Convolution Operation.

$$y_{ij}^{(k)} = \sum^{C} \sum^{M} \sum^{N} x^{(c)}_{i+m-1, j+n-1} \cdot w^{(c,k)}_{m,n} + b^{(k)}$$

$y_{i,j}^{(k)}$ = output feature map at position (i,j) for filter (k)

$x^{(c)}$ = I/p image/ch - c

$w^{(c,k)}$ = kernal w of I/P chanale & filter k

$b^{(k)}$ = bias for Filter k

$M \times N$ = kernal size

C = I/P ch.

Exp 14

IMPLEMENT A PRE-TRAINED CNN AS A FEATURE EXTRACTOR USING TRANSFER LEARNING

AIM

To implement transfer learning by using pre-trained CNN (ResNet-18) as a feature extractor and train a classifier on a new data (CIFAR-10) for improved accuracy and faster convergence.

OBJECTIVE

* Load a pre trained CNN (ResNet-18)
* Freeze convolutional layers to use them as feature extractors.
* Replace the final classification layer to match the target dataset (eg 10 classes of CIFAR-10)
* Train the new classifier using the extracted features.
* Visualise accuracy, loss curve and sample predictions

PSEUDOCODE

↳ Import libraries (

↳ Load pre-trained model (ResNet-18)

↳ Freeze all convolutional layers

↳ Replace the final fully connected layer with a new classifier for CIFAR-10

↳ Load CIFAR-10 dataset with transforms

↳ Define loss function (Cross Entropy) and optimizer (Adam)

↳ Training loop:

For each epoch:

Forward pass, Compute loss, Back pass.

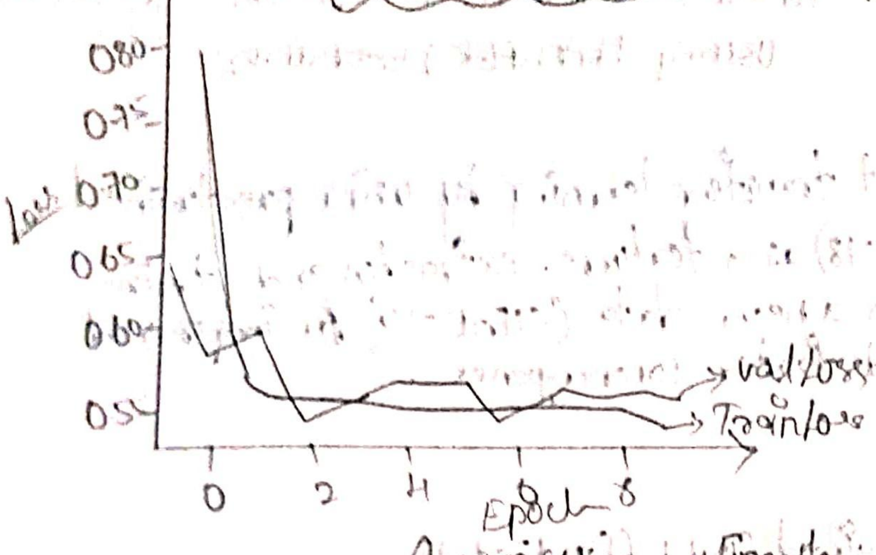
Update only classifier weights.

Record training loss and accuracy.

