2. Installing Keras, Tensorflow and Pytorch libraries and making use of them

AIM:

Installing Keras, Tensorflow and Pytorch libraries and making use of them

DESCRIPTION:

To install Keras, TensorFlow, and PyTorch libraries and make use of them, you can follow the steps below:

1. Install Python and pip (if not already installed)	Ensure Python is installed and download the latest version from the official website: https://www.python.org/downloads/ Python comes pre-installed with pip; install if unavailable via pip website: https://pip.pypa.io/en/stable/installing/
2. Install TensorFlow	Open a command prompt or terminal and run the following command to install TensorFlow using pip: pip install tensorflow
3. Install Keras	Keras integrated into TensorFlow, ensuring automatic installation during installation. However, you can explicitly install Keras using pip: pip install keras
4. Install PyTorch	Install PyTorch by visiting official website, selecting appropriate command based on system configuration: https://pytorch.org/get-started/locally/ For example, to install the CPU-only version of PyTorch using pip, you can run: pip install torch torchvision

5. Verify installations	After installing the libraries, you can verify that everything is set up correctly by launching a Python interpreter or creating a Python script and importing the libraries:
	import tensorflow as tf
	import keras
	import torch
	print("TensorFlow version:", tf.__version__)
	print("Keras version:", keras.__version__)
	print("PyTorch version:", torch.__version__)
	This code will output the versions of the installed libraries, confirming that everything is installed correctly.

6. Using the libraries	Install libraries, use for machine learning models training. Here's a basic example of how you can create a simple

neural network using TensorFlow/Keras and PyTorch:

- **Using TensorFlow/Keras:**

```
import tensorflow as tf
from tensorflow.keras import layers
# Create a simple neural network
model = tf.keras.Sequential([
  layers.Dense(64, activation='relu',
input_shape=(784,)),
  layers.Dense(10, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam',
          loss='sparse categorical_crossentropy',
          metrics=['accuracy'])
# Train the model (example data used here)
# model.fit(train_data, train_labels, epochs=10,
batch_size=32)
```

- **Using PyTorch:**

```
import torch
import torch.nn as nn
import torch.optim as optim
# Create a simple neural network
class SimpleNet(nn.Module):
  def __init__(self):
    super(SimpleNet, self).__init__()
    self.fc1 = nn.Linear(784, 64)
    self.fc2 = nn.Linear(64, 10)
  def forward(self, x):
    x = torch.relu(self.fc1(x))
    x = self.fc2(x)
    return x
model = SimpleNet()
# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
# Train the model (example data used here)
# for epoch in range(10):
#    running_loss = 0.0
#    for data, labels in train_loader:
#       optimizer.zero_grad()
#       outputs = model(data)
#       loss = criterion(outputs, labels)
```

```
#       loss.backward()
#       optimizer.step()
#       running_loss += loss.item()
#    print(f"Epoch {epoch+1}, Loss:
{running_loss/len(train_loader)}")
```

In these examples, we've created simple neural network architectures using TensorFlow/Keras and PyTorch, but you can build more complex models depending on your specific tasks and requirements.