```
import torch
import torch.nn as nn
from torchvision.models._api import WeightsEnum

from torch.hub import load_state_dict_from_url
from pathlib import Path
from os import listdir
import numpy as np
import torchvision
import torch.optim as optim
import torch.nn.functional as F
from torch.utils.data import DataLoader, TensorDataset

import matplotlib.pyplot as plt
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from torchvision.models._api import WeightsEnum
from torch.hub import load_state_dict_from_url

from torchvision.models._api import WeightsEnum
from torch.hub import load_state_dict_from_url

def get_state_dict(self, *args, **kwargs):
    kwargs.pop("check_hash")
    return load_state_dict_from_url(self.url, *args, **kwargs)
WeightsEnum.get_state_dict = get_state_dict

model =
torchvision.models.efficientnet_b0(weights=torchvision.models.Efficien
tNet_B0_Weights.IMAGENET1K_V1)
```

```
Downloading:
"https://download.pytorch.org/models/efficientnet_b0_rwightman-
7f5810bc.pth" to
/root/.cache/torch/hub/checkpoints/efficientnet_b0_rwightman-
7f5810bc.pth
100%|██████████| 20.5M/20.5M [00:00<00:00, 83.3MB/s]
```

```
def get_state_dict(self, *args, **kwargs):
    kwargs.pop("check_hash")
    return load_state_dict_from_url(self.url, *args, **kwargs)
WeightsEnum.get_state_dict = get_state_dict

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

print(device)
```

```
cuda
```

```
X_train = torch.load('/kaggle/input/artist/tensor_stack.pt')

X_test = torch.load('/kaggle/input/artist/val_tensor_stack.pt')
```

```
y_train = torch.load('/kaggle/input/artist/artist_training_tensor.pt')

y_test = torch.load('/kaggle/input/artist/artist_val_tensor.pt')

first_tensor_shape = X_train[0].shape if isinstance(X_train, list)
else X_train.shape

print(first_tensor_shape)
```

```
torch.Size([12000, 3, 224, 224])
```

```
channels = first_tensor_shape[1]
height = first_tensor_shape[2]
width = first_tensor_shape[3]

input_shape = (32, channels, height, width)

for param in model.parameters():
    param.requires_grad = False

total_params = sum(p.numel() for p in model.parameters())
print("Total parameters in the model:", total_params)
```

```
Total parameters in the model: 5288548
```

```
num_fc_layers = sum(isinstance(module, nn.Linear) for module in
model.modules())
print("Number of fully connected layers in the model:", num_fc_layers)
```

```
Number of fully connected layers in the model: 1
```

```
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset

train_dataset = TensorDataset(X_train, y_train)

val_dataset = TensorDataset(X_test, y_test)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32)

def get_state_dict(self, *args, **kwargs):
    kwargs.pop("check_hash")
    return load_state_dict_from_url(self.url, *args, **kwargs)
