Transfer Learning

November 9, 2025

[1]:
```
from torch.utils.data import Dataset, DataLoader
from torchvision.transforms import ToTensor,transforms,v2
from pathlib import Path
from PIL import Image
from torch import nn
import torch
import random
import matplotlib.pyplot as plt
import torchvision
from torchvision import datasets
from torchinfo import summary
```

[2]:
```
device = "cuda" if torch.cuda.is_available() else "cpu"
```

[3]:
```
def accuracy_fn(y_pred, y_true):
    correct_num = (y_pred == y_true).sum()
    acc = correct_num / len(y_true) * 100
    return acc
```

[4]:
```
def train_step(dataloader, model, cost_fn, optimizer, accuracy_fn, device):
    train_cost = 0
    train_acc = 0
    for batch, (x, y) in enumerate(dataloader):

        x = x.to(device)
        y = y.to(device)

        model.train()
        y_pred =  model(x)
        cost = cost_fn(y_pred,y)
        train_acc += accuracy_fn(y_pred.argmax(dim=1), y)

        train_cost += cost

        optimizer.zero_grad()
        cost.backward()
        optimizer.step()
```

```
    train_cost /= len(dataloader)
    train_acc /= len(dataloader)

    return train_cost, train_acc

def test_step(dataloader, model, cost_fn, accuracy_fn, device):
    test_cost = 0
    test_acc = 0
    model.eval()
    with torch.inference_mode():
        for x, y in dataloader:
            x = x.to(device)
            y = y.to(device)
            test_pred =  model(x)

            test_cost += cost_fn(test_pred,y)
            test_acc += accuracy_fn(test_pred.argmax(dim=1), y)

        test_cost /= len(dataloader)
        test_acc /= len(dataloader)

        return test_cost, test_acc
```

0.1 Model

[5]:
```
weights = torchvision.models.EfficientNet_B1_Weights.DEFAULT
model = torchvision.models.efficientnet_b1(weights = weights)
model.to(device)
```

[5]:
```
EfficientNet(
  (features): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=32, bias=False)
            (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
```

```
          )
          (1): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(32, 8, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(8, 32, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.0, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=16, bias=False)
            (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(16, 4, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(4, 16, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(16, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.008695652173913044, mode=row)
      )
    )
    (2): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(16, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), groups=96, bias=False)
            (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(96, 4, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(4, 96, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(96, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.017391304347826087, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=144, bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
```

```
          (3): Conv2dNormActivation(
            (0): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.026086956521739136, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=144, bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.034782608695652174, mode=row)
      )
    )
    (3): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
```

```
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(144, 144, kernel_size=(5, 5), stride=(2, 2), padding=(2,
2), groups=144, bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(144, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(40, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.043478260869565216, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(240, 240, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=240, bias=False)
            (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(240, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
            (1): BatchNorm2d(40, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.05217391304347827, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(240, 240, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=240, bias=False)
            (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(240, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(40, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.06086956521739131, mode=row)
      )
    )
    (4): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
```

```
            (0): Conv2d(240, 240, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), groups=240, bias=False)
            (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(240, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.06956521739130435, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
          )
        )
        (stochastic_depth): StochasticDepth(p=0.0782608695652174, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.08695652173913043, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.09565217391304348, mode=row)
      )
    )
    (5): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(480, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
```

```
        (stochastic_depth): StochasticDepth(p=0.10434782608695654, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=672, bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.11304347826086956, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=672, bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
```

```
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.12173913043478261, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=672, bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.13043478260869565, mode=row)
      )
    )
    (6): Sequential(
```

```
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(2, 2), padding=(2,
2), groups=672, bias=False)
            (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(672, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.1391304347826087, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
```

```
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.14782608695652175, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.1565217391304348, mode=row)
      )
```

```
      (3): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.16521739130434784, mode=row)
      )
      (4): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
```

```
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.17391304347826086, mode=row)
      )
    )
    (7): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(1152, 1152, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=1152, bias=False)
            (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(1152, 320, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
          )
        )
        (stochastic_depth): StochasticDepth(p=0.1826086956521739, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(320, 1920, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1920, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(1920, 1920, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=1920, bias=False)
            (1): BatchNorm2d(1920, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1920, 80, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(80, 1920, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(1920, 320, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.19130434782608696, mode=row)
      )
    )
    (8): Conv2dNormActivation(
      (0): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(1280, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=1)
  (classifier): Sequential(
    (0): Dropout(p=0.2, inplace=True)
```

```
    (1): Linear(in_features=1280, out_features=1000, bias=True)
  )
)
```