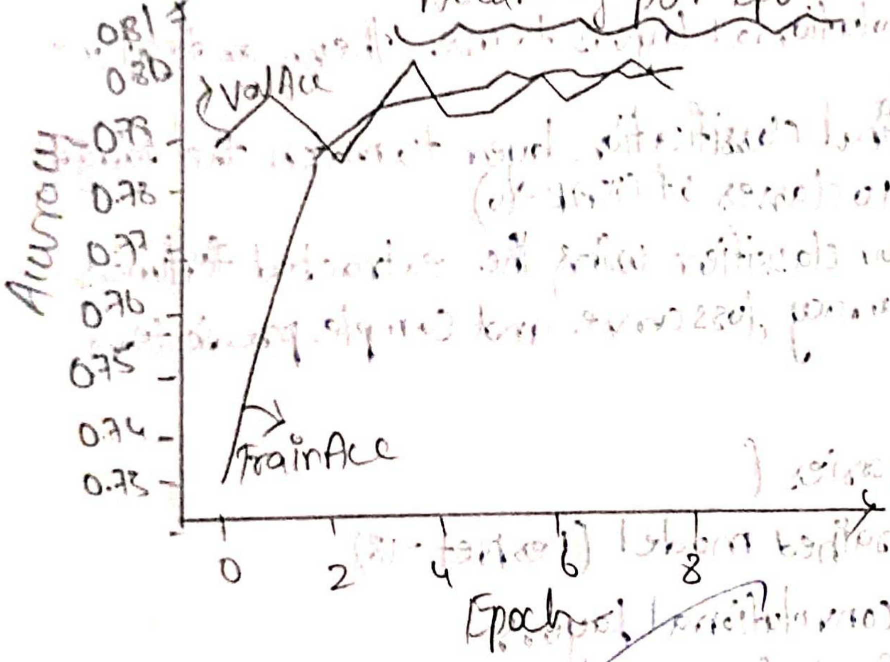
↳ Evaluate model on test dataset

↳ Plot training & validation accuracy / loss curves.

Loss per Epoch



Accuracy per Epoch



↳ Display sample predictions with input images

OBSERVATIONS:

Epoch [1/10] Train Loss | 0.8327 | Train Acc 0.7293 | Val Loss 0.6272 | Acc .7892

Epoch [2/10] Train Loss | 0.6185 | Train Acc 0.7870 | Val Loss 0.5881 | Acc .8012

Epoch [3/10] Train Loss | 0.5911 | Train Acc 0.7962 | Val Loss 0.5974 | Acc .7929

Epoch [4/10] Train Loss | 0.5770 | Train Acc 0.7998 | Val Loss 0.5691 | Acc .8078

Epoch [5/10] Train Loss | 0.5677 | Train Acc 0.8018 | Val Loss 0.5832 | Acc .802

⋮

Epoch [9/10] Train Loss | 0.5513 | Train Acc 0.8085 | Val Loss 0.5591 | Acc .8100

Epoch [10/10] Train Loss | 0.5449 | Train Acc 0.8106 | Val Loss 0.5016 | Acc .8092

RESULT:

Successfully implemented a pre-trained CNN as a feature extractor using transfer learning.

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

Device setup

```python
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Device:", device)
```

```
Device: cuda
```

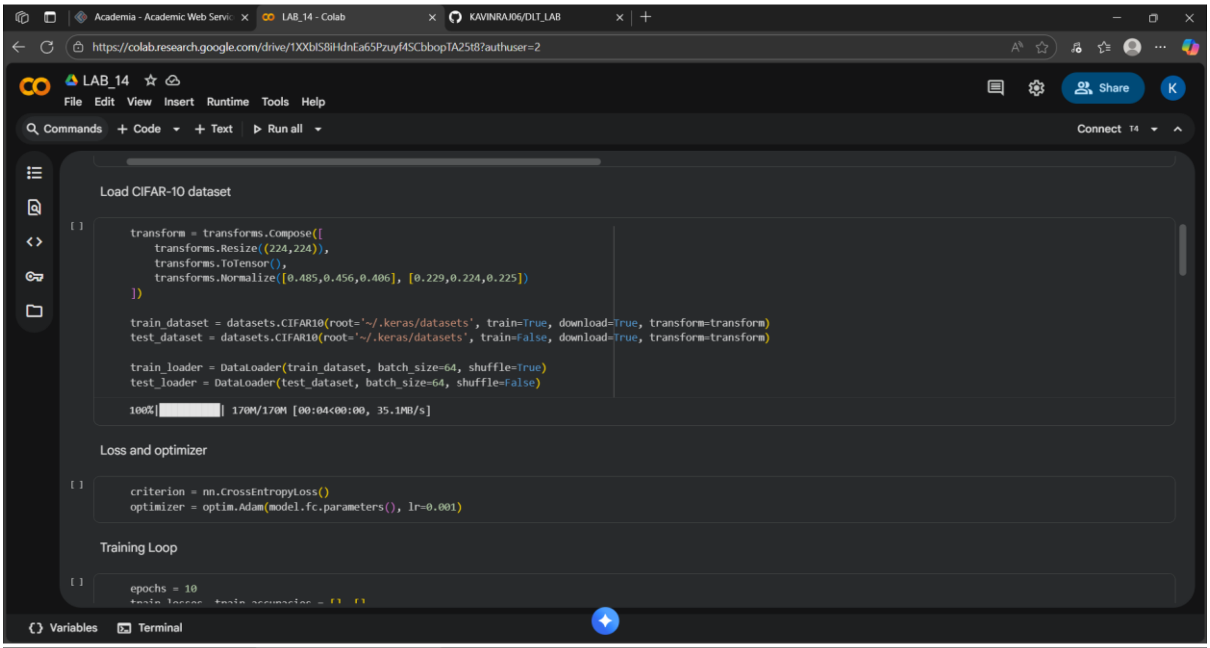
Load pre-trained model

```python
model = models.resnet18(pretrained=True)
# Freeze all convolutional layers
for param in model.parameters():
    param.requires_grad = False

# Replace the final layer for CIFAR-10 (10 classes)
num_features = model.fc.in_features
model.fc = nn.Linear(num_features, 10)
model = model.to(device)
```

```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
/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 the future,
  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 since 0.13 a
  warnings.warn(msg)
100%|██████████| 44.7M/44.7M [00:00<00:00, 199MB/s]
```