WeightsEnum.get_state_dict = get_state_dict

model =
torchvision.models.efficientnet_b0(weights=torchvision.models.Efficien
tNet_B0_Weights.IMAGENET1K_V1)
print(model)
```

```
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)
      )
    )
    (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=0.0125, 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=0.025, 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=0.037500000000000006,
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=0.05, 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=0.0625, 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=0.07500000000000001,
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=0.08750000000000001,
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=0.1, 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=0.1125, 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=0.125, 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=0.1375, 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=0.15000000000000002,
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=0.1625, 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=0.17500000000000002,
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=0.1875, 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)
  )
)

model.classifier[1] = nn.Sequential(
    nn.Linear(in_features=1280, out_features=1024, bias=True),
    nn.ReLU(inplace = True),
    nn.BatchNorm1d(1024),
    nn.Dropout(0.2),
    nn.Linear(in_features=1024, out_features=23, bias=True)
)
model.to(device)
```

```
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)
      )
```

```
    )
    (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=0.0125, 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=0.025, 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=0.037500000000000006,
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=0.05, 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=0.0625, 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=0.07500000000000001,
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=0.08750000000000001,
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=0.1, 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=0.1125, 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=0.125, 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=0.1375, 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=0.15000000000000002,
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=0.1625, 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=0.17500000000000002,
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=0.1875, 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): Sequential(
      (0): Linear(in_features=1280, out_features=1024, bias=True)
      (1): ReLU(inplace=True)
