← Go Back to Deep Learning

Course Content

New Package Introduction: Introduction to Deep Learning

Let's go through some of the common functions used in this LVC case studies.

TensorFlow

It is a free and open-source library in Python that is used for **Deep Learning**. It has a multitude of applications that it is used in but majorly it is used for training and implementing Deep Neural Networks.

TensorFlow being a large library can sometimes cause problem while installing it on local machine. It is a good practice to run TensorFlow using Google Colab.

To import this library, the below line of code can be used in Google Colab notebook:

```
import tensorflow as tf
```

To gain more information about TensorFlow one can refer to this link.

to_categorical

Using the method to_categorical(), a numpy array (or) a vector which has integers that represent different categories, can be converted into a numpy array (or) a matrix which has binary values and has columns equal to the number of categories in the data. For example:

```
y_train = tf.keras.utils.to_categorical(trainY, num_classes = 10)
```

Here, trainY is a vector that has 10 unique integers representing 10 classes. So, the output, y_train, will have as many rows as trainY and 10 binary columns, where 1 represents the row belong to that category and rest of the columns are zero.

To get more clarity on this you can refer to this link.

Sequential

This method is used to apply sequential model over a certain data. Sequential model is more suitable for a plain stack of layers, where each layer has exactly one input tensor and one output tensor. The below code shows an example:

```
# Initializing a sequential model
model = tf.keras.Sequential([

 tf.keras.layers.Flatten(input_shape = (28, 28)),

 tf.keras.layers.Dense(64, activation = 'relu'),

 tf.keras.layers.Dense(10, activation = 'softmax')
])
```

Here, the model is trained with one **flatten layer** and one **dense layer** with **relu** as the activation function and one output layer with **softmax** as the activation function.

To get more clarity on this, you can refer this link.

Compile

In TensorFlow, once a model is trained, it needs to be compiled. The below code demonstrates the same:

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

Here, the model is compiled with the help of **adam optimizer**, **categorical_crossentropy** as the loss function and **accuracy** as the performance metrics. Compile defines the loss function, the optimizer and the metrics.

To get more clarity, you can refer to this link.

Fit

A compiled model needs to be fit over the **training data**. This is done as follows:

```
# Let us now fit the model
fit_history = model.fit(X_train, y_train, validation_split = 0.1, verbose = 1, epochs = 10, batch_size = 64)
```

Here, the model is fit over the mentioned training data with specific parameters.

To get more clarity on this, you can refer to this link.

Evaluate

As the final step, a trained model needs to be evaluated over the **test data**. The below code is used for the same:

```
model.evaluate(X_test, y_test, verbose = 1)
```

To get more clarity on this, you can refer to this link.

Happy learning!

Previous Next