Lab 14: Implement a Pre-trained CNN model as a feature extraction using transfer learning

Aim:

To use a pre-trained CNN (like VGG16 or ResNet50) as a feature extractor for a new image classification task.

Description:

Transfer learning leverages features learned from large datasets and applies them to smaller, related tables.

The convolutional base of the model is frozen and new dense layer are added for specific classification outputs.

Procedure:

1) Load a pre-trained model without its top classifier.
2) Freeze convolutional layers
3) Add a dense layer for the target dataset
4) Compile and train the new model.
5) Evaluate accuracy and visualize predictions

Pseudocode:

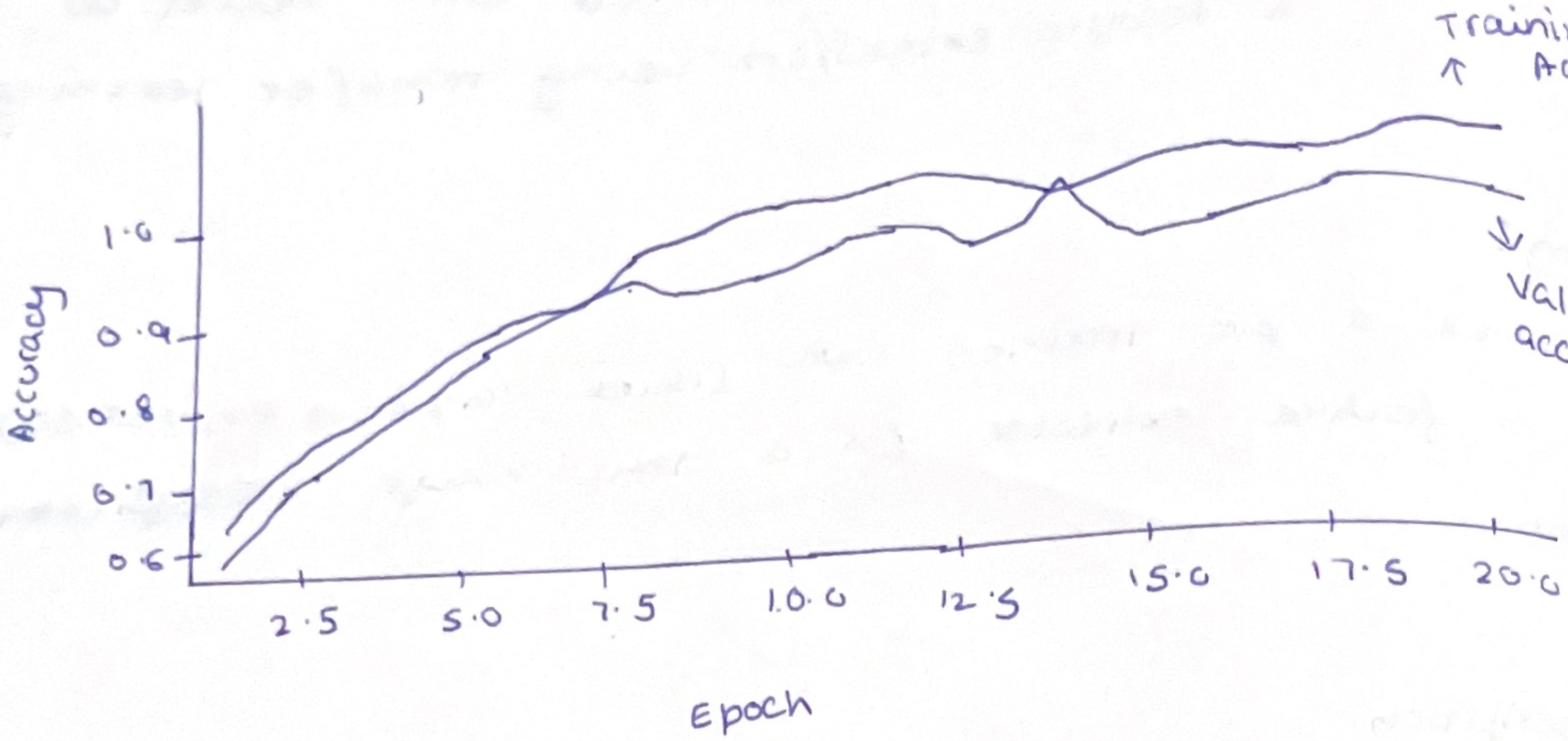
base = VGG16 (include_top: false, weights: "imagenet")
freeze base layer
Add Flatten → Dense → Output layers
compile and train on new dataset
Evaluate and predict.

Observation:

Model achieves high accuracy with fewer epochs due to reused feature maps.
Feature extraction reduces training time and data requirements

Result:

successfully used pre trained CNN as a feature extractor for a new classification model.



Epoch	Training acc	Val. Acc.
1	58·2	55·1
5	72·4	69·3
10	83·7	80·1
15	90·5	87·2
20	99·8	94·7

colab.research.google.com/drive/1MtfHfPH8LbbDoVvOYbTaPYu5eTblU3QN?usp=sharing

Week14.ipynb Changes will not be saved

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

Share Connect T4

```
import torch, torch.nn as nn, torch.optim as optim
from torchvision import models, datasets, transforms
from torch.utils.data import DataLoader
from torchinfo import summary
from time import time

torch.manual_seed(42)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print("Device:", device)

# --- Data (CIFAR-10 -> 224, ImageNet stats)
img_size = 224
batch_size = 128
num_classes = 10

weights18 = models.ResNet18_Weights.IMAGENET1K_V1
mean, std = weights18.meta.get('mean', (0.485,0.456,0.406)), weights18.meta.get('std', (0.229,0.224,0.225))

tf_train = transforms.Compose([
    transforms.Resize((img_size,img_size)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomCrop(img_size, padding=4),
    transforms.ToTensor(),
    transforms.Normalize(mean=mean, std=std)
])
tf_val = transforms.Compose([
    transforms.Resize((img_size,img_size)),
    transforms.ToTensor(),
    transforms.Normalize(mean=mean, std=std)
```

