- NIELIT CHANDIGARH

Basics of TensorFlow & Keras

- An Introduction to Deep Learning Frameworks

@NIELITCHANDIGARH

What is TensorFlow?

- Open-source machine learning framework by Google
- Optimized for deep learning with CPU/GPU/TPU support
- Used for neural networks, computer vision, NLP, and more
- Highly scalable for cloud and edge computing

What is Keras?

- High-level deep learning API built on TensorFlow
- Simplifies neural network design and training
- Supports multiple backends (TensorFlow, Theano, etc.)
- Ideal for fast prototyping and experimentation

Installing TensorFlow & Keras

- To install TensorFlow (which includes Keras), use:

```
pip install tensorflow
```

- To check the installation:

```
import tensorflow as tf
print(tf.__version__)
```

Working with Tensors

- Tensors are multi-dimensional arrays used in TensorFlow.

Example:

```
import tensorflow as tf
a = tf.constant(5)
b = tf.constant([1, 2, 3])
c = tf.constant([[1, 2], [3, 4]])
print(a, b, c)
```

Building a Simple Neural Network

- Creating a Feedforward Neural Network using Keras:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(784,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

Compiling and Training the Model

- Compile and train the model with MNIST dataset:
- NOTE: The **MNIST (Modified National Institute of Standards and Technology) dataset** is one of the most famous datasets in machine learning and deep learning. It is used for training and testing image classification models, especially for handwritten digit recognition.

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


# Load dataset

mnist = keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()


# Normalize data

x_train, x_test = x_train / 255.0, x_test / 255.0


# Train the model

model.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
```

Evaluating and Using the Model

- To evaluate model accuracy:

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test Accuracy:', test_acc)
```

Making predictions:

```
predictions = model.predict(x_test)
print('Predicted class:', predictions[0].argmax())
```

TensorFlow vs. Keras: Comparison

Feature	TensorFlow	Keras
Complexity	High	Low
Performance	High	Moderate
Flexibility	Very Flexible	Easy to use
Best for	Large-Scale ML/DL	Quick Prototyping

Next Steps

- Learn Convolutional Neural Networks (CNNs) for image processing
- Explore Recurrent Neural Networks (RNNs) & LSTMs for sequential data
- Try Transfer Learning with pre-trained models (VGG16, ResNet, MobileNet)
- Deploy models using TensorFlow Lite and TensorFlow.js