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 1

Name: Heramb Ramakant Pawar	**Class/Roll No.:** D16AD/67	**Grade:**

Title of Experiment: Explore Python Libraries Tensor Flow and Keras

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 (multi-dimensional 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.

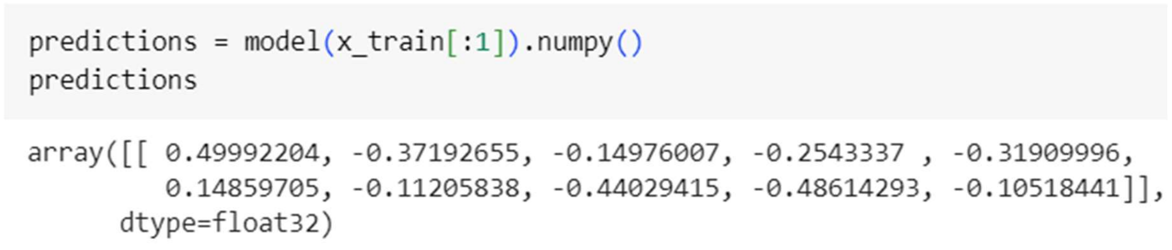
Program:

```
#importing TensorFlow Library
import tensorflow as tf

#Loading Dataset
mnist = tf.keras.datasets.mnist

#Spliting Dataset into Training and Testing Set
(x_train, y_train), (x_test, y_test) = mnist.load_data()

#Normailizing The Data
  x_train, x_test = x_train / 255.0, x_test / 255.0

#creating Sequential Model
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)
])

# prediction on First Training Sample
predictions = model(x_train[:1]).numpy()
predictions
```

```
# converting Raw Output values into probabilities using Softmax
tf.nn.softmax(predictions).numpy()
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

# Calculating Loss For First Training Sample
loss_fn(y_train[:1], predictions).numpy()

# Compiling Model
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test,  y_test, verbose=2)

# Creating Probablity Model
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
probability_model(x_test[:5])
```

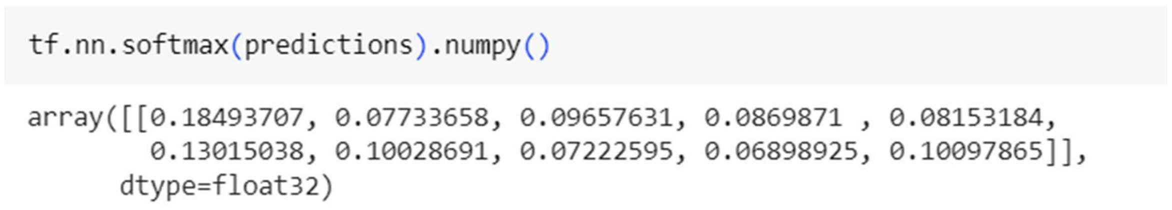
Output:

Making Predictions on the First Training Sample and Converting Raw Output Values Of the Model into Probabilities using SoftMax Function.



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

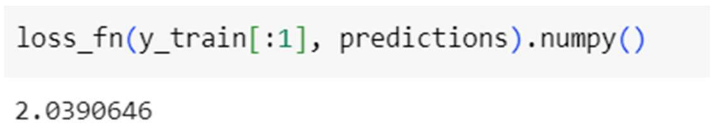
```
array([[ 0.49992204, -0.37192655, -0.14976007, -0.2543337 , -0.31909996,
         0.14859705, -0.11205838, -0.44029415, -0.48614293, -0.10518441]],
      dtype=float32)
```



```
tf.nn.softmax(predictions).numpy()
```

```
array([[0.18493707, 0.07733658, 0.09657631, 0.0869871 , 0.08153184,
        0.13015038, 0.10028691, 0.07222595, 0.06898925, 0.10097865]],
      dtype=float32)
```

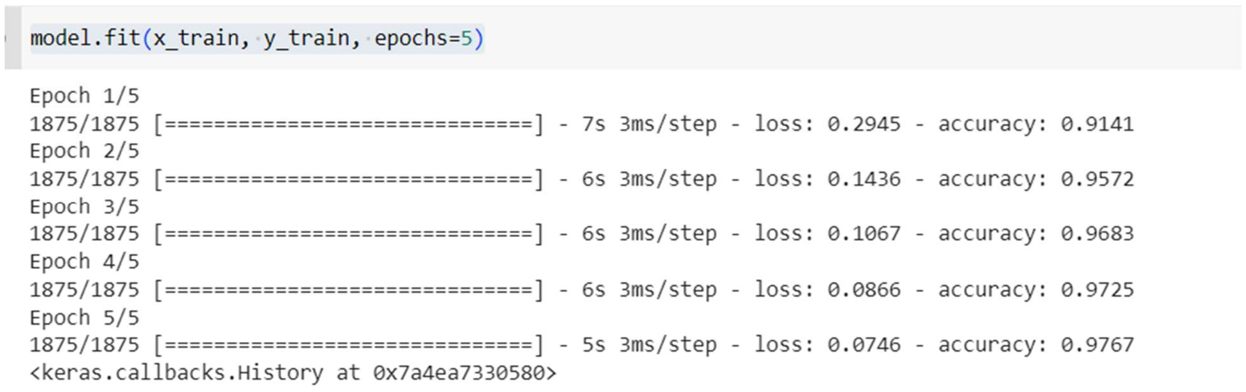
calculating the loss for the first training sample by comparing the true label



