Deep Learning Lab Programs

1. Setting up the Spyder IDE Environment and Executing a Python Program

Aim:

To set up spyder IDE environment using different ways.

Theory:

Python is a versatile programming language that has gained popularity due to its ease of use, wide range of libraries and frameworks, and active community of developers. Having an Integrated Development Environment (IDE) can be beneficial when working with Python, especially for larger projects. An IDE provides an environment to write, test, and debug code, making it easier to manage your projects. Spyder is one such IDE designed specifically for scientific computing with Python.

Installing Spyder IDE
Spyder IDE can be installed on Windows, macOS, and Linux operating systems. There are several ways to install the software. Installing Spyder using Anaconda

One of the easiest ways to install Spyder IDE is to use Anaconda, a popular Python distribution with many pre-installed packages, including Spyder IDE. The following steps outline how to install Spyder IDE using Anaconda:

- Download the Anaconda distribution for your operating system fromtheofficial website: https://www.anaconda.com/products/individual
- Install Anaconda by following the installation wizard.
- Once installed, open the Anaconda Navigator application.
- Click on the "Environments" tab in the left panel and select the "base (root)" environment.
- In the "base (root)" environment, search for "spyder" in the search bar.
- Click on the "Install" button next to "spyder" to install the latest version of Spyder IDE.

Installing Spyder using pip
Another way to install Spyder IDE is to use pip, a package manager for Python. The following steps outline how to install Spyder IDE using pip:

- Open the terminal/command prompt on your system.
- Type the following command to install Spyder using pip:
- `pip install Spyder`
-
- Once installed, you can open Spyder IDE by typing the following command in the terminal/command prompt:

- `spyder`
-

Installing Spyder using the official website
You can also download and install Spyder IDE from the official website: The following steps outline how to install Spyder IDE using the official website:

- Go to the official Spyder IDE website: https://www.spyder-ide.org/.
- Choose "Download" from the top menu's button selection.
- Choose the right download link for your operating system.
- Follow the installation wizard to install Spyder IDE.

Program:

Python Program to Make a Simple Calculator

In [ ]:

```
def add(P, Q):
  return P + Q
def subtract(P, Q):
  return P - Q
def multiply(P, Q):
  return P * Q
def divide(P, Q):
  return P / Q

print ("Please select the operation.")
print ("a. Add")
print ("b. Subtract")
print ("c. Multiply")
print ("d. Divide")

choice = input("Please enter choice (a/ b/ c/ d): ")

num_1 = int (input ("Please enter the first number: "))
num_2 = int (input ("Please enter the second number: "))

if choice == 'a':
  print (num_1, " + ", num_2, " = ", add(num_1, num_2))

elif choice == 'b':
  print (num_1, " - ", num_2, " = ", subtract(num_1, num_2))

elif choice == 'c':
  print (num1, " * ", num2, " = ", multiply(num1, num2))
```

```
elif choice == 'd':
   print (num_1, " / ", num_2, " = ", divide(num_1, num_2))

else:
   print ("This is an invalid input")
```

Output:

```
Please select the operation.
a. Add
b. Subtract
c. Multiply
d. Divide
20  +  10  =  30
```

Python Function to Display Calendar

In [ ]:

```
import calendar
yy = int(input("Enter year: "))
mm = int(input("Enter month: "))
print(calendar.month(yy,mm))
```