[6]:
```
weights.transforms()
```

[6]:
```
ImageClassification(
    crop_size=[240]
    resize_size=[255]
    mean=[0.485, 0.456, 0.406]
    std=[0.229, 0.224, 0.225]
    interpolation=InterpolationMode.BILINEAR
)
```

[7]:
```
efficientnet_b1_transforms = weights.transforms()
```

[8]:
```
train_dataset = datasets.CIFAR10(
    root = "image",
    train = True,
    download = True,
    transform = efficientnet_b1_transforms
)

test_dataset = datasets.CIFAR10(
    root = "image",
    train = False,
    download = True,
    transform = efficientnet_b1_transforms
)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

[9]:
```
BATCH_SIZE = 32

train_dataloader = DataLoader(train_dataset,batch_size = BATCH_SIZE,shuffle =␣
 ↪True)
test_dataloader = DataLoader(test_dataset,batch_size = BATCH_SIZE,shuffle =␣
 ↪False)
```

[10]:
```
len(train_dataloader)
```

[10]:
```
1563
```

[11]:
```
summary(model=model,
        input_size=(32, 3, 128, 128),
        col_names=["input_size","output_size","num_params","trainable"],
        row_settings=["var_names"])
```

[11]:
```
==========================================================================================
==========================================================================================
Layer (type (var_name))                                       Input Shape
Output Shape              Param #                     Trainable
==========================================================================================
==========================================================================================
EfficientNet (EfficientNet)                                   [32, 3, 128, 128]
[32, 1000]                --                          True
 Sequential (features)                                        [32, 3, 128, 128]
[32, 1280, 4, 4]          --                          True
     Conv2dNormActivation (0)                                 [32, 3, 128, 128]
[32, 32, 64, 64]          --                          True
          Conv2d (0)                                          [32, 3, 128, 128]
[32, 32, 64, 64]          864                         True
          BatchNorm2d (1)                                     [32, 32, 64, 64]
[32, 32, 64, 64]          64                          True
          SiLU (2)                                            [32, 32, 64, 64]
[32, 32, 64, 64]          --                          --
     Sequential (1)                                           [32, 32, 64, 64]
[32, 16, 64, 64]          --                          True
          MBConv (0)                                          [32, 32, 64, 64]
[32, 16, 64, 64]          1,448                       True
          MBConv (1)                                          [32, 16, 64, 64]
[32, 16, 64, 64]          612                         True
     Sequential (2)                                           [32, 16, 64, 64]
[32, 24, 32, 32]          --                          True
          MBConv (0)                                          [32, 16, 64, 64]
[32, 24, 32, 32]          6,004                       True
          MBConv (1)                                          [32, 24, 32, 32]
[32, 24, 32, 32]          10,710                      True
          MBConv (2)                                          [32, 24, 32, 32]
[32, 24, 32, 32]          10,710                      True
     Sequential (3)                                           [32, 24, 32, 32]
[32, 40, 16, 16]          --                          True
          MBConv (0)                                          [32, 24, 32, 32]
[32, 40, 16, 16]          15,350                      True
          MBConv (1)                                          [32, 40, 16, 16]
[32, 40, 16, 16]          31,290                      True
          MBConv (2)                                          [32, 40, 16, 16]
[32, 40, 16, 16]          31,290                      True
     Sequential (4)                                           [32, 40, 16, 16]
[32, 80, 8, 8]            --                          True
          MBConv (0)                                          [32, 40, 16, 16]
[32, 80, 8, 8]            37,130                      True
          MBConv (1)                                          [32, 80, 8, 8]
[32, 80, 8, 8]            102,900                     True
          MBConv (2)                                          [32, 80, 8, 8]
```

