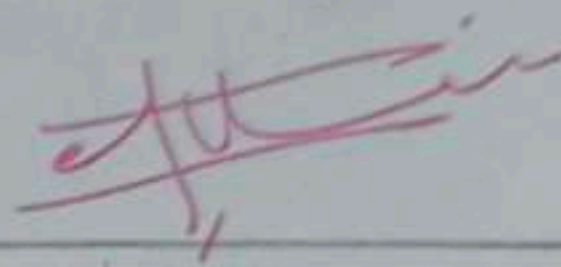
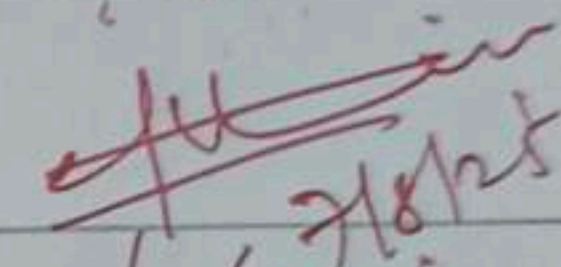
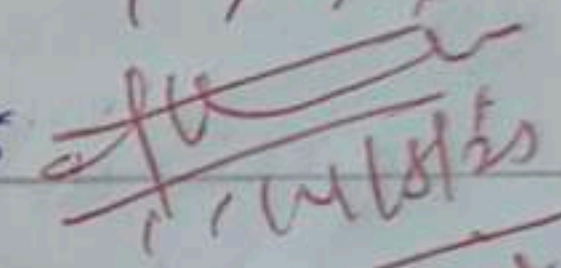
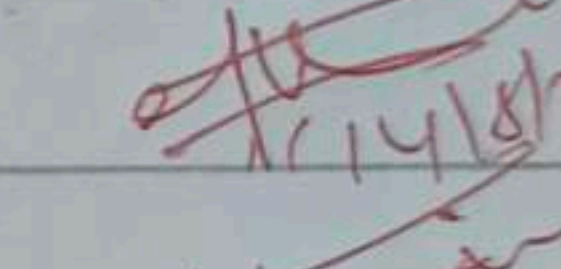
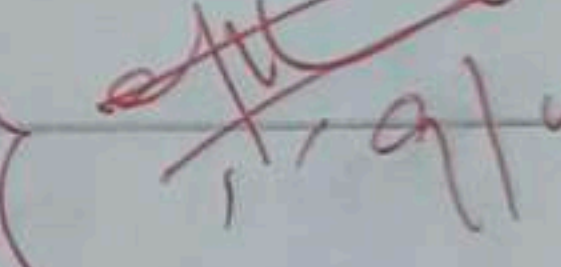
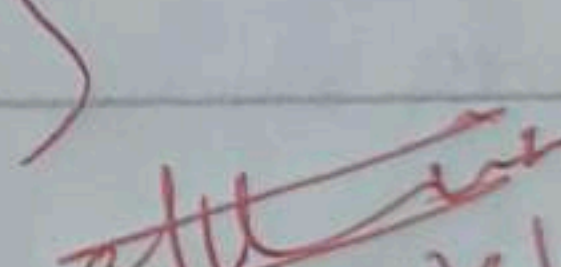
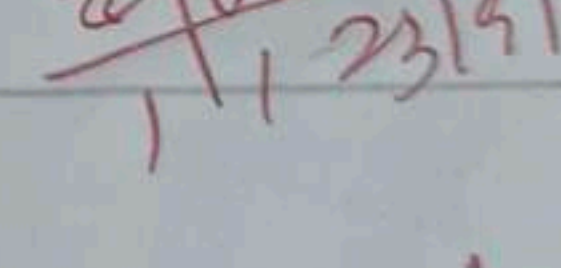
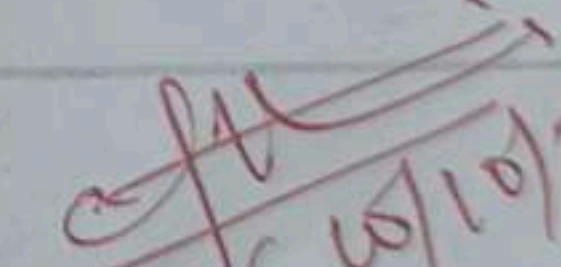
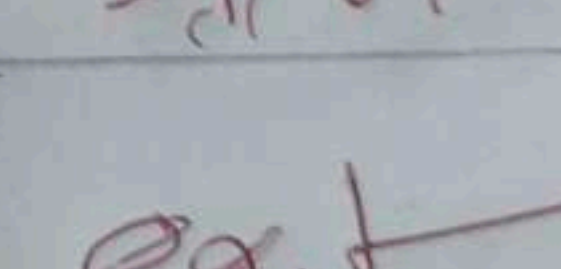
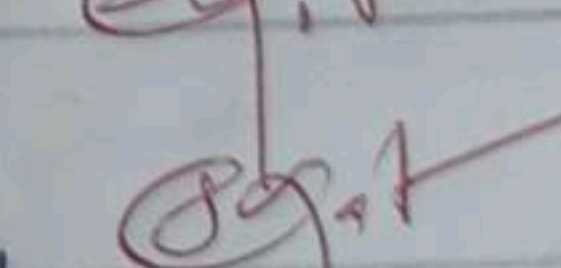
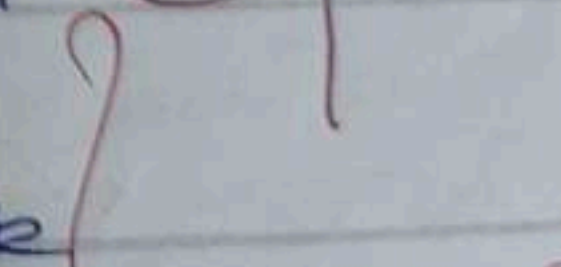