Load CIFAR-10 dataset

```python
transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485,0.456,0.406], [0.229,0.224,0.225])
```



Load CIFAR-10 dataset

```python
transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485,0.456,0.406], [0.229,0.224,0.225])
])

train_dataset = datasets.CIFAR10(root='~/.keras/datasets', train=True, download=True, transform=transform)
test_dataset = datasets.CIFAR10(root='~/.keras/datasets', train=False, download=True, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

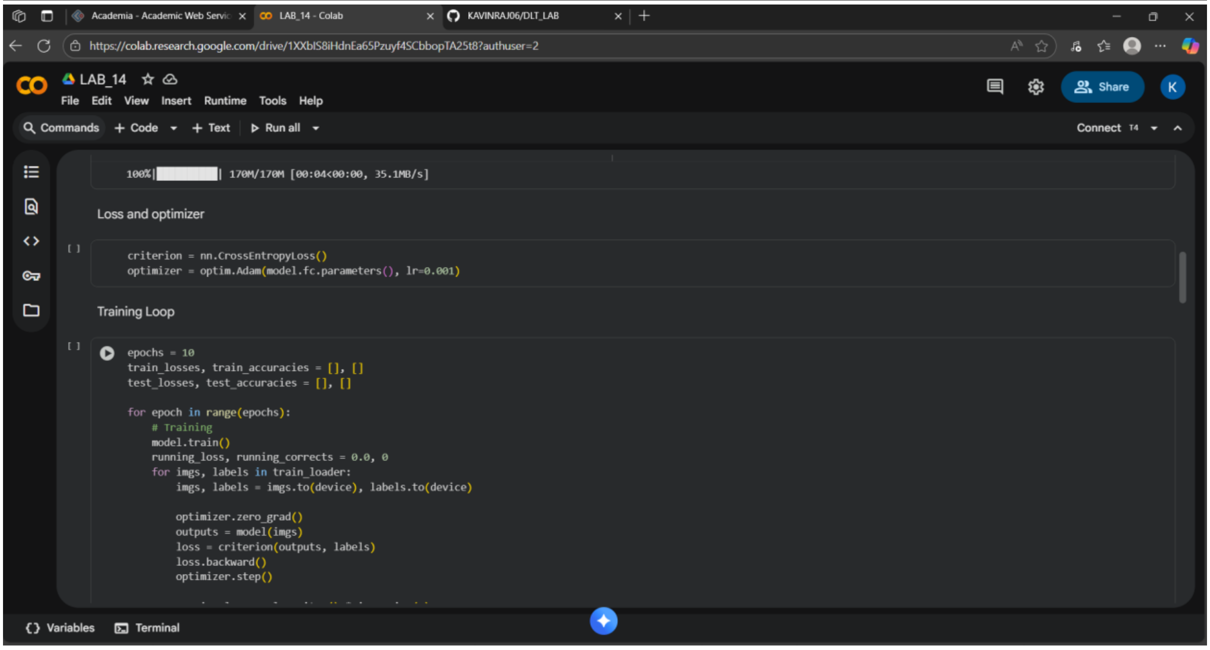
```
100%|██████████| 170M/170M [00:04<00:00, 35.1MB/s]
```

Loss and optimizer

