Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai
Artificial Intelligence and Data Science Department

Deep Learning/Odd Sem 2023-23/Experiment 1b

Name : Shreya Singh	Class/Roll No. :D16AD/55	Grade :

Title of Experiment :

To explore python libraries for deep learning e.g. Theano,TensorFlow etc.

Objective of Experiment :

Explore and understand the Python libraries TensorFlow and Keras for building and training deep learning models.

Outcome of Experiment :

We successfully implemented the Practical using TensorFlow and Keras python Libraries.

Problem Statement :

Develop a basic image classification model using TensorFlow and Keras to categorize grayscale images (28x28 pixels) into one of ten possible classes. Evaluate the model's accuracy and loss on a test dataset, and obtain predicted probabilities for sample test images.

Description / Theory :

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google. It is widely used in the field of deep learning for building and training neural networks. TensorFlow provides a flexible and efficient way to work with large-scale data, enabling researchers and developers to create sophisticated deep learning models for tasks like image recognition, natural language processing, and more. Its usefulness lies in its ability to automatically handle complex mathematical computations on tensors (multidimensional arrays) and efficiently optimize models using specialized hardware, such as GPUs, making deep learning tasks faster and more accessible.

Keras
Keras is an open-source high-level neural networks API written in Python. It is built on top of TensorFlow, Theano, and CNTK, and provides a user-friendly interface for creating and training deep learning models. Keras simplifies the process of building complex neural networks by offering a streamlined and easy-to-use syntax. Its usefulness lies in enabling researchers and practitioners to rapidly prototype and experiment with deep learning architectures, making it a popular choice for beginners and experts alike in the field of deep learning.

Program and Output:

```
[1] import tensorflow as tf
    print("TensorFlow version:", tf.__version__)
```

```
TensorFlow version: 2.13.0
```

```
[2] mnist = tf.keras.datasets.mnist

    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
```

```
[3] model = tf.keras.models.Sequential([
      tf.keras.layers.Flatten(input_shape=(28, 28)),
      tf.keras.layers.Dense(128, activation='relu'),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Dense(10)
    ])
```

```
predictions = model(x_train[:1]).numpy()
predictions
```

```
array([[ 0.05628643,  0.14975291, -0.0973175 , -0.13103065, -0.59004533,
        -0.21109745, -0.27316445,  0.9292571 ,  0.24429236, -0.19345057]],
      dtype=float32)
```

```
[5] tf.nn.softmax(predictions).numpy()
```

```
array([[0.09829664, 0.10792714, 0.08430034, 0.08150569, 0.05150393,
        0.0752342 , 0.0707066 , 0.23532334, 0.11862838, 0.07657364]],
      dtype=float32)
```

```
[6] loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
loss_fn(y_train[:1], predictions).numpy()
```

```
2.5871491
```

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

```
[9] model.fit(x_train, y_train, epochs=5)

Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3019 - accuracy: 0.9130
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1456 - accuracy: 0.9572
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1085 - accuracy: 0.9674
Epoch 4/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.0877 - accuracy: 0.9727
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0747 - accuracy: 0.9768
<keras.src.callbacks.History at 0x7c29b765c610>
```

```
[10] model.evaluate(x_test,  y_test, verbose=2)

313/313 - 2s - loss: 0.0705 - accuracy: 0.9776 - 2s/epoch - 5ms/step
[0.07054876536130905, 0.9775999784469604]
```

```
[11] probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
```

```
probability_model(x_test[:5])

<tf.Tensor: shape=(5, 10), dtype=float32, numpy=
array([[1.1771471e-07, 2.6828664e-10, 1.2635549e-05, 3.2423026e-04,
        7.3285218e-11, 7.9796081e-07, 2.1356013e-13, 9.9962246e-01,
        2.7614149e-06, 3.6881840e-05],
       [1.8067753e-09, 6.6799088e-04, 9.9932814e-01, 1.4544297e-06,
        1.3549276e-12, 2.4558030e-06, 2.2674069e-08, 5.6476075e-13,
        5.1290236e-08, 1.5701376e-12],
       [2.1234973e-07, 9.9822491e-01, 2.1516280e-04, 1.1975584e-05,
        2.0389748e-04, 4.5092835e-05, 9.0035956e-06, 1.1642851e-03,
        1.2313765e-04, 2.3096436e-06],
       [9.9935502e-01, 2.9263372e-10, 4.4568546e-06, 1.0344659e-06,
        3.1732973e-06, 2.8066197e-06, 6.1741722e-04, 8.8418665e-06,
        8.9348612e-10, 7.2905582e-06],
       [9.3926619e-06, 2.8974008e-09, 1.6800981e-05, 1.4076870e-08,
        9.9898833e-01, 1.2605791e-05, 4.0222440e-05, 3.4580505e-04,
        2.5563668e-07, 5.8645918e-04]], dtype=float32)>
```

Results and Discussions :

The code successfully implements a basic image classification model using TensorFlow and Keras. The model achieved an accuracy of 97% on the test dataset. By using the SoftMax activation, we obtained predicted probabilities for sample test images. The model's simplicity makes it a good starting point, but further improvements can be explored by trying different architectures, hyperparameter tuning, and data augmentation techniques