Output:

```
    August 2000
Mo Tu We Th Fr Sa Su
    1 2 3 4  5  6
 7 8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

Viva:

Question: What is Spyder IDE used for?
Answer: Spyder is an Integrated Development Environment (IDE) primarily used for scientific and data analysis programming tasks in Python.

Question: How can you install Spyder IDE?
Answer: You can install Spyder using the command pip install spyder or by installing it through Anaconda Navigator.

Question: What is the purpose of the IPython console in Spyder?
Answer: The IPython console in Spyder provides an interactive environment for running Python code, displaying results in real-time.

Question: How do you execute a Python program in Spyder?
Answer: Write your code in the editor, then press the "Run" button or use the shortcut F5 to execute the entire script.

Question: Explain the role of the Variable Explorer in Spyder.
Answer: The Variable Explorer displays the current variables in memory, their values, and types, helping users to monitor and manipulate data during program execution.

Question: What is the purpose of the File Explorer in Spyder?
Answer: The File Explorer displays the files in the current working directory and allows easy navigation and management of project files.

Question: How can you set breakpoints in your code within Spyder?
Answer: You can set breakpoints by clicking on the line number in the editor where you want the program to pause during debugging.

Question: Describe the use of the Plots pane in Spyder.
Answer: The Plots pane is used to display and interact with plots and graphs generated within the Python program.

Question: How can you inspect the documentation of a function in Spyder?
Answer: You can place the cursor on the function and press Shift + Tab to view the documentation in a pop-up window.

Question: What is the benefit of using Spyder for data analysis tasks?
Answer: Spyder's integration with tools like IPython, Variable Explorer, and plots makes it well-suited for data analysis, providing an interactive and efficient workflow.

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

Aim:
Installing Keras, Tensorflow and Pytorch libraries
Theroy:

Install Python: Ensure you have Python installed on your system. . You can download Python from the official website(https://www.python.org/downloads/) and follow the installation instructions.

Create a virtual environment (optional but recommended): It's good practice to create a virtual environment to keep your project dependencies isolated. You can use virtualenv or conda to create .Ex virtualenv : Install TensorFlow, Keras, and PyTorch: Now that you have your virtual environment set up (if you chose to use one), you can install the libraries using pip:

2. Install TensorFlow, Keras, and PyTorch: Now that you have your virtual environment set up (if you chose to use one), you can install the libraries using pip:

What is Deep Learning?
Deep learning and machine learning are part of the artificial intelligence family, though deep learning is also a subset of machine learning. Understanding these concepts is essential for any discussion of Keras vs TensorFlow vs PyTorch.

Deep learning imitates the human brain's neural pathways in processing data, using it for decision-making, detecting objects, recognizing speech, and translating languages. It learns without human supervision or intervention, pulling from unstructured and unlabeled data.

What is Keras?
Keras is an effective high-level neural network Application Programming Interface (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can run on top of CNTK, TensorFlow, and Theano. Keras focuses on being modular, user-friendly, and extensible. It doesn't handle low-level computations; instead, it hands them off to another library called the Backend.

Keras was adopted and integrated into TensorFlow in mid-2017. Users can access it via the tf.keras module. However, the Keras library can still operate separately and independently.

What is PyTorch?
PyTorch is a relatively new deep learning framework based on Torch. Developed by Facebook's AI research group and open-sourced on GitHub in 2017, it's used for natural language processing applications. PyTorch has a reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs. It also feels native, making coding more manageable and increasing processing speed.

What is TensorFlow?
TensorFlow is an end-to-end open-source deep learning framework developed by Google and released in 2015. It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android.

TensorFlow is a symbolic math library used for neural networks and is best suited for dataflow programming across a range of tasks. It offers multiple abstraction levels for building and training models.

A promising and fast-growing entry in the world of deep learning, TensorFlow offers a flexible, comprehensive ecosystem of community resources, libraries, and tools that facilitate building and deploying machine learning apps.

	Keras	**PyTorch**	**TensorFlow**
API Level	High	Low	High and Low

	Keras	PyTorch	TensorFlow
Architecture	Simple, concise, readable	Large datasets, high performance	Large datasets, high performance
Datasets	Smaller datasets	Complex, less readable	Not easy to use
Debugging	Simple network, so debugging is not often needed	Good debugging capabilities	Difficult to conduct debugging
Does It Have Trained Models?	Yes	Yes	Yes
Popularity	Most popular	Third most popular	Second most popular
Speed	Slow, low performance	Fast, high-performance	Fast, high-performance
Written In	Python	Lua	C++, CUDA, Python

Which is Better PyTorch or TensorFlow or Keras?
Everyone's situation and needs are different, so we use them based on the most for your AI project.

Let's start by installing the libraries:

1. TensorFlow: TensorFlow is a popular deep learning library developed by Google. You can install it using pip:

   `pip install tensorflow`

2. Keras: Keras is a high-level API built on top of TensorFlow, designed to make deep learning easier and more accessible. Since TensorFlow 2.0, Keras is integrated into TensorFlow, so there's no need to install it separately.

It can be imported directly from TensorFlow:

import tensorflow as tf

from tensorflow import keras

3. PyTorch: PyTorch is another popular deep learning library developed by Facebook.

You can install it using pip:

`pip install torch`

Programs:

Now, let's provide some basic usage examples:

TensorFlow/Keras Example:

In [1]:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
data = np.random.random((100, 20))
labels = np.random.randint(2, size=(100, 1))

# Create a sequential model
model = Sequential()

# Add a dense layer with one neuron (binary classification)
model.add(Dense(1, input_dim=20, activation='sigmoid'))

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

# Train the model
model.fit(data, labels, epochs=10, batch_size=10)

# Predict using the trained model
new_data = np.random.random((10, 20))
```

```
predictions = model.predict(new_data)

print("Predictions:")
print(predictions)
```