```python
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=0.001)
```

Training Loop

```python
epochs = 10
```



```
100%|██████████| 170M/170M [00:04<00:00, 35.1MB/s]
```

Loss and optimizer

```python
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=0.001)
```

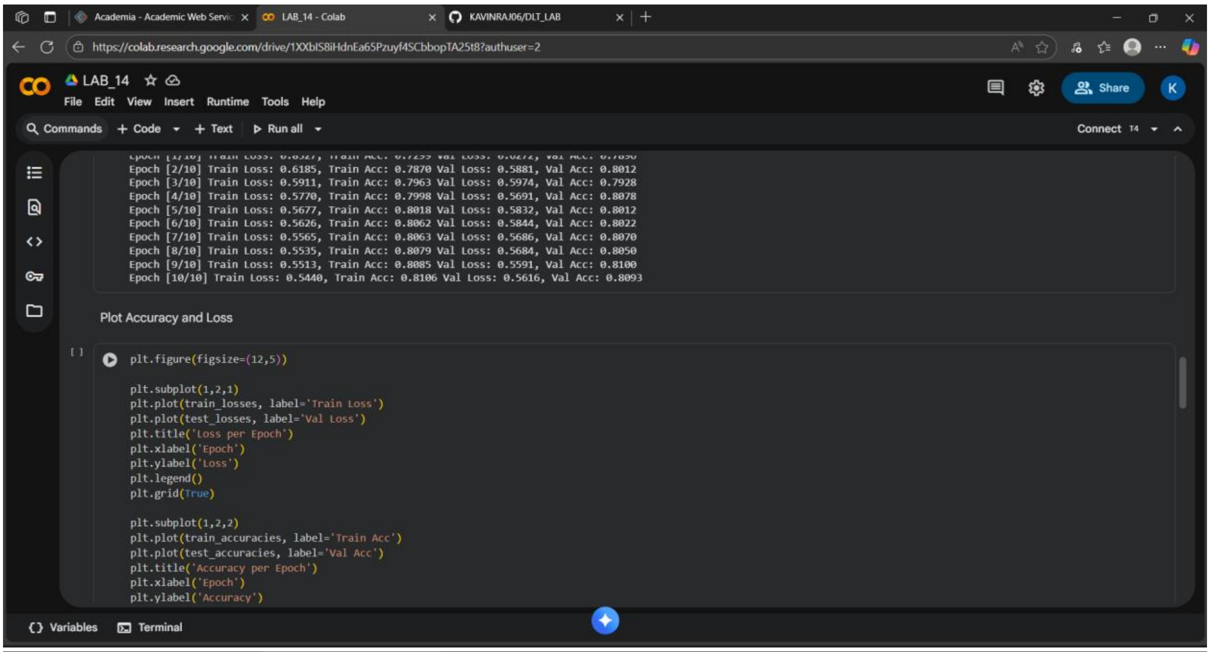
Training Loop

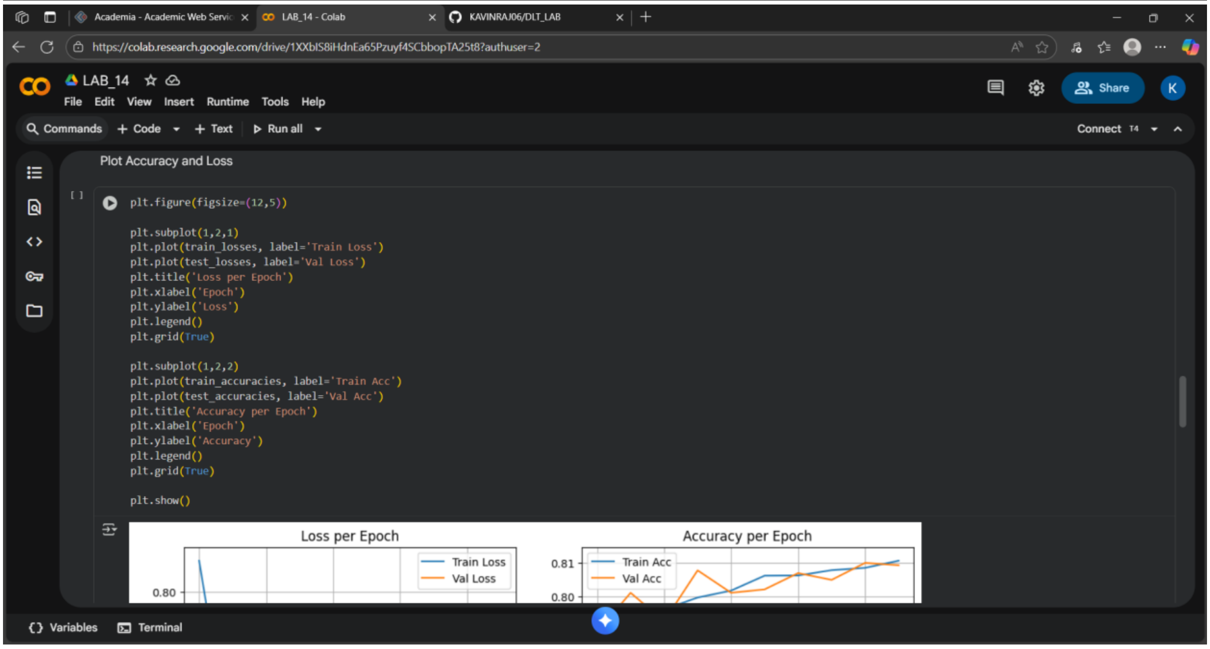
```python
epochs = 10
train_losses, train_accuracies = [], []
test_losses, test_accuracies = [], []