```
[32, 80, 8, 8]            102,900                  True
          MBConv (3)                                       [32, 80, 8, 8]
[32, 80, 8, 8]            102,900                  True
     Sequential (5)                                        [32, 80, 8, 8]
[32, 112, 8, 8]           --                       True
          MBConv (0)                                       [32, 80, 8, 8]
[32, 112, 8, 8]           126,004                  True
          MBConv (1)                                       [32, 112, 8, 8]
[32, 112, 8, 8]           208,572                  True
          MBConv (2)                                       [32, 112, 8, 8]
[32, 112, 8, 8]           208,572                  True
          MBConv (3)                                       [32, 112, 8, 8]
[32, 112, 8, 8]           208,572                  True
     Sequential (6)                                        [32, 112, 8, 8]
[32, 192, 4, 4]           --                       True
          MBConv (0)                                       [32, 112, 8, 8]
[32, 192, 4, 4]           262,492                  True
          MBConv (1)                                       [32, 192, 4, 4]
[32, 192, 4, 4]           587,952                  True
          MBConv (2)                                       [32, 192, 4, 4]
[32, 192, 4, 4]           587,952                  True
          MBConv (3)                                       [32, 192, 4, 4]
[32, 192, 4, 4]           587,952                  True
          MBConv (4)                                       [32, 192, 4, 4]
[32, 192, 4, 4]           587,952                  True
     Sequential (7)                                        [32, 192, 4, 4]
[32, 320, 4, 4]           --                       True
          MBConv (0)                                       [32, 192, 4, 4]
[32, 320, 4, 4]           717,232                  True
          MBConv (1)                                       [32, 320, 4, 4]
[32, 320, 4, 4]           1,563,600                True
     Conv2dNormActivation (8)                              [32, 320, 4, 4]
[32, 1280, 4, 4]          --                       True
          Conv2d (0)                                       [32, 320, 4, 4]
[32, 1280, 4, 4]          409,600                  True
          BatchNorm2d (1)                                  [32, 1280, 4, 4]
[32, 1280, 4, 4]          2,560                    True
          SiLU (2)                                         [32, 1280, 4, 4]
[32, 1280, 4, 4]          --                       --
 AdaptiveAvgPool2d (avgpool)                               [32, 1280, 4, 4]
[32, 1280, 1, 1]          --                       --
 Sequential (classifier)                                   [32, 1280]
[32, 1000]                --                       True
     Dropout (0)                                           [32, 1280]
[32, 1280]                --                       --
     Linear (1)                                            [32, 1280]
[32, 1000]                1,281,000                True
```

```
==========================================================================================
==========================================================================================
Total params: 7,794,184
Trainable params: 7,794,184
Non-trainable params: 0
Total mult-adds (G): 6.01
==========================================================================================
==========================================================================================
Input size (MB): 6.29
Forward/backward pass size (MB): 1565.47
Params size (MB): 31.18
Estimated Total Size (MB): 1602.94
==========================================================================================
==========================================================================================
```

[12]:
```
# This is transfer learning (feature extraction mode):

# Freeze backbone (remain the pretrained features)

# Train only the final classifier to map features → the labels
```

[13]:
```
model.classifier[1] = nn.Linear(in_features=1280,out_features=10,bias=True).
 ↪to(device)
```

[14]:
```
for param in model.features.parameters():
    param.requires_grad=False
    # print(param)
```

[15]:
```
from torch.optim.lr_scheduler import StepLR

cost_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(params = model.parameters(), lr = 0.01,␣
 ↪weight_decay=1e-4)
scheduler = StepLR(optimizer, step_size=5, gamma=0.5)
```

[16]:
```
train_losses, test_losses = [], []
train_accuracies, test_accuracies = [], []

epochs = 10
for epoch in range(epochs):
    print(f"Epoch: {epoch} \n -----")

    train_cost, train_acc = train_step(train_dataloader, model, cost_fn,␣
 ↪optimizer, accuracy_fn, device)

    test_cost, test_acc = test_step(test_dataloader, model, cost_fn,␣
 ↪accuracy_fn, device)
```