      (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (3): Dropout(p=0.2, inplace=False)
      (4): Linear(in_features=1024, out_features=23, bias=True)
    )
  )
)

total_params = sum(p.numel() for p in model.parameters())
print(f"Total number of parameters in resnet: {total_params}")

Total number of parameters in resnet: 5344915

batch_size = 32
learning_rate = 0.001

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

def train(model, train_loader, optimizer, criterion):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        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()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_loss = running_loss / len(train_loader)
    train_accuracy = 100 * correct / total
    return train_loss, train_accuracy

import torch
import matplotlib.pyplot as plt

def test_plot(model, test_loader, criterion):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    correct_labels = []
    misclassified_labels = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
            for i in range(len(labels)):
                if predicted[i] == labels[i]:
                    correct_labels.append(labels[i].item())
                else:
                    misclassified_labels.append((labels[i].item(),
predicted[i].item()))

    test_loss = running_loss / len(test_loader)
    test_accuracy = 100 * correct / total

    plot_labels(correct_labels, misclassified_labels)

    return test_loss, test_accuracy


def test(model, test_loader, criterion):
```

```
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    test_loss = running_loss / len(test_loader)
    test_accuracy = 100 * correct / total
    return test_loss, test_accuracy

def plot_labels(correct_labels, misclassified_labels):
    all_labels = correct_labels + [label[0] for label in
misclassified_labels]
    all_predictions = correct_labels + [label[1] for label in
misclassified_labels]

    plt.figure(figsize=(10, 5))
    plt.hist([all_labels, all_predictions],
bins=range(min(all_labels), max(all_labels) + 2), alpha=0.5,
label=['Correct', 'Misclassified'])
    plt.xticks(range(min(all_labels), max(all_labels) + 1))
    plt.xlabel('Class')
    plt.ylabel('Count')
    plt.title('Correct vs Misclassified Classes')
    plt.legend()
    plt.show()

num_epochs = 100
train_losses = []
train_accuracies = []
test_losses = []
test_accuracies = []
epochs = []

for epoch in range(num_epochs):
    train_loss, train_accuracy = train(model, train_loader, optimizer,
criterion)
    test_loss, test_accuracy = test(model, val_loader, criterion)
    train_losses.append(train_loss)
    train_accuracies.append(train_accuracy)
```

```
    test_losses.append(test_loss)
    test_accuracies.append(test_accuracy)
    epochs.append(epoch + 1)
    print(f"Epoch [{epoch + 1}/{num_epochs}], Train Loss:
{train_loss:.4f}, Train Accuracy: {train_accuracy:.2f}%, Test Loss:
{test_loss:.4f}, Test Accuracy: {test_accuracy:.2f}%")
train_loss, train_accuracy = train(model, train_loader, optimizer,
criterion)
test_loss, test_accuracy = test_plot(model, val_loader, criterion)
print(f"Train Loss: {train_loss:.4f}, Train Accuracy:
{train_accuracy:.2f}%, Test Loss: {test_loss:.4f}, Test Accuracy:
{test_accuracy:.2f}%")
```

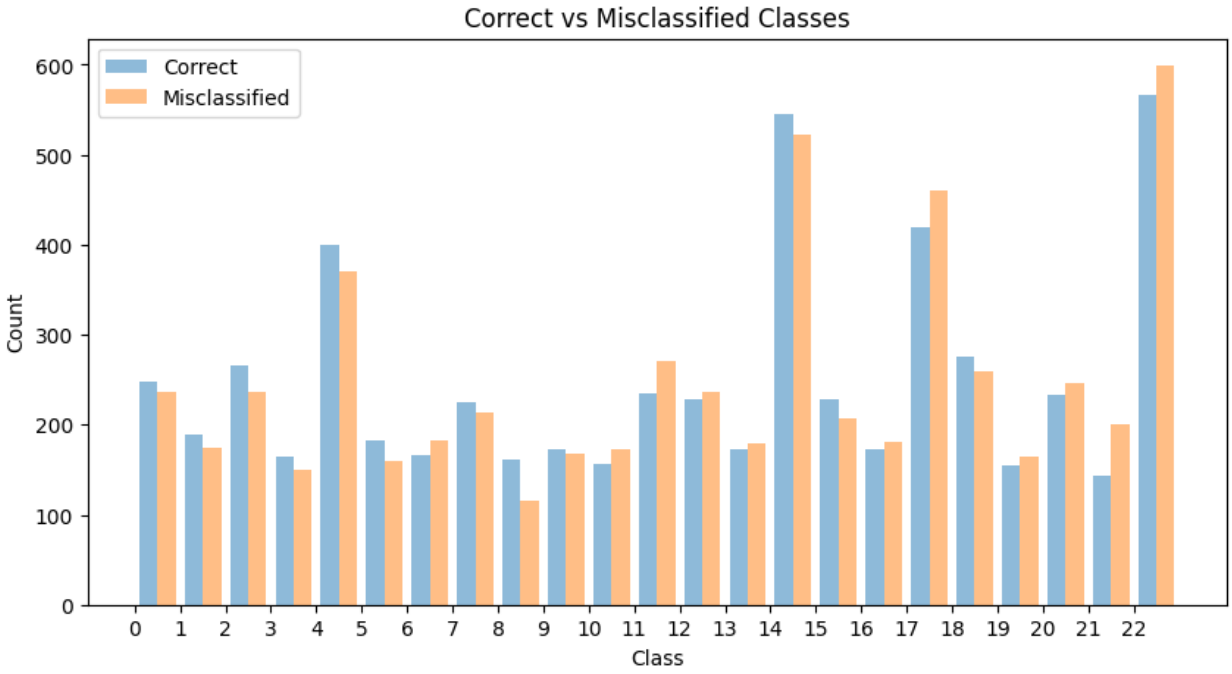
```
Epoch [1/100], Train Loss: 1.3738, Train Accuracy: 61.05%, Test Loss:
0.9438, Test Accuracy: 73.31%
Epoch [2/100], Train Loss: 0.8331, Train Accuracy: 75.38%, Test Loss:
0.8091, Test Accuracy: 76.48%
Epoch [3/100], Train Loss: 0.6222, Train Accuracy: 81.56%, Test Loss:
0.7983, Test Accuracy: 77.78%
Epoch [4/100], Train Loss: 0.5044, Train Accuracy: 84.68%, Test Loss:
0.8999, Test Accuracy: 76.85%
Epoch [5/100], Train Loss: 0.4335, Train Accuracy: 86.86%, Test Loss:
0.7569, Test Accuracy: 79.69%
Epoch [6/100], Train Loss: 0.3806, Train Accuracy: 88.15%, Test Loss:
0.7997, Test Accuracy: 79.13%
Epoch [7/100], Train Loss: 0.3324, Train Accuracy: 89.62%, Test Loss:
0.8657, Test Accuracy: 78.92%
Epoch [8/100], Train Loss: 0.2784, Train Accuracy: 91.49%, Test Loss:
0.8699, Test Accuracy: 78.78%
Epoch [9/100], Train Loss: 0.2636, Train Accuracy: 91.81%, Test Loss:
0.8242, Test Accuracy: 80.16%
Epoch [10/100], Train Loss: 0.2453, Train Accuracy: 92.45%, Test Loss:
0.9546, Test Accuracy: 78.76%
Epoch [11/100], Train Loss: 0.2237, Train Accuracy: 93.08%, Test Loss:
0.8219, Test Accuracy: 80.42%
Epoch [12/100], Train Loss: 0.2182, Train Accuracy: 93.37%, Test Loss:
0.8539, Test Accuracy: 80.02%
Epoch [13/100], Train Loss: 0.2160, Train Accuracy: 93.47%, Test Loss:
0.8238, Test Accuracy: 80.83%
Epoch [14/100], Train Loss: 0.1773, Train Accuracy: 94.42%, Test Loss:
0.9253, Test Accuracy: 79.78%
Epoch [15/100], Train Loss: 0.1719, Train Accuracy: 94.48%, Test Loss:
0.8781, Test Accuracy: 80.28%
Epoch [16/100], Train Loss: 0.1870, Train Accuracy: 94.33%, Test Loss:
0.9329, Test Accuracy: 79.25%
Epoch [17/100], Train Loss: 0.2272, Train Accuracy: 93.11%, Test Loss:
0.7823, Test Accuracy: 81.84%
Epoch [18/100], Train Loss: 0.1318, Train Accuracy: 95.86%, Test Loss:
0.9235, Test Accuracy: 80.34%
Epoch [19/100], Train Loss: 0.1570, Train Accuracy: 95.04%, Test Loss:
```

```
0.8409, Test Accuracy: 81.21%
Epoch [20/100], Train Loss: 0.1398, Train Accuracy: 95.67%, Test Loss:
1.0477, Test Accuracy: 78.69%
Epoch [21/100], Train Loss: 0.1925, Train Accuracy: 94.20%, Test Loss:
1.0017, Test Accuracy: 78.37%
Epoch [22/100], Train Loss: 0.1468, Train Accuracy: 95.52%, Test Loss:
0.9650, Test Accuracy: 80.00%
Epoch [23/100], Train Loss: 0.1485, Train Accuracy: 95.38%, Test Loss:
0.8351, Test Accuracy: 81.44%