for epoch in range(epochs):
    # Training
    model.train()
    running_loss, running_corrects = 0.0, 0
    for imgs, labels in train_loader:
        imgs, labels = imgs.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(imgs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

```
    for imgs, labels in train_loader:
        imgs, labels = imgs.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(imgs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item() * imgs.size(0)
        running_corrects += (outputs.argmax(1) == labels).sum().item()

    epoch_loss = running_loss / len(train_dataset)
    epoch_acc = running_corrects / len(train_dataset)
    train_losses.append(epoch_loss)
    train_accuracies.append(epoch_acc)

    # Validation
    model.eval()
    val_loss, val_corrects = 0.0, 0
    with torch.no_grad():
        for imgs, labels in test_loader:
            imgs, labels = imgs.to(device), labels.to(device)
            outputs = model(imgs)
            loss = criterion(outputs, labels)
            val_loss += loss.item() * imgs.size(0)
            val_corrects += (outputs.argmax(1) == labels).sum().item()

    val_epoch_loss = val_loss / len(test_dataset)
    val_epoch_acc = val_corrects / len(test_dataset)
```

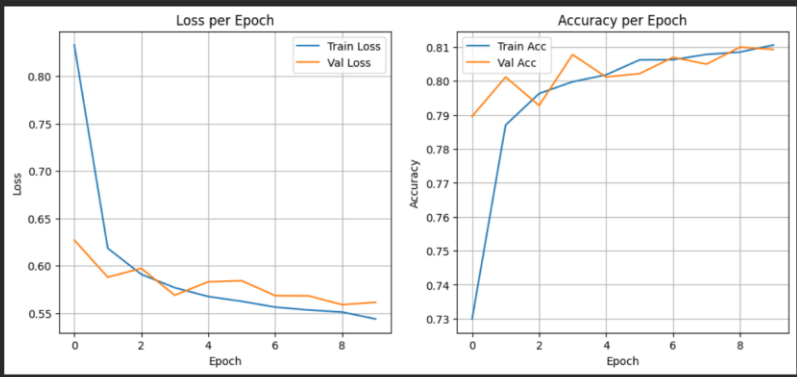
```
    # Validation
    model.eval()
    val_loss, val_corrects = 0.0, 0
    with torch.no_grad():
        for imgs, labels in test_loader:
            imgs, labels = imgs.to(device), labels.to(device)
            outputs = model(imgs)
            loss = criterion(outputs, labels)
            val_loss += loss.item() * imgs.size(0)
            val_corrects += (outputs.argmax(1) == labels).sum().item()

    val_epoch_loss = val_loss / len(test_dataset)
    val_epoch_acc = val_corrects / len(test_dataset)
    test_losses.append(val_epoch_loss)
    test_accuracies.append(val_epoch_acc)

    print(f"Epoch [{epoch+1}/{epochs}] "
          f"Train Loss: {epoch_loss:.4f}, Train Acc: {epoch_acc:.4f} "
          f"Val Loss: {val_epoch_loss:.4f}, Val Acc: {val_epoch_acc:.4f}")
```

```
Epoch [1/10] Train Loss: 0.8327, Train Acc: 0.7299 Val Loss: 0.6272, Val Acc: 0.7896
Epoch [2/10] Train Loss: 0.6185, Train Acc: 0.7870 Val Loss: 0.5881, Val Acc: 0.8012
Epoch [3/10] Train Loss: 0.5911, Train Acc: 0.7963 Val Loss: 0.5974, Val Acc: 0.7928
Epoch [4/10] Train Loss: 0.5770, Train Acc: 0.7998 Val Loss: 0.5691, Val Acc: 0.8078
Epoch [5/10] Train Loss: 0.5677, Train Acc: 0.8018 Val Loss: 0.5832, Val Acc: 0.8012
Epoch [6/10] Train Loss: 0.5626, Train Acc: 0.8062 Val Loss: 0.5844, Val Acc: 0.8022
Epoch [7/10] Train Loss: 0.5565, Train Acc: 0.8063 Val Loss: 0.5686, Val Acc: 0.8070
Epoch [8/10] Train Loss: 0.5535, Train Acc: 0.8079 Val Loss: 0.5684, Val Acc: 0.8050
Epoch [9/10] Train Loss: 0.5513, Train Acc: 0.8085 Val Loss: 0.5591, Val Acc: 0.8100
Epoch [10/10] Train Loss: 0.5440, Train Acc: 0.8106 Val Loss: 0.5616, Val Acc: 0.8093
```