~~Completed~~

D. Khaja Nawaz

RA2311047010037

Deep Learning Techniques (Lab)

Date	Title	Sign
24/07/2025	1. Exploring the deep learning platform	
31/07/25	2. Implement a classifier using open-source dataset	
31/07/25	3. Study of the classifiers with respect to statistical parameters	
14/08/25	4. Build a simple feed forward neural network to recognize handwritten characters	
22/08/25	5. Study of Activation functions and their role	
09/09/25	6. Implement gradient descent and backpropagation in deep neural network	
16/09/25	7. Build a CNN model to classify cat and Dog image	
30/09/2025	8. Experiment of LSTM	
30/09/2025	9. Build a Recurrent Neural Network	
9/10/2025	10. perform compression on MNIST dataset using autoencoder	
9/10/2025	11. Experiments using variational autoen	
03/11/25	12. Implement a Deep Convolutional GAN to generate complex color images	
03/11/25	13. Understanding the architecture of pre-trained model	
03/11/25	14. Implement a pre-trained CNN model as feature Extractor using	
03/11/25	15. Implement a yolo model to detect objects	

03/11/25

Lab: 14

Implement a pre-trained CNN model as a Feature Extractor using transfer learning models.

* Aim :

To implement a pre-trained CNN model as a feature extractor using transfer learning, where the convolutional layers are frozen and only the final classifier is trained for a new task.

* Objective :

1. Use a pre-trained model (ResNet18) for image net
2. Freeze convolutional layers to use them as feature extractor
3. Replace and train the final classification layer on a new dataset
4. Evaluate accuracy and visualize training and testing loss.

* pseudocode :

1. Load CIFAR-10 dataset and preprocess images
2. Load pre-trained ResNet18 model
3. Replace final layer with new classifier (10 classes).
4. Freeze all convolutional layers.
5. Define loss (cross entropy) and optimizer (Adam)

*output

Epoch(1/5), loss: 1.0176, Test accuracy: 76.7.