Epoch [24/100], Train Loss: 0.1193, Train Accuracy: 96.28%, Test Loss:
0.9194, Test Accuracy: 80.34%
Epoch [25/100], Train Loss: 0.0981, Train Accuracy: 96.86%, Test Loss:
0.8833, Test Accuracy: 81.34%
Epoch [26/100], Train Loss: 0.1159, Train Accuracy: 96.55%, Test Loss:
1.0353, Test Accuracy: 78.60%
Epoch [27/100], Train Loss: 0.1438, Train Accuracy: 95.33%, Test Loss:
0.9280, Test Accuracy: 81.46%
Epoch [28/100], Train Loss: 0.1213, Train Accuracy: 96.13%, Test Loss:
0.9385, Test Accuracy: 80.27%
Epoch [29/100], Train Loss: 0.1000, Train Accuracy: 96.91%, Test Loss:
0.8733, Test Accuracy: 80.81%
Epoch [30/100], Train Loss: 0.0893, Train Accuracy: 97.10%, Test Loss:
0.9717, Test Accuracy: 81.02%
Epoch [31/100], Train Loss: 0.1130, Train Accuracy: 96.57%, Test Loss:
0.9885, Test Accuracy: 81.49%
Epoch [32/100], Train Loss: 0.1126, Train Accuracy: 96.48%, Test Loss:
0.9206, Test Accuracy: 81.28%
Epoch [33/100], Train Loss: 0.0936, Train Accuracy: 97.05%, Test Loss:
1.0040, Test Accuracy: 79.65%
Epoch [34/100], Train Loss: 0.0893, Train Accuracy: 97.21%, Test Loss:
0.9342, Test Accuracy: 80.48%
Epoch [35/100], Train Loss: 0.1220, Train Accuracy: 96.28%, Test Loss:
0.9775, Test Accuracy: 79.79%
Epoch [36/100], Train Loss: 0.0946, Train Accuracy: 97.03%, Test Loss:
0.8739, Test Accuracy: 81.55%
Epoch [37/100], Train Loss: 0.0984, Train Accuracy: 97.03%, Test Loss:
0.9090, Test Accuracy: 81.46%
Epoch [38/100], Train Loss: 0.1242, Train Accuracy: 96.28%, Test Loss:
1.0023, Test Accuracy: 80.30%
Epoch [39/100], Train Loss: 0.0780, Train Accuracy: 97.48%, Test Loss:
0.8729, Test Accuracy: 82.09%
Epoch [40/100], Train Loss: 0.0752, Train Accuracy: 97.52%, Test Loss:
0.9319, Test Accuracy: 82.05%
Epoch [41/100], Train Loss: 0.1006, Train Accuracy: 96.99%, Test Loss:
0.9439, Test Accuracy: 81.28%
Epoch [42/100], Train Loss: 0.0676, Train Accuracy: 97.93%, Test Loss:
0.9653, Test Accuracy: 81.32%
Epoch [43/100], Train Loss: 0.0791, Train Accuracy: 97.54%, Test Loss:
1.0493, Test Accuracy: 79.69%
```

```
Epoch [44/100], Train Loss: 0.1042, Train Accuracy: 96.89%, Test Loss:
1.0030, Test Accuracy: 80.30%
Epoch [45/100], Train Loss: 0.0775, Train Accuracy: 97.76%, Test Loss:
0.8894, Test Accuracy: 82.16%
Epoch [46/100], Train Loss: 0.0563, Train Accuracy: 98.22%, Test Loss:
1.0250, Test Accuracy: 80.20%
Epoch [47/100], Train Loss: 0.0965, Train Accuracy: 96.97%, Test Loss:
1.0413, Test Accuracy: 79.67%
Epoch [48/100], Train Loss: 0.0840, Train Accuracy: 97.34%, Test Loss:
0.9457, Test Accuracy: 81.13%
Epoch [49/100], Train Loss: 0.0514, Train Accuracy: 98.33%, Test Loss:
0.9174, Test Accuracy: 81.67%
Epoch [50/100], Train Loss: 0.0560, Train Accuracy: 98.07%, Test Loss:
0.9661, Test Accuracy: 80.65%
Epoch [51/100], Train Loss: 0.0773, Train Accuracy: 97.64%, Test Loss:
1.1563, Test Accuracy: 79.04%
Epoch [52/100], Train Loss: 0.0739, Train Accuracy: 97.65%, Test Loss:
1.0361, Test Accuracy: 80.56%
Epoch [53/100], Train Loss: 0.0737, Train Accuracy: 97.58%, Test Loss:
0.9867, Test Accuracy: 81.49%
Epoch [54/100], Train Loss: 0.0556, Train Accuracy: 98.38%, Test Loss:
0.9683, Test Accuracy: 82.02%
Epoch [55/100], Train Loss: 0.0757, Train Accuracy: 97.71%, Test Loss:
0.9619, Test Accuracy: 82.11%
Epoch [56/100], Train Loss: 0.0720, Train Accuracy: 97.87%, Test Loss:
1.0075, Test Accuracy: 80.28%
Epoch [57/100], Train Loss: 0.1158, Train Accuracy: 96.27%, Test Loss:
0.9837, Test Accuracy: 80.30%
Epoch [58/100], Train Loss: 0.0555, Train Accuracy: 98.25%, Test Loss:
0.8642, Test Accuracy: 82.46%
Epoch [59/100], Train Loss: 0.0400, Train Accuracy: 98.69%, Test Loss:
1.0151, Test Accuracy: 81.11%
Epoch [60/100], Train Loss: 0.0368, Train Accuracy: 98.83%, Test Loss:
1.0519, Test Accuracy: 81.32%
Epoch [61/100], Train Loss: 0.0663, Train Accuracy: 98.01%, Test Loss:
1.1461, Test Accuracy: 81.14%
Epoch [62/100], Train Loss: 0.0656, Train Accuracy: 97.88%, Test Loss:
0.9858, Test Accuracy: 81.49%
Epoch [63/100], Train Loss: 0.0509, Train Accuracy: 98.47%, Test Loss:
0.9823, Test Accuracy: 81.06%
Epoch [64/100], Train Loss: 0.0633, Train Accuracy: 97.99%, Test Loss:
1.1092, Test Accuracy: 80.06%
Epoch [65/100], Train Loss: 0.0675, Train Accuracy: 97.85%, Test Loss:
1.0206, Test Accuracy: 81.95%
Epoch [66/100], Train Loss: 0.0411, Train Accuracy: 98.63%, Test Loss:
1.0171, Test Accuracy: 80.77%
Epoch [67/100], Train Loss: 0.0513, Train Accuracy: 98.35%, Test Loss:
1.0906, Test Accuracy: 80.25%
Epoch [68/100], Train Loss: 0.0587, Train Accuracy: 98.15%, Test Loss:
```

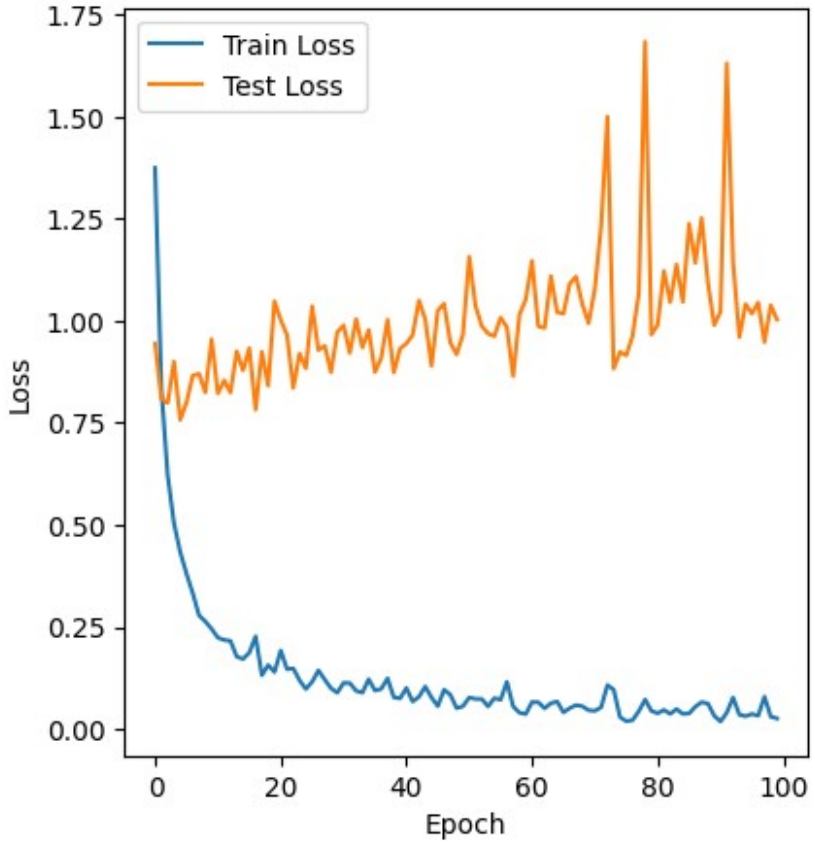
```
1.1078, Test Accuracy: 80.83%
Epoch [69/100], Train Loss: 0.0560, Train Accuracy: 98.29%, Test Loss:
1.0408, Test Accuracy: 80.39%
Epoch [70/100], Train Loss: 0.0462, Train Accuracy: 98.45%, Test Loss:
0.9936, Test Accuracy: 81.72%
Epoch [71/100], Train Loss: 0.0448, Train Accuracy: 98.50%, Test Loss:
1.0777, Test Accuracy: 80.20%
Epoch [72/100], Train Loss: 0.0534, Train Accuracy: 98.26%, Test Loss:
1.2319, Test Accuracy: 78.01%
Epoch [73/100], Train Loss: 0.1077, Train Accuracy: 96.84%, Test Loss:
1.4994, Test Accuracy: 73.75%
Epoch [74/100], Train Loss: 0.0966, Train Accuracy: 96.95%, Test Loss:
0.8825, Test Accuracy: 82.05%
Epoch [75/100], Train Loss: 0.0294, Train Accuracy: 98.91%, Test Loss:
0.9233, Test Accuracy: 83.05%
Epoch [76/100], Train Loss: 0.0193, Train Accuracy: 99.32%, Test Loss:
0.9150, Test Accuracy: 83.02%
Epoch [77/100], Train Loss: 0.0218, Train Accuracy: 99.28%, Test Loss:
0.9625, Test Accuracy: 82.25%
Epoch [78/100], Train Loss: 0.0440, Train Accuracy: 98.57%, Test Loss:
1.0679, Test Accuracy: 80.72%
Epoch [79/100], Train Loss: 0.0722, Train Accuracy: 97.82%, Test Loss:
1.6828, Test Accuracy: 72.29%
Epoch [80/100], Train Loss: 0.0445, Train Accuracy: 98.63%, Test Loss:
0.9660, Test Accuracy: 81.72%
Epoch [81/100], Train Loss: 0.0389, Train Accuracy: 98.67%, Test Loss:
0.9899, Test Accuracy: 82.18%
Epoch [82/100], Train Loss: 0.0469, Train Accuracy: 98.51%, Test Loss:
1.1215, Test Accuracy: 81.13%
Epoch [83/100], Train Loss: 0.0374, Train Accuracy: 98.71%, Test Loss:
1.0451, Test Accuracy: 80.90%
Epoch [84/100], Train Loss: 0.0491, Train Accuracy: 98.53%, Test Loss:
1.1376, Test Accuracy: 80.70%
Epoch [85/100], Train Loss: 0.0372, Train Accuracy: 98.83%, Test Loss:
1.0462, Test Accuracy: 81.49%
Epoch [86/100], Train Loss: 0.0384, Train Accuracy: 98.68%, Test Loss:
1.2369, Test Accuracy: 79.92%
Epoch [87/100], Train Loss: 0.0549, Train Accuracy: 98.41%, Test Loss:
1.1415, Test Accuracy: 80.04%
Epoch [88/100], Train Loss: 0.0661, Train Accuracy: 97.89%, Test Loss:
1.2513, Test Accuracy: 78.23%
Epoch [89/100], Train Loss: 0.0625, Train Accuracy: 98.08%, Test Loss:
1.0919, Test Accuracy: 80.16%
Epoch [90/100], Train Loss: 0.0326, Train Accuracy: 98.92%, Test Loss:
0.9892, Test Accuracy: 82.32%
Epoch [91/100], Train Loss: 0.0186, Train Accuracy: 99.42%, Test Loss:
1.0224, Test Accuracy: 82.07%
Epoch [92/100], Train Loss: 0.0402, Train Accuracy: 98.72%, Test Loss:
1.6298, Test Accuracy: 76.04%
```

```
Epoch [93/100], Train Loss: 0.0772, Train Accuracy: 97.50%, Test Loss:
1.1437, Test Accuracy: 79.13%
Epoch [94/100], Train Loss: 0.0353, Train Accuracy: 98.83%, Test Loss:
0.9599, Test Accuracy: 81.98%
Epoch [95/100], Train Loss: 0.0317, Train Accuracy: 99.02%, Test Loss:
1.0406, Test Accuracy: 80.77%
Epoch [96/100], Train Loss: 0.0370, Train Accuracy: 98.92%, Test Loss:
1.0170, Test Accuracy: 81.58%
Epoch [97/100], Train Loss: 0.0330, Train Accuracy: 98.98%, Test Loss:
1.0445, Test Accuracy: 81.62%
Epoch [98/100], Train Loss: 0.0794, Train Accuracy: 97.68%, Test Loss:
0.9475, Test Accuracy: 81.76%
Epoch [99/100], Train Loss: 0.0305, Train Accuracy: 99.02%, Test Loss:
1.0375, Test Accuracy: 80.67%
Epoch [100/100], Train Loss: 0.0262, Train Accuracy: 99.15%, Test
Loss: 1.0027, Test Accuracy: 81.79%
```

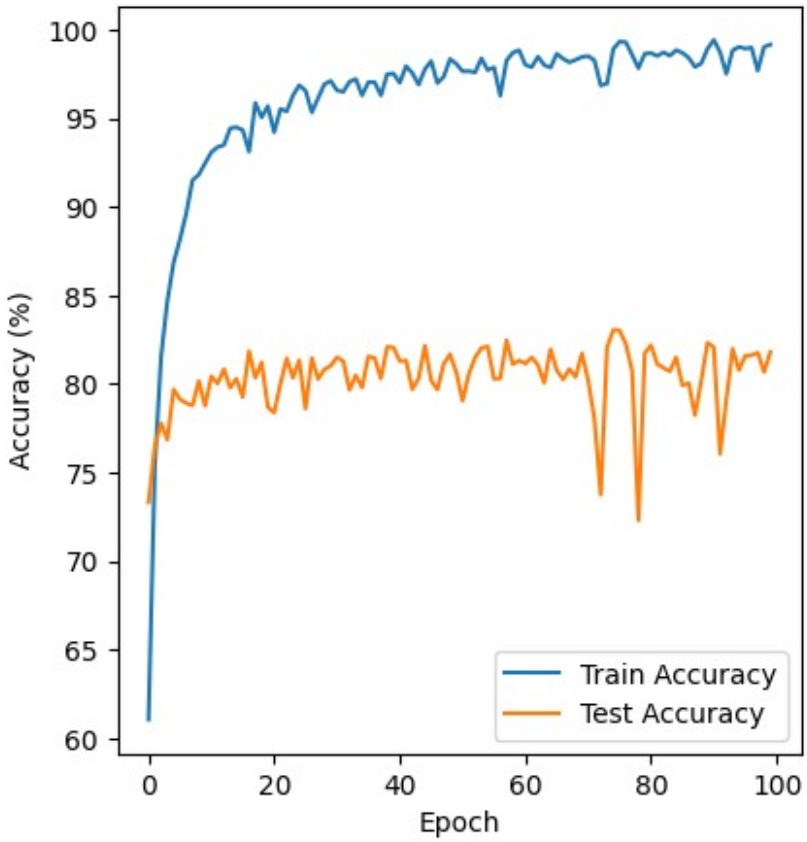



```
Train Loss: 0.0479, Train Accuracy: 98.62%, Test Loss: 1.0172, Test
Accuracy: 81.28%

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7d9905fd5240>
```




```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 2)
plt.plot(train_accuracies, label='Train Accuracy')
plt.plot(test_accuracies, label='Test Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy (%)')
plt.legend()

<matplotlib.legend.Legend at 0x7d9905d8b550>
```




```
torch.save(model.state_dict(), 'model.pth')
```