Output:

```
Epoch 1/10
10/10 [==============================] - 0s 2ms/step - loss: 0.7260 - accuracy: 0.4900
Epoch 2/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7243 - accuracy: 0.4900
Epoch 3/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7237 - accuracy: 0.4800
Epoch 4/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7220 - accuracy: 0.4800
Epoch 5/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7209 - accuracy: 0.4900
Epoch 6/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7202 - accuracy: 0.4900
Epoch 7/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7192 - accuracy: 0.5000
Epoch 8/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7187 - accuracy: 0.5000
Epoch 9/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7178 - accuracy: 0.5100
Epoch 10/10
10/10 [==============================] - 0s 1ms/step - loss: 0.7171 - accuracy: 0.5200
1/1 [==============================] - 0s 67ms/step
Predictions:
[[0.6329271 ]
 [0.47963628]
 [0.67352545]
 [0.55293083]
 [0.6328302 ]
 [0.59050083]
 [0.6455461 ]
 [0.46901232]
 [0.46941304]
 [0.54009974]]
```

PyTorch Example:

In [2]:

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim

# Create some toy data
data = torch.tensor(np.random.random((100, 20)), dtype=torch.float32)
labels = torch.tensor(np.random.randint(2, size=(100, 1)), dtype=torch.float32)

# Define a simple neural network model
model = nn.Sequential(
```

```
    nn.Linear(20, 1),
    nn.Sigmoid()
)

# Define loss function and optimizer
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
for epoch in range(10):
    optimizer.zero_grad()
    outputs = model(data)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
    print(f"Epoch [{epoch+1}/10], Loss: {loss.item():.4f}")

# Predict using the trained model
new_data = torch.tensor(np.random.random((10, 20)), dtype=torch.float32)
predictions = model(new_data)
print("Predictions:")
print(predictions)
```

output:

```
Epoch [1/10], Loss: 0.7101
Epoch [2/10], Loss: 0.7094
Epoch [3/10], Loss: 0.7088
Epoch [4/10], Loss: 0.7082
Epoch [5/10], Loss: 0.7076
Epoch [6/10], Loss: 0.7071
Epoch [7/10], Loss: 0.7066
Epoch [8/10], Loss: 0.7061
Epoch [9/10], Loss: 0.7057
Epoch [10/10], Loss: 0.7052
Predictions:
tensor([[0.4779],
        [0.4603],
        [0.4144],
        [0.4880],
        [0.5590],
        [0.4811],
        [0.4716],
        [0.5048],
        [0.4880],
        [0.4950]], grad_fn=<SigmoidBackward0>)
```

Viva:

Question 1:

Q: What is Spyder IDE?

A: Spyder IDE is an integrated development environment for Python that provides a comprehensive workspace for coding, debugging, and running Python programs.

Question 2:

2. Q: How can you install Spyder?

A: Spyder can be installed using the command pip install spyder or through Anaconda distribution.

Question 3:

3. Q: What are the main panes in Spyder's interface?

A: Spyder consists of Editor, Console, Variable Explorer, and Help panes.

Question 4:

4. Q: How do you create a new Python script in Spyder?

A: Click on "File" > "New File" in the menu, and choose "Python script".

Question 5:

5. Q: How do you execute a Python program in Spyder?

A: Write your code in the Editor pane, then click the green "Play" button or press F5.

Question 6:

6. Q: How can you set breakpoints in Spyder for debugging?

A: Click in the left margin of the Editor pane at the line where you want to set a breakpoint.

Question 7:

7. Q: What is the Variable Explorer in Spyder?

A: The Variable Explorer displays the current values of variables and allows you to inspect their contents during program execution.

Question 8:

8. Q: How can you install external libraries in Spyder?

A: Use the command !pip install library_name in the Console pane or install through the Conda package manager.

Question 9:

9. Q: How do you access documentation for a function in Spyder?

A: Place the cursor on the function and press Shift + Tab to bring up the documentation.

Question 10:

10. Q: Can you customize the Spyder interface?

A: Yes, you can customize Spyder's appearance, layout, and shortcuts through the "Preferences" menu.

3. Applying the Convolution Neural Network on computer vision problems

Aim:

To implement cnn on computer vision

Theory:

Convolutional Neural Networks (CNNs) are specialized deep learning models for computer vision tasks. They use convolutional layers to detect features and patterns in images. Pooling layers reduce spatial dimensions, while activation functions introduce non-linearity. CNN architectures combine these layers, often ending with fully connected layers for prediction. With applications like image classification, object detection, and segmentation, CNNs have revolutionized computer vision.

Program:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load and preprocess MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
train_labels, test_labels = to_categorical(train_labels), to_categorical(test_labels)

# Build and compile CNN model
model = models.Sequential([
    layers.Input(shape=(28, 28, 1)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# Train the model
model.fit(train_images, train_labels, epochs=5, batch_size=64, validation_split=0.2)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc}")
```