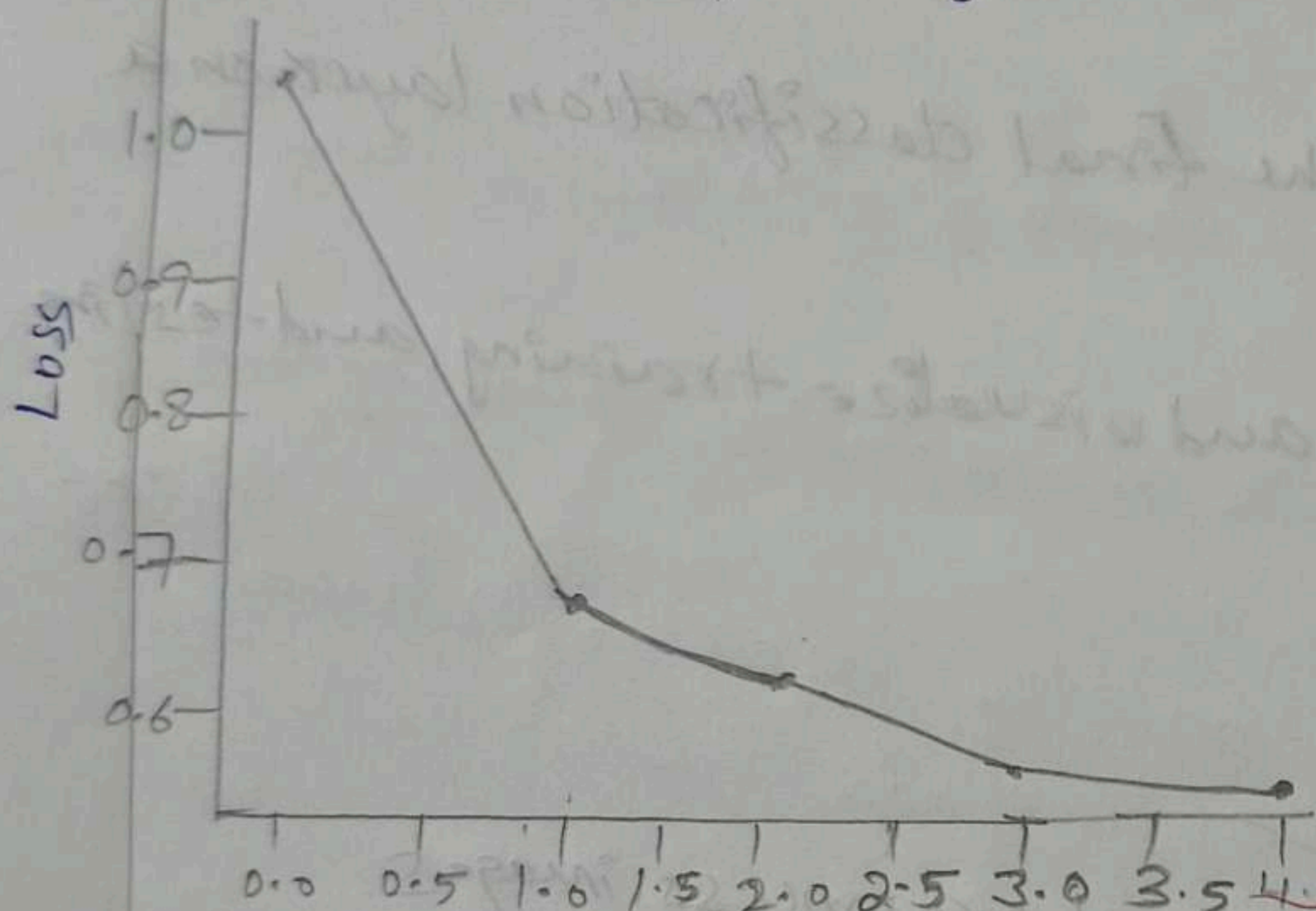
Epoch(2/5), loss: 0.7011, Test accuracy: 77.27.

