Kanika Thombre 22070126052 AIML A3

TensorFlow:

Google created the open-source deep learning framework TensorFlow. It makes it easier to create and train machine learning models on a variety of hardware platforms, such as CPUs, GPUs, and TPUs.

Key Ideas:

Tensors: TensorFlow's basic data structures are multi-dimensional arrays. Computational graphs, in which nodes stand for operations and edges for tensors, are used to depict the data flow and processes in a model. TensorFlow 2.x introduced Eager Execution, which enables instantaneous operation execution for simpler development and debugging. Keras API: TensorFlow's high-level API that makes neural network construction and training easier.

PyTorch:

Facebook's AI Research lab created the open-source deep learning platform PyTorch. It is well-liked by developers and researchers due to its dynamic computation graph and user-friendliness.

Key Ideas:

- Tensors are the basic data structure that can take advantage of GPU acceleration, much like NumPy arrays.
- Dynamic Computation Graphs: Make it easy to design and debug models by enabling real-time graph modifications.
- Autograd is an automatic differentiation method used in neural network training that computes gradients for backpropagation.

Caffe: The Berkeley Vision and Learning Center (BVLC) created the open-source Caffe deep learning framework. It is largely focused on convolutional neural networks (CNNs) and is built for speed and modularity.

Key Ideas:

- Model Definition: Protocol buffers (Protobuf), which define the architecture and parameters, are used to define models.
- Layers: Caffe places special emphasis on layers that carry out particular network operations (such as pooling and convolution).
- Solvers: To manage optimization tasks and decide how to modify model weights during training, Caffe employs solvers.

```
import tensorflow as tf
from tensorflow import keras



# Load the MNIST dataset
mnist = keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the data
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
# Build the model
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),  # Flatten the input
    keras.layers.Dense(128, activation='relu'),   # Hidden layer with ReLU activation
    keras.layers.Dense(10, activation='softmax')   # Output layer with softmax activation
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: Use
  super().__init__(**kwargs)
```

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



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

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 6ms/step - accuracy: 0.8771 - loss: 0.4387
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.9646 - loss: 0.1212
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9766 - loss: 0.0774
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9836 - loss: 0.0546
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9875 - loss: 0.0431
<keras.src.callbacks.history.History at 0x7bfb0d9ad210>
```

```
# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
```

```
print(f'\nTest accuracy: {test_acc}')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9732 - loss: 0.0936

Test accuracy: 0.9771000146865845
```

Conclusion: The model achieved a 97.8% accuracy on the MNIST test set, demonstrating strong performance in recognizing handwritten digits. This high accuracy reflects the effectiveness of neural networks for image classification tasks on well-known datasets like MNIST.