Name - Dhanashree Thakur
Roll No - 56
Branch - CSE(DS)
Experiment N0- 2

Tensorflow

What is Tensorflow ?

TensorFlow is an open-source machine learning library developed by Google. TensorFlow is used to build and train deep learning models as it facilitates the creation of computational graphs and efficient execution on various hardware platforms. It provides a comprehensive ecosystem of tools, libraries, and community resources that lets researchers and developers build and deploy machine learning models easily. TensorFlow is basically a software library for numerical computation using data flow graphs where nodes in the graph represent mathematical operations and edges in the graph represent the multidimensional data arrays (called tensors) communicated between them.

Key features of TensorFlow

1. **Flexibility**: It supports multiple levels of abstraction, allowing developers to choose the right level of abstraction for their needs.
2. **Scalability**: TensorFlow can run on multiple CPUs and GPUs, and it supports distributed computing.
3. **Comprehensive ecosystem**: It includes tools for model development (TensorFlow Core), higher-level APIs for easy model building (like Keras), and tools for deployment (TensorFlow Serving).
4. **Community and support**: TensorFlow has a large and active community, which contributes to its development, shares knowledge, and creates additional tools and libraries.

Operations in Tensorflow:

Tensor Addition

You can add two tensors using tensorA.add(tensorB):

const tensorA = tf.tensor([[1, 2], [3, 4], [5, 6]]);

```
 const tensorB = tf.tensor([[1,-1], [2,-2], [3,-3]]);
 // Tensor Addition
 const tensorNew = tensorA.add(tensorB);
 // Result:
[ [2, 1], [5, 2], [8, 3] ]
```

Tensor Subtraction

You can subtract two tensors using tensorA.sub(tensorB):

```
const tensorA = tf.tensor([[1, 2], [3, 4], [5, 6]]);
const tensorB = tf.tensor([[1,-1], [2,-2], [3,-3]]);
 // Tensor Subtraction
 const tensorNew = tensorA.sub(tensorB);
// Result:
 [ [0, 3], [1, 6], [2, 9] ]
```

Tensor Multiplication

You can multiply two tensors using tensorA.mul(tensorB):

```
const tensorA = tf.tensor([1, 2, 3, 4]);
 const tensorB = tf.tensor([4, 4, 2, 2]);
// Tensor Multiplication
const tensorNew = tensorA.mul(tensorB);
 // Result:
[ 4, 8, 6, 8 ]
```

Tensor Division

You can divide two tensors using tensorA.div(tensorB):

```
 const tensorA = tf.tensor([2, 4, 6, 8]);
 const tensorB = tf.tensor([1, 2, 2, 2]);
// Tensor Division
 const tensorNew = tensorA.div(tensorB);
// Result:
 [ 2, 2, 3, 4 ]
```

Keras

What is Keras ?

Keras is a high-level neural networks API, originally developed as an independent open-source project, and later integrated into TensorFlow as its official high-level API. It provides an easy-to-use interface for building, training, evaluating, and deploying deep learning models.

Key Features of Keras

1. **User Friendliness**: Keras is designed to be user-friendly, modular, and easy to extend. It allows for fast prototyping and supports both convolutional networks (CNNs) and recurrent networks (RNNs), as well as combinations of the two.
2. **Modularity**: Models in Keras are defined as a sequence of layers, each performing a specific transformation on the input to generate the output. This modular approach enables easy building and modification of complex neural network architectures.
3. **Extensibility**: Keras supports the addition of custom layers, loss functions, and metrics. It allows developers to define and use their own custom components seamlessly within the framework.
4. **Compatibility**: Keras is compatible with both CPU and GPU computations. It seamlessly integrates with other popular Python libraries like NumPy, Scikit-Learn, and TensorFlow.
5. **Integration**: As of TensorFlow 2.0, Keras has been integrated deeply into TensorFlow as tf.keras. This integration allows users to access all the capabilities of TensorFlow while leveraging the simplicity and ease of use of Keras.

Methods in keras

Import Keras
```
from keras.models import Sequential
from keras.layers import Dense
```

Create a Sequential model
```
model = Sequential()
```
Add layers
```
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

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

Train the model
```
model.fit(X_train, y_train, batch_size=32, epochs=10)
```

Evaluate the model
```
loss, accuracy = model.evaluate(X_test, y_test)
print('Test loss:', loss)
print('Test accuracy:', accuracy)
```

Make predictions
```
predictions = model.predict(X_new)
```