Exp 14. Implement a pre-trained CNN model as a feature extractor using transfer learning

Aim:

Implementing a pre-trained CNN model as a feature extractor using transfer learning.

Pseudo code:

1. Import necessary libraries (torch, torchvision, matplotlib, etc.)
2. Load the pre-trained CNN model (eg: ResNet50 or VGG16) from torchvision.models
3. Freeze the convolutional base to retain pre-trained weights.
4. Replace the final classification layer with a new one suitable for the new dataset's no. of classes
5. Load the dataset and apply image transformations (resize, normalize)
6. Extract features using the frozen model
7. Train only the newly added fully connected layers.
8. Evaluate model performance using accuracy & f1-score.
9. Visualize training/validation accuracy & loss.

Justification:-

CNNs are deep learning architectures widely used for img. classification & recognition:

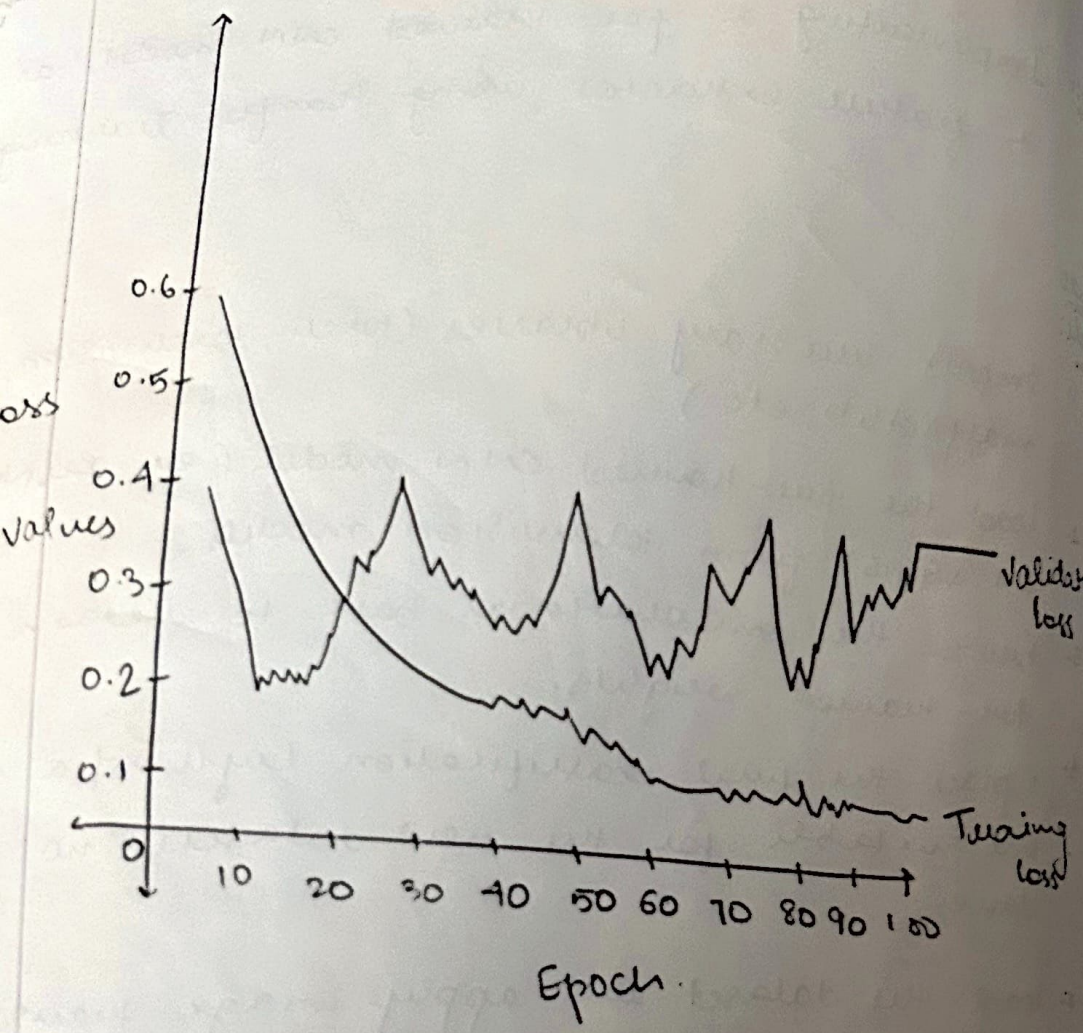
They consist of:

- convolutional layers for feature extraction
- Pooling layers for downsampling

- fully connected layer for classification

Result: Program implemented successfully.

Output


0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100
Epoch
Validation loss
Training loss

Safari File Edit View History Bookmarks Window Help Tue Nov 4 7:45 PM

colab.research.google.com

clean coal technology diagrams - Google Search | SRM Student Portal | Untitled8.ipynb - Colab | 21AIC301J- Lab AI-A

Untitled8.ipynb Saving failed since 7:38 PM

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

Epoch

[2]

```
import torch
import torch.nn as nn
import torchvision.models as models
from torch.utils.data import DataLoader, TensorDataset
from sklearn.metrics import accuracy_score, f1_score, recall_score
import matplotlib.pyplot as plt
import numpy as np
class PretrainedCNNFeatureExtractor(nn.Module):
    def __init__(self, output_dim=2):
        super(PretrainedCNNFeatureExtractor, self).__init__()
        self.resnet = models.resnet18(pretrained=True)
        for param in self.resnet.parameters():
            param.requires_grad = False
        num_features = self.resnet.fc.in_features
        self.resnet.fc = nn.Linear(num_features, output_dim)
    def forward(self, x):
        return self.resnet(x)
np.random.seed(0)
X_train = np.random.rand(100, 3, 64, 64).astype(np.float32)
y_train = np.random.randint(0, 2, 100)
X_test = np.random.rand(30, 3, 64, 64).astype(np.float32)
y_test = np.random.randint(0, 2, 30)
train_data = TensorDataset(torch.from_numpy(X_train), torch.from_numpy(y_train))
test_data = TensorDataset(torch.from_numpy(X_test), torch.from_numpy(y_test))

train_loader = DataLoader(train_data, batch_size=16, shuffle=True)
test_loader = DataLoader(test_data, batch_size=10, shuffle=False)
model = PretrainedCNNFeatureExtractor(output_dim=2)
criterion = nn.CrossEntropyLoss()
```