```
    scheduler.step()

    train_losses.append(train_cost.detach().cpu().item())
    test_losses.append(test_cost.detach().cpu().item())
    train_accuracies.append(train_acc.detach().cpu().item())
    test_accuracies.append(test_acc.detach().cpu().item())

    print(f"\nTrain Cost: {train_cost:.4f}, {train_acc:.2f}")

    print(f"Test Cost: {test_cost:.4f}, {test_acc:.2f} \n")
```

```
Epoch: 0
 -----

Train Cost: 0.7885, 72.88
Test Cost: 0.6055, 79.57

Epoch: 1
 -----

Train Cost: 0.7516, 74.62
Test Cost: 0.5754, 80.43

Epoch: 2
 -----

Train Cost: 0.7471, 74.76
Test Cost: 0.5979, 79.51

Epoch: 3
 -----

Train Cost: 0.7477, 74.67
Test Cost: 0.5694, 80.54

Epoch: 4
 -----

Train Cost: 0.7473, 74.71
Test Cost: 0.5856, 79.88

Epoch: 5
 -----

Train Cost: 0.6918, 76.12
Test Cost: 0.5443, 81.19
```

```
Epoch: 6
 -----

Train Cost: 0.6779, 76.67
Test Cost: 0.5303, 81.83

Epoch: 7
 -----

Train Cost: 0.6760, 76.65
Test Cost: 0.5278, 81.66

Epoch: 8
 -----

Train Cost: 0.6716, 76.79
Test Cost: 0.5345, 81.70

Epoch: 9
 -----

Train Cost: 0.6718, 76.54
Test Cost: 0.5324, 81.78
```

[17]:
```
# Retraining the last layer is not enough

# Because the model needs to:

# Adapt deeper features

# Ignore irrelevant ImageNet patterns (dog fur, cat eyes, textures, etc.)
```

[18]:
```
for param in model.features[-2:].parameters():
    param.requires_grad = True
```

[19]:
```
optimizer = torch.optim.Adam(model.parameters(), lr=1e-5)
```

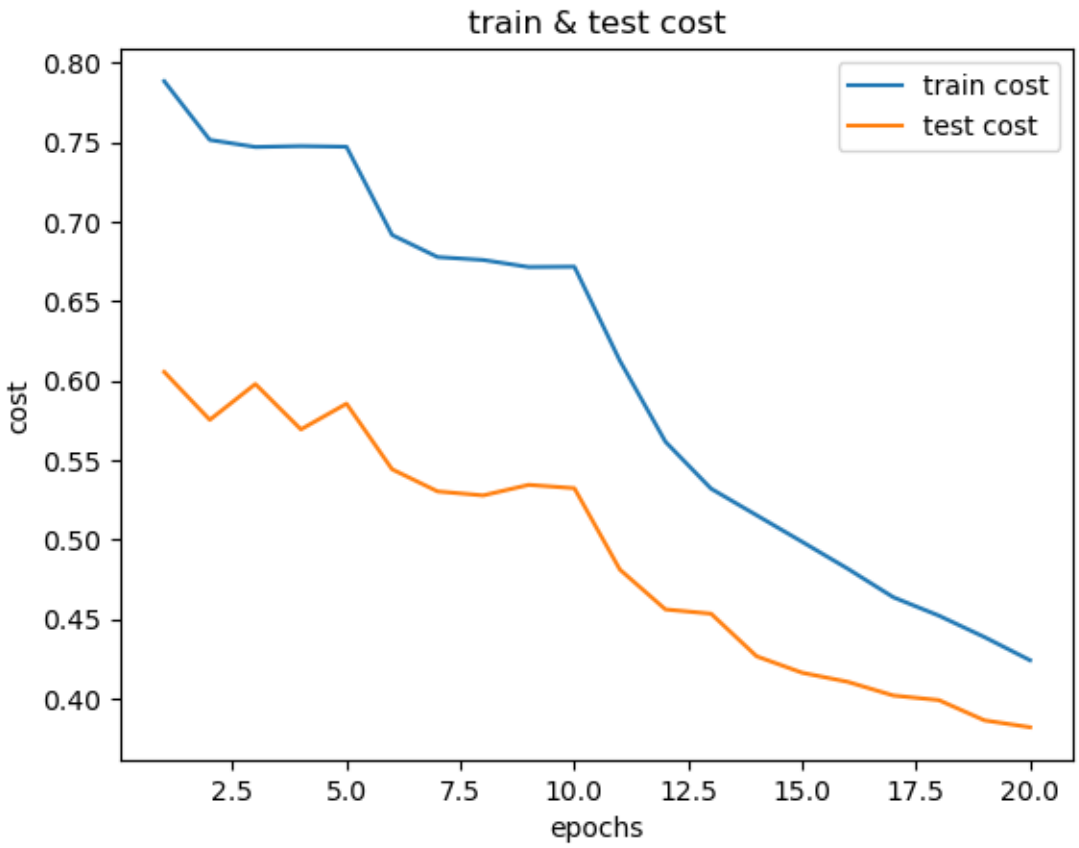
[20]:
```
print("\nPhase 2: Fine-tuning the backbone")