output:

```
In [1]: runfile('C:/Users/sheri/.spyder-py3/temp.py', wdir='C:/Users/sheri/.spyder-py3')
Epoch 1/5
750/750 [==============================] - 28s 36ms/step - loss: 0.2143 - accuracy: 0.9345 - val_loss: 0.0827 - val_accuracy: 0.9740
Epoch 2/5
750/750 [==============================] - 25s 33ms/step - loss: 0.0578 - accuracy: 0.9818 - val_loss: 0.0637 - val_accuracy: 0.9815
Epoch 3/5
750/750 [==============================] - 33s 44ms/step - loss: 0.0403 - accuracy: 0.9876 - val_loss: 0.0427 - val_accuracy: 0.9873
Epoch 4/5
750/750 [==============================] - 47s 63ms/step - loss: 0.0310 - accuracy: 0.9901 - val_loss: 0.0447 - val_accuracy: 0.9859
Epoch 5/5
750/750 [==============================] - 27s 35ms/step - loss: 0.0256 - accuracy: 0.9923 - val_loss: 0.0488 - val_accuracy: 0.9863
313/313 [==============================] - 2s 6ms/step - loss: 0.0366 - accuracy: 0.9890
Test accuracy: 0.9890000224113464
```

Viva:

1. What's the advantage of using Convolutional Neural Networks (CNNs) in computer vision over traditional methods?

Answer: CNNs automatically learn features from data, eliminating the need for manual feature engineering, unlike traditional methods.

2. How do pooling layers contribute to CNNs?

Answer: Pooling layers downsample feature maps, reducing computational load while maintaining essential information and introducing translation invariance.

3. How does a Convolutional Neural Network (CNN) differ from a traditional feedforward neural network for computer vision tasks?

Answer: CNNs use specialized layers like convolution and pooling to automatically learn spatial features, making them more effective for visual data compared to traditional neural networks.

4. Explain the concept of transfer learning in the context of CNNs.

Answer: Transfer learning involves utilizing pre-trained CNN models on large datasets and fine-tuning them for specific tasks with smaller datasets, benefiting from learned features.

5. What role does the activation function (e.g., ReLU) play in a CNN?

Answer: The activation function introduces non-linearity, allowing CNNs to capture complex relationships in data, which is crucial for tasks like image recognition.

6. In CNNs, why is data augmentation used during training?

Answer: Data augmentation involves applying random transformations (rotation, flipping) to training data, increasing dataset diversity and improving model generalization.

7. Can you explain the purpose of padding in CNNs?

Answer: Padding adds extra pixels to the input before convolution, preserving spatial dimensions and preventing information loss at the edges.

8. What's the significance of the loss function in training a CNN?

Answer: The loss function quantifies the difference between predicted and actual values, guiding the network's weight updates during optimization to improve performance.

9. How do CNNs handle translation invariance in image data?

Answer: CNNs achieve translation invariance through pooling layers, which aggregate local features and their variations across an image, making the network more robust to position changes.