Epoch(3/5), loss: 0.6392, Test accuracy: 77.12.

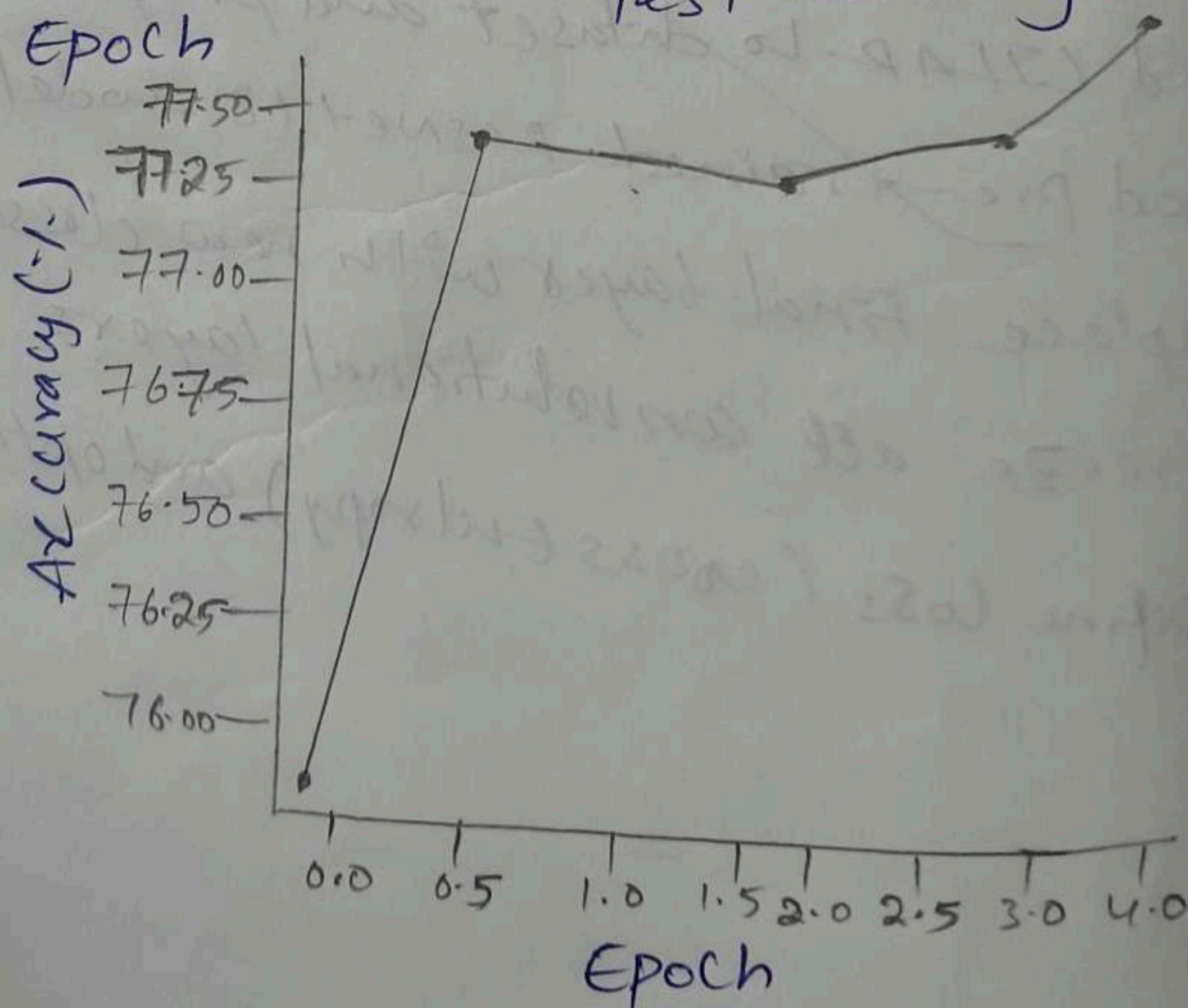
Epoch(4/5), loss: 0.6121, Test accuracy: 77.17.