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

```
2.0390646
```

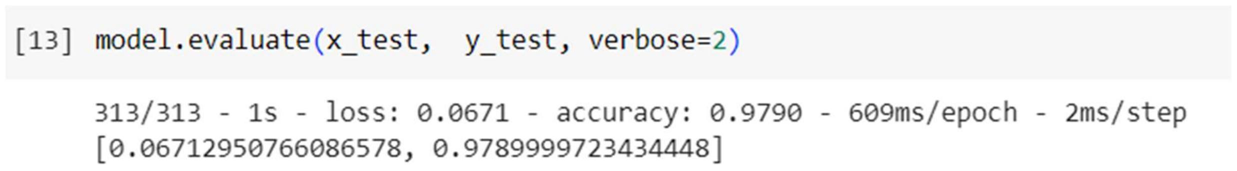
Training the Model Using the Data For 5 Epochs



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

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2945 - accuracy: 0.9141
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1436 - accuracy: 0.9572
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1067 - accuracy: 0.9683
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0866 - accuracy: 0.9725
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0746 - accuracy: 0.9767
<keras.callbacks.History at 0x7a4ea7330580>
```

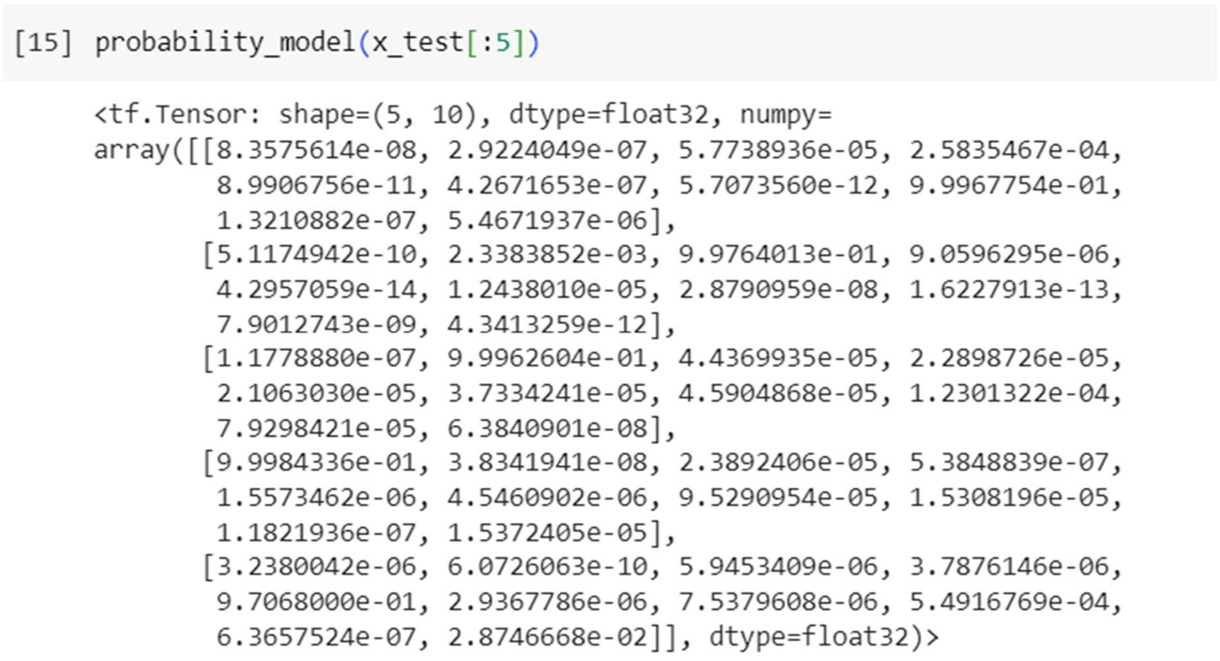
Evaluating the model's performance on the test dataset and prints the test loss and accuracy.



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

```
313/313 - 1s - loss: 0.0671 - accuracy: 0.9790 - 609ms/epoch - 2ms/step
[0.06712950766086578, 0.9789999723434448]
```

making predictions on the first 5 test samples using the probability_model probabilities for each class for each sample.



```
[15] probability_model(x_test[:5])

     <tf.Tensor: shape=(5, 10), dtype=float32, numpy=
     array([[8.3575614e-08, 2.9224049e-07, 5.7738936e-05, 2.5835467e-04,
             8.9906756e-11, 4.2671653e-07, 5.7073560e-12, 9.9967754e-01,
             1.3210882e-07, 5.4671937e-06],
            [5.1174942e-10, 2.3383852e-03, 9.9764013e-01, 9.0596295e-06,
             4.2957059e-14, 1.2438010e-05, 2.8790959e-08, 1.6227913e-13,
             7.9012743e-09, 4.3413259e-12],
            [1.1778880e-07, 9.9962604e-01, 4.4369935e-05, 2.2898726e-05,
             2.1063030e-05, 3.7334241e-05, 4.5904868e-05, 1.2301322e-04,
             7.9298421e-05, 6.3840901e-08],
            [9.9984336e-01, 3.8341941e-08, 2.3892406e-05, 5.3848839e-07,
             1.5573462e-06, 4.5460902e-06, 9.5290954e-05, 1.5308196e-05,
             1.1821936e-07, 1.5372405e-05],
            [3.2380042e-06, 6.0726063e-10, 5.9453409e-06, 3.7876146e-06,
             9.7068000e-01, 2.9367786e-06, 7.5379608e-06, 5.4916769e-04,
             6.3657524e-07, 2.8746668e-02]], 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