logs in order to display the various visualizations.

If we want to reload the TensorBoard extension, we can use the reload magic method

%reload_ext tensorboard

Import

```
model = create_model()
from tensorflow.keras.callbacks import TensorBoard
tb_callback = TensorBoard(log_dir=log_folder, histogram_freq=1)
history = model.fit(X_train, y_train, epochs=500, batch_size=512,
                    validation_data = (X_val, y_val), verbose=0, callbacks=[tb_callback])
```

Callback arguments:

- **log_dir** - (Path)
  - directory where logs will be saved
  - This directory should not be used by any other callbacks.
- **update_freq** - (int/str)
  - how frequently losses/metrics are updated.
  - when set to batch, losses/ metrics will be updated after every batch/iteration
  - when set to an integer N, updation will be done after N batches
  - when set to 'epoch', updation will be done after every epoch
- **histogram_freq** - (int)
  - how frequently (in epochs) histograms(Distribution of W) will be updated.

    - Setting this to 0 means, histograms will not be computed.
- **write_graph** - (Bool), True if we want to visualize our training process
- **write_images** - (Bool), True if we want to visualize our model weights

Launch tensorboard using following command

%tensorboard --logdir={log_folder}