Epoch(5/5), loss: 0.6022, Test accuracy: 77.61.

Training loss



Test Accuracy



6.
For 10 epochs:

- Train on training data and compute loss
- Evaluate on test data for loss and accuracy

7. plot training vs test loss graph

* observation :-

- The pre-trained ResNet18 achieved good accuracy even with few epochs
- Training loss decreased steadily, confirming effective feature use.
- Test accuracy remained stable, showing generalization from pre-learned features

* result :-

successfully implemented a pre-trained model as feature extractor using transfer learning models

~~11/11~~



```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
from tqdm import tqdm

# ✅ 1. Use smaller images for speed
transform = transforms.Compose([
    transforms.Resize((128, 128)), # smaller input size
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                          std=[0.229, 0.224, 0.225])
])

# ✅ 2. Use only 10k samples for faster training
train_data_full = datasets.CIFAR10(root='./data', train=True, transform=transform, download=True)
test_data = datasets.CIFAR10(root='./data', train=False, transform=transform, download=True)
train_data = torch.utils.data.Subset(train_data_full, range(10000))

train_loader = DataLoader(train_data, batch_size=64, shuffle=True, num_workers=2)
test_loader = DataLoader(test_data, batch_size=64, shuffle=False, num_workers=2)

# ✅ 3. Load smaller pre-trained model (MobileNetV2)
model = models.mobilenet_v2(pretrained=True)
for param in model.parameters():
    param.requires_grad = False # freeze base layers

# ✅ 4. Replace final classifier for CIFAR-10 (10 classes)
model.classifier[1] = nn.Linear(model.classifier[1].in_features, 10)

# ✅ 5. Define loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier[1].parameters(), lr=0.001)

# ✅ 6. Use GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
model.to(device)

# ✅ Lists to store progress
train_losses = []
test accuracies = []

# ✅ 7. Train loop with tqdm progress bar
num_epochs = 5
```

```

model.train()
running_loss = 0.0
loop = tqdm(train_loader, desc=f"Epoch [{epoch+1}/{num_epochs}]", leave=False)

for images, labels in loop:
    images, labels = images.to(device), labels.to(device)
    optimizer.zero_grad()
    outputs = model(images)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
    running_loss += loss.item()
    loop.set_postfix(loss=loss.item())

avg_loss = running_loss / len(train_loader)
train_losses.append(avg_loss)

# ✅ Evaluate accuracy after each epoch
model.eval()
correct, total = 0, 0
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
acc = 100 * correct / total
test accuracies.append(acc)

print(f"Epoch [{epoch+1}/{num_epochs}] → Loss: {avg_loss:.4f} | Test Accuracy: {acc:.2f}%")

# ✅ 8. Plot Loss & Accuracy graphs
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(train_losses, marker='o')
plt.title('Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')

plt.subplot(1,2,2)
plt.plot(test_accuracies, marker='o', color='orange')
plt.title('Test Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy (%)')

plt.tight_layout()
plt.show()

```

```
100%|██████████| 170M/170M [00:06<00:00, 27.3MB/s]
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in a future version.
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/mobilenet_v2-b0353104.pth" to /root/.cache/torch/hub/checkpoints/mobilenet_v2-b0353104.pth
100%|██████████| 13.6M/13.6M [00:00<00:00, 120MB/s]
Using device: cuda
Epoch [1/5] → Loss: 1.0176 | Test Accuracy: 76.00%
Epoch [2/5] → Loss: 0.7011 | Test Accuracy: 77.27%
Epoch [3/5] → Loss: 0.6392 | Test Accuracy: 77.12%
Epoch [4/5] → Loss: 0.6121 | Test Accuracy: 77.17%
Epoch [5/5] → Loss: 0.6032 | Test Accuracy: 77.64%
```

