

**Name: Shubham  
Warik  
Roll no.: 66  
DL Experiment 1**

## **TensorFlow And Keras**

**TensorFlow** is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

### **TensorFlow Methods:**

- **tf.constant()**
- This method allows to create a constant tensor with a specified value

```
import tensorflow as tf
# Create a constant tensor
tensor = tf.constant([1, 2, 3, 4, 5])
```

- **tf.Variable()**
- The tf.Variable() method is used to create a mutable tensor that can be modified during model training. Variables are often used to store the model's trainable parameters.

```
import tensorflow as tf
# Create a trainable variable
weights = tf.Variable(tf.random.normal(shape=(10, 10)))
```

- **tf.data.Dataset**
- The tf.data.Dataset API allows you to create efficient input pipelines for your TensorFlow models. You can use it to load and preprocess data easily.

```
import tensorflow as tf
```

```
# Create a dataset from a list
data = [1, 2, 3, 4,5]
dataset = tf.data.Dataset.from_tensor_slices(data)
```

- **tf.keras.layers**
- TensorFlow provides a high-level Keras API for building and training neural networks. The tf.keras.layers module offers a wide range of pre-built layers that you can use to construct your model.

```
import tensorflow as tf
# Create a simple neural network using Keras layers
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(input_size)),
    tf.keras.layers.Dense(10, activation='softmax')
```

- **tf.keras.losses**
- The tf.keras.losses module contains various loss functions that can be used during model training to compute the difference between predicted and actual values.

```
import tensorflow as tf
# Categorical cross-entropy loss
loss = tf.keras.losses.CategoricalCrossentropy()
```

- **tf.keras.optimizers**
- The tf.keras.optimizers module provides different optimization algorithms that can be used to update the model's parameters during training.

```
import tensorflow as tf
# Stochastic Gradient Descent optimizer
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)
```

- **tf.keras.metrics**
- The tf.keras.metrics module offers various evaluation metrics that can be used to monitor model performance during training and testing.

```
import tensorflow as
tf# Accuracy metric
```

```
accuracy = tf.keras.metrics.Accuracy()
```

- **tf.keras.callbacks**
- The tf.keras.callbacks module provides callback functions that allow you to perform actions at different stages of the training process.  
import tensorflow as tf  
# Accuracy metric  
accuracy = tf.keras.metrics.Accuracy()

- **tf.keras.callbacks**
- The tf.keras.callbacks module provides callback functions that allow you to perform actions at different stages of the training process.

```
import tensorflow as tf  
# Model checkpoint callback  
checkpoint_callback =  
tf.keras.callbacks.ModelCheckpoint(filepath='model_checkpoint.h5',  
save_best_only=True)
```

**Keras** is a compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with the focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. The creation of framework can be of the following two types –

- Sequential API
- Functional API

## **Keras Methods:**

- **keras.models.Sequential**
- The Sequential class in Keras allows you to create a linear stack of layers for building a neural network  
import keras  
from keras.models import  
Sequential# Create a sequential  
model  
model = Sequential()

- **keras.layers**
- Similar to TensorFlow's tf.keras.layers, Keras also provides a wide range of pre-built layers that you can use to construct your model.

```
import keras
from keras.models import Sequential

from keras.layers import Dense
# Create a simple neural network using Keras layers
model = Sequential([Dense(64, activation='relu',
input_shape=(input_size,)),Dense(10,
activation='softmax')])
```

- **keras.losses**
- Keras also has a losses module that contains various loss functions for model training.

```
import keras
from keras.losses import
categorical_crossentropy# Categorical cross-
entropy loss
loss = categorical_crossentropy(y_true, y_pred)
```

- **keras.optimizers**
- The optimizers module in Keras provides different optimization algorithms for updating the model's parameters during training.

```
import keras
from keras.optimizers import SGD
# Stochastic Gradient Descent optimizer
optimizer = SGD(learning_rate=0.01)
```

- **keras.metrics**
- Keras offers a metrics module that contains various evaluation metrics to monitor model performance during training and testing.

```
import keras
from keras.metrics import
accuracy# Accuracy metric
accuracy_metric = accuracy(y_true, y_pred)
```

- **keras.callbacks**

- Similar to TensorFlow, Keras also provides a callbacks module with callback functions to perform actions at different stages of the training process.

```
import keras
```

```
from keras.callbacks import
```

```
ModelCheckpoint# Model checkpoint
```

```
callback
```

```
checkpoint_callback = ModelCheckpoint(filepath='model_checkpoint.h5',
```

```
save_best_only=True)
```