Variables Terminal

7:43 PM Python 3

Safari File Edit View History Bookmarks Window Help

Tue Nov 4 7:45 PM

colab.research.google.com

clean coal technology diagrams - Google Search | SRM Student Portal | Untitled8.ipynb - Colab | 21AIC301J- Lab AI-A

Untitled8.ipynb ☆ Saving failed since 7:38 PM

File Edit View Insert Runtime Tools Help

Share A

Commands + Code + Text ▷ Run all

RAM Disk

[2]

```
train_loader = DataLoader(train_data, batch_size=16, shuffle=True)
test_loader = DataLoader(test_data, batch_size=10, shuffle=False)
model = PretrainedCNNFeatureExtractor(output_dim=2)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.resnet.fc.parameters(), lr=0.001)
num_epochs = 10
train_losses = []
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    for inputs, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
    epoch_loss = running_loss / len(train_loader.dataset)
    train_losses.append(epoch_loss)
    print(f'Epoch {epoch+1}/{num_epochs}, Loss: {epoch_loss:.4f}')
model.eval()
all_preds, all_labels = [], []
with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        all_preds.extend(preds.numpy())
        all_labels.extend(labels.numpy())
accuracy = accuracy_score(all_labels, all_preds)
f1 = f1_score(all_labels, all_preds)
recall = recall_score(all_labels, all_preds)
print(f'Accuracy: {accuracy:.4f}')
```

Variables Terminal

✓ 7:43

Safari File Edit View History Bookmarks Window Help Tue Nov 4 7:45 PM

colab.research.google.com

clean coal technology diagrams - Google Search | SRM Student Portal | Untitled8.ipynb - Colab | 21AIC301J- Lab AI-A

Untitled8.ipynb Saving failed since 7:38 PM

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

[2]

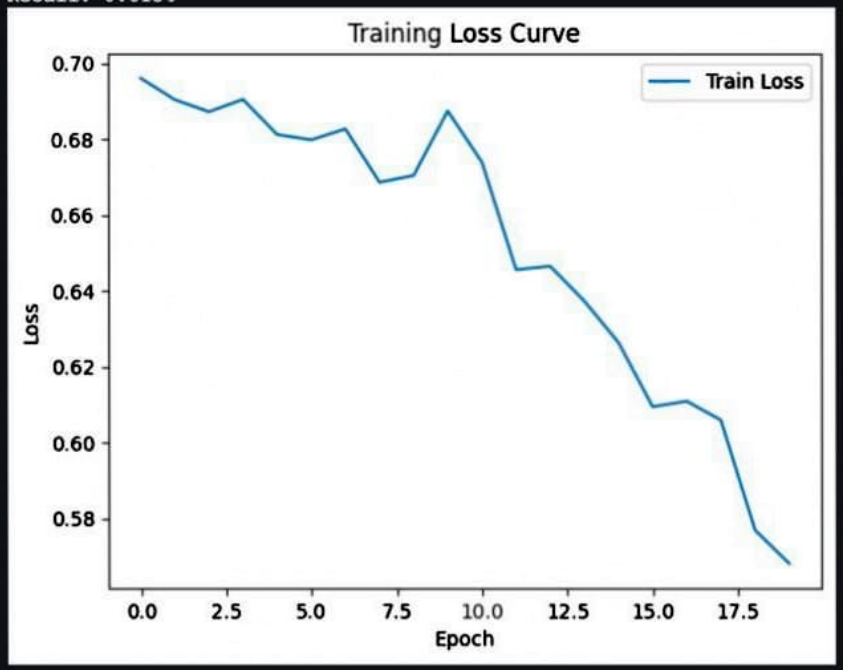
```
        _, preds = torch.max(outputs, 1)
        all_preds.extend(preds.numpy())
        all_labels.extend(labels.numpy())
accuracy = accuracy_score(all_labels, all_preds)
f1 = f1_score(all_labels, all_preds)
recall = recall_score(all_labels, all_preds)
print(f'Accuracy: {accuracy:.4f}')
print(f'F1 Score: {f1:.4f}')
print(f'Recall: {recall:.4f}')
plt.plot(train_losses, label='Train Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be re
  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 de
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|██████████| 44.7M/44.7M [00:00<00:00, 127MB/s]
Epoch 1/10, Loss: 0.7812
Epoch 2/10, Loss: 0.6875
Epoch 3/10, Loss: 0.6530
Epoch 4/10, Loss: 0.5271
Epoch 5/10, Loss: 0.5459
Epoch 6/10, Loss: 0.5682
Epoch 7/10, Loss: 0.5525
Epoch 8/10, Loss: 0.5192
Epoch 9/10, Loss: 0.4761
```

Variables Terminal

7:43

```
Epoch 18/20, Loss: 0.6059
Epoch 19/20, Loss: 0.5769
Epoch 20/20, Loss: 0.5681
Accuracy: 0.5333
F1 Score: 0.5333
Recall: 0.6154
```




```
[2]
import torch
```