10. What are some real-world applications of CNNs beyond image classification?

Answer: CNNs find use in object detection, semantic segmentation, medical image analysis, autonomous vehicles, and various tasks involving structured data with spatial relationships.

4. Image classification on MNIST dataset (CNN model with Fully connected layer)

Aim:

To implement image classification on MNIST dataset

Theory:

Image classification using CNN on the MNIST dataset involves building a Convolutional Neural Network. The CNN includes convolutional and pooling layers that learn relevant features from the handwritten digit images. The final fully connected layers aggregate these features and make predictions about the digit classes. Training involves optimizing weights using a loss function like softmax cross-entropy, resulting in a model capable of accurately classifying handwritten digits.

Program:

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical

# Load and preprocess MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
train_labels, test_labels = to_categorical(train_labels), to_categorical(test_labels)

# Build and compile CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train and evaluate the model
model.fit(train_images, train_labels, epochs=5, batch_size=64, validation_split=0.2)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc}")
```

output:

```
Epoch 1/5
750/750 [==============================] - 46s 60ms/step - loss: 0.2153 - accuracy: 0.9336 - val_loss: 0.0688 - val_accuracy: 0.9803
Epoch 2/5
750/750 [==============================] - 30s 40ms/step - loss: 0.0581 - accuracy: 0.9821 - val_loss: 0.0567 - val_accuracy: 0.9819
Epoch 3/5
750/750 [==============================] - 24s 32ms/step - loss: 0.0419 - accuracy: 0.9867 - val_loss: 0.0491 - val_accuracy: 0.9861
Epoch 4/5
750/750 [==============================] - 24s 32ms/step - loss: 0.0314 - accuracy: 0.9899 - val_loss: 0.0430 - val_accuracy: 0.9877
Epoch 5/5
750/750 [==============================] - 24s 32ms/step - loss: 0.0274 - accuracy: 0.9911 - val_loss: 0.0408 - val_accuracy: 0.9875
313/313 [==============================] - 2s 5ms/step - loss: 0.0318 - accuracy: 0.9885
Test accuracy: 0.9884999990463257
```

Viva:

1. Question: In image classification using a CNN, why is it important to preprocess the input images before training?

Answer: Preprocessing, like normalizing pixel values, ensures consistent data range and helps the network converge faster during training.

2. Question: How do you define the architecture of a CNN for MNIST digit classification? Mention the types of layers and their sequence.

Answer: The architecture includes convolutional layers for feature extraction, pooling layers for downsampling, followed by fully connected layers for classification.

3. Question: What's the significance of the "relu" activation function in the convolutional layers of a CNN?

Answer: ReLU introduces non-linearity, allowing the network to capture complex patterns and make it more capable of learning from data.

4. Question: Describe the process of backpropagation in the context of training a CNN model.

Answer: Backpropagation involves calculating gradients of the loss with respect to model weights. These gradients guide weight updates during optimization.

5. Question: How do you prevent overfitting when training a CNN on the MNIST dataset?

Answer: Techniques like dropout and data augmentation can be used. Dropout randomly deactivates neurons during training, and data augmentation introduces variations to the training data.

6. Question: Explain how the pooling layers contribute to reducing the dimensionality of feature maps in a CNN.

Answer: Pooling layers reduce spatial dimensions by selecting the maximum or average value in local regions, helping to retain important information while decreasing computation.

7. Question: In the context of CNNs, what is transfer learning, and why is it useful?

Answer: Transfer learning involves using pre-trained models for new tasks. It's useful because pre-trained models have learned generic features that can be fine-tuned for specific tasks, often requiring less training data.

8. Question: Describe the role of loss functions in training a CNN for image classification.

Answer: Loss functions quantify the difference between predicted and actual outputs. The goal is to minimize this difference during training to improve the model's accuracy.

9. Question: How does padding impact the size of feature maps in a CNN?

Answer: Padding adds extra pixels to the input, maintaining spatial dimensions in the feature maps after convolution operations.

10. Question: Could you explain how a fully connected layer operates in the context of a CNN's architecture?

Answer: Fully connected layers aggregate features learned from previous layers and make final predictions. They map the extracted features to class probabilities for image classification tasks.