Academia - Academic Web Servic
LAB_14 - Colab
KAVINRAJ06/DLT_LAB
https://colab.research.google.com/drive/1XXbIS8iHdnEa65Pzuyf4SCbbopTA25t8?authuser=2
LAB_14
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Connect
Epoch [2/10] Train Loss: 0.6185, Train Acc: 0.7870 Val Loss: 0.5881, Val Acc: 0.8012
Epoch [3/10] Train Loss: 0.5911, Train Acc: 0.7963 Val Loss: 0.5974, Val Acc: 0.7928
Epoch [4/10] Train Loss: 0.5770, Train Acc: 0.7998 Val Loss: 0.5691, Val Acc: 0.8078
Epoch [5/10] Train Loss: 0.5677, Train Acc: 0.8018 Val Loss: 0.5832, Val Acc: 0.8012
Epoch [6/10] Train Loss: 0.5626, Train Acc: 0.8062 Val Loss: 0.5844, Val Acc: 0.8022
Epoch [7/10] Train Loss: 0.5565, Train Acc: 0.8063 Val Loss: 0.5686, Val Acc: 0.8070
Epoch [8/10] Train Loss: 0.5535, Train Acc: 0.8079 Val Loss: 0.5684, Val Acc: 0.8050
Epoch [9/10] Train Loss: 0.5513, Train Acc: 0.8085 Val Loss: 0.5591, Val Acc: 0.8100
Epoch [10/10] Train Loss: 0.5440, Train Acc: 0.8106 Val Loss: 0.5616, Val Acc: 0.8093
Plot Accuracy and Loss
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Val Loss')
plt.title('Loss per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.subplot(1,2,2)
plt.plot(train_accuracies, label='Train Acc')
plt.plot(test_accuracies, label='Val Acc')
plt.title('Accuracy per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
Variables
Terminal

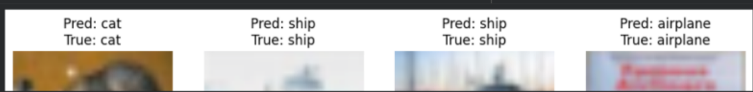

Academia - Academic Web Servic
LAB_14 - Colab
KAVINRAJ06/DLT_LAB
https://colab.research.google.com/drive/1XXbIS8iHdnEa65Pzuyf4SCbbopTA25t8?authuser=2
LAB_14
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Connect
Plot Accuracy and Loss
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Val Loss')
plt.title('Loss per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.subplot(1,2,2)
plt.plot(train_accuracies, label='Train Acc')
plt.plot(test_accuracies, label='Val Acc')
plt.title('Accuracy per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
Loss per Epoch
Train Loss
Val Loss
0.80
Accuracy per Epoch
Train Acc
Val Acc
0.81
0.80
Variables
Terminal