Variables Terminal

32°C Mostly cloudy

Search

ENG IN 13:55 04-11-2025

Week14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

```
            x, y = x.to(device), y.to(device)
            optimizer.zero_grad()
            logits = model(x)
            loss = criterion(logits, y)
            loss.backward()
            optimizer.step()
        vloss, vacc = evaluate()
        best_acc = max(best_acc, vacc)
        print(f"Epoch {ep}/{epochs} | val_loss={vloss:.4f} | val_acc={vacc*100:.2f}% | time={(time()-t0):.1f}s")

print(f"Done. Best Val Acc (frozen backbone): {best_acc*100:.2f}%")

# --- Optional: Fine-tune last block + head
for m in [model.layer4, model.fc]:
    for p in m.parameters(): p.requires_grad = True

optimizer = optim.AdamW(filter(lambda p: p.requires_grad, model.parameters()), lr=5e-4)
epochs_ft = 3
for ep in range(1, epochs_ft+1):
    for x, y in train_loader:
        x, y = x.to(device), y.to(device)
        optimizer.zero_grad()
        loss = criterion(model(x), y)
        loss.backward()
        optimizer.step()
    vloss, vacc = evaluate()
    print(f"[FT] Epoch {ep}/{epochs_ft} | val_loss={vloss:.4f} | val_acc={vacc*100:.2f}%")

print("Transfer learning complete.")
```

Variables Terminal

32°C
Mostly cloudy

Week14.ipynb ☆ Changes will not be saved

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all Copy to Drive

Share

Connect T4

```
print("Transfer learning complete.")
```

```
Device: cuda
100%|| 170M/170M [00:03<00:00, 48.3MB/s]
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, 157MB/s]
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
ResNet                                   [1, 10]                   --
├─Conv2d: 1-1                            [1, 64, 112, 112]         (9,408)
├─BatchNorm2d: 1-2                       [1, 64, 112, 112]         (128)
├─ReLU: 1-3                              [1, 64, 112, 112]         --
├─MaxPool2d: 1-4                         [1, 64, 56, 56]           --
├─Sequential: 1-5                        [1, 64, 56, 56]           --
│    └─BasicBlock: 2-1                   [1, 64, 56, 56]           --
│    │    └─Conv2d: 3-1                  [1, 64, 56, 56]           (36,864)
│    │    └─BatchNorm2d: 3-2             [1, 64, 56, 56]           (128)
│    │    └─ReLU: 3-3                    [1, 64, 56, 56]           --
│    │    └─Conv2d: 3-4                  [1, 64, 56, 56]           (36,864)
│    │    └─BatchNorm2d: 3-5             [1, 64, 56, 56]           (128)
│    │    └─ReLU: 3-6                    [1, 64, 56, 56]           --
│    └─BasicBlock: 2-2                   [1, 64, 56, 56]           --
│    │    └─Conv2d: 3-7                  [1, 64, 56, 56]           (36,864)
│    │    └─BatchNorm2d: 3-8             [1, 64, 56, 56]           (128)
│    │    └─ReLU: 3-9                    [1, 64, 56, 56]           --
│    │    └─Conv2d: 3-10                 [1, 64, 56, 56]           (36,864)
│    │    └─BatchNorm2d: 3-11            [1, 64, 56, 56]           (128)
│    │    └─ReLU: 3-12                   [1, 64, 56, 56]           --
├─Sequential: 1-6                        [1, 128, 28, 28]          --
│    └─BasicBlock: 2-3                   [1, 128, 28, 28]          --
│    │    └─Conv2d: 3-13                 [1, 128, 28, 28]          (73,728)
```

```
    └─BatchNorm2d: 3-47                [1, 512, 7, 7]            (1,024)
    └─ReLU: 3-48                       [1, 512, 7, 7]            --
    └─Conv2d: 3-49                     [1, 512, 7, 7]            (2,359,296)
    └─BatchNorm2d: 3-50                [1, 512, 7, 7]            (1,024)
    └─ReLU: 3-51                       [1, 512, 7, 7]            --
├─AdaptiveAvgPool2d: 1-9               [1, 512, 1, 1]            --
├─Sequential: 1-10                     [1, 10]                   --
    └─Dropout: 2-9                     [1, 512]                  --
    └─Linear: 2-10                     [1, 10]                   5,130
==========================================================================================
Total params: 11,181,642
Trainable params: 5,130
Non-trainable params: 11,176,512
Total mult-adds (Units.GIGABYTES): 1.81
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 39.74
Params size (MB): 44.73
Estimated Total Size (MB): 85.07
==========================================================================================
Epoch 1/5 | val_loss=0.7023 | val_acc=76.37% | time=105.0s
Epoch 2/5 | val_loss=0.6327 | val_acc=78.72% | time=104.4s
Epoch 3/5 | val_loss=0.6006 | val_acc=79.65% | time=107.7s
Epoch 4/5 | val_loss=0.5957 | val_acc=79.47% | time=103.7s
Epoch 5/5 | val_loss=0.5881 | val_acc=79.87% | time=106.7s
Done. Best Val Acc (frozen backbone): 79.87%
[FT] Epoch 1/3 | val_loss=0.3267 | val_acc=88.60%
[FT] Epoch 2/3 | val_loss=0.2760 | val_acc=91.04%
[FT] Epoch 3/3 | val_loss=0.2866 | val_acc=90.60%
Transfer learning complete.
```