epochs = 10
for epoch in range(epochs):
    print(f"Epoch: {epoch} \n -----")

    train_cost, train_acc = train_step(train_dataloader, model, cost_fn,␣
 ↪optimizer, accuracy_fn, device)
```

```
    test_cost, test_acc = test_step(test_dataloader, model, cost_fn,␣
↪accuracy_fn, device)

    scheduler.step()

    train_losses.append(train_cost.detach().cpu().item())
    test_losses.append(test_cost.detach().cpu().item())
    train_accuracies.append(train_acc.detach().cpu().item())
    test_accuracies.append(test_acc.detach().cpu().item())

    print(f"\nTrain Cost: {train_cost:.4f}, {train_acc:.2f}")

    print(f"Test Cost: {test_cost:.4f}, {test_acc:.2f} \n")
```

```
Phase 2: Fine-tuning the backbone
Epoch: 0
 -----

Train Cost: 0.6126, 78.72
Test Cost: 0.4811, 83.61

Epoch: 1
 -----

Train Cost: 0.5616, 80.41
Test Cost: 0.4561, 84.44

Epoch: 2
 -----

Train Cost: 0.5321, 81.31
Test Cost: 0.4534, 84.72

Epoch: 3
 -----

Train Cost: 0.5153, 81.93
Test Cost: 0.4266, 85.50

Epoch: 4
 -----

Train Cost: 0.4986, 82.40
Test Cost: 0.4162, 85.88

Epoch: 5
 -----
```

```
Train Cost: 0.4817, 83.04
Test Cost: 0.4105, 86.09

Epoch: 6
 -----

Train Cost: 0.4637, 83.86
Test Cost: 0.4019, 86.45

Epoch: 7
 -----

Train Cost: 0.4521, 84.11
Test Cost: 0.3990, 86.45

Epoch: 8
 -----

Train Cost: 0.4387, 84.58
Test Cost: 0.3862, 87.01

Epoch: 9
 -----

Train Cost: 0.4241, 85.10
Test Cost: 0.3819, 87.00
```

[21]:
```
plt.plot(range(1,len(train_losses) + 1), train_losses, label = "train cost")
plt.plot(range(1,len(test_losses) + 1), test_losses, label = "test cost")
plt.title("train & test cost")
plt.xlabel("epochs")
plt.ylabel("cost")
plt.legend()
plt.show()
```



0.2 save model

[25]:
```
# torch.save(model.state_dict(), "efficientnet_cifar10.pth")
```