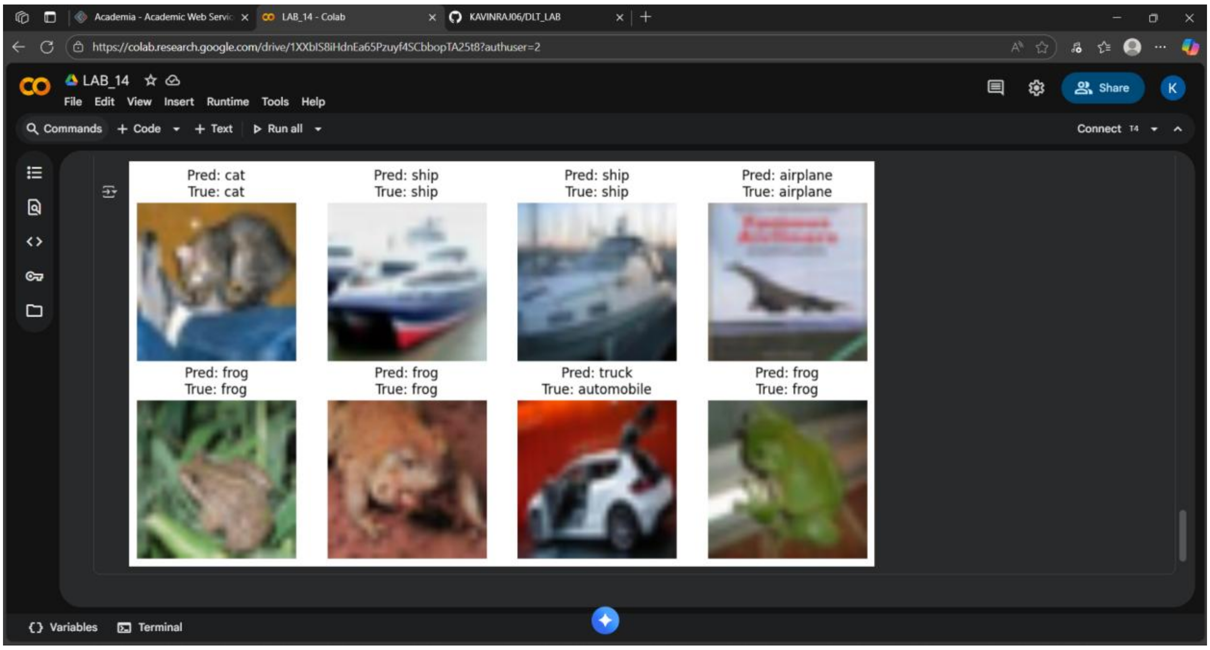
```
plt.show()
```



Display sample predictions

```
import matplotlib.pyplot as plt

classes = train_dataset.classes
model.eval()
imgs, labels = next(iter(test_loader))
imgs, labels = imgs.to(device), labels.to(device)
outputs = model(imgs)
preds = outputs.argmax(1)

plt.figure(figsize=(12,6))
for i in range(8):
    plt.subplot(2,4,i+1)
    img_np = imgs[i].cpu().permute(1,2,0).numpy()
    img_np = img_np * np.array([0.229,0.224,0.225]) + np.array([0.485,0.456,0.406])  # denormalize
    img_np = np.clip(img_np,0,1)
    plt.imshow(img_np)
    plt.title(f"Pred: {classes[preds[i]]}\nTrue: {classes[labels[i]]}")
    plt.axis('off')
plt.show()
```




Academia - Academic Web Servic
LAB_14 - Colab
KAVINRAJ06/DLT_LAB
https://colab.research.google.com/drive/1XXbIS8iHdnEa65Pzuyf4SCbbopTA25t8?authuser=2
LAB_14
File Edit View Insert Runtime Tools Help
Share
Commands
Code
Text
Run all
Connect
Pred: cat
True: cat
Pred: ship
True: ship
Pred: ship
True: ship
Pred: airplane
True: airplane
Pred: frog
True: frog
Pred: frog
True: frog
Pred: truck
True: automobile
Pred: frog
True: frog
Variables
Terminal