```python
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, 155MB/s]
```

```python
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/style-dataset/style_training_tensor_stack_in
put.pt')

X_test =
torch.load('/kaggle/input/style-dataset/style_testing_tensor_stack_inp
ut.pt')

y_train =
torch.load('/kaggle/input/style-dataset/style_training_tensor_output.p
t')

y_test =
torch.load('/kaggle/input/style-dataset/style_testing_tensor_output.pt
')

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

print(first_tensor_shape)
```

```
torch.Size([14257, 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=27, 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=27, 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: 5349015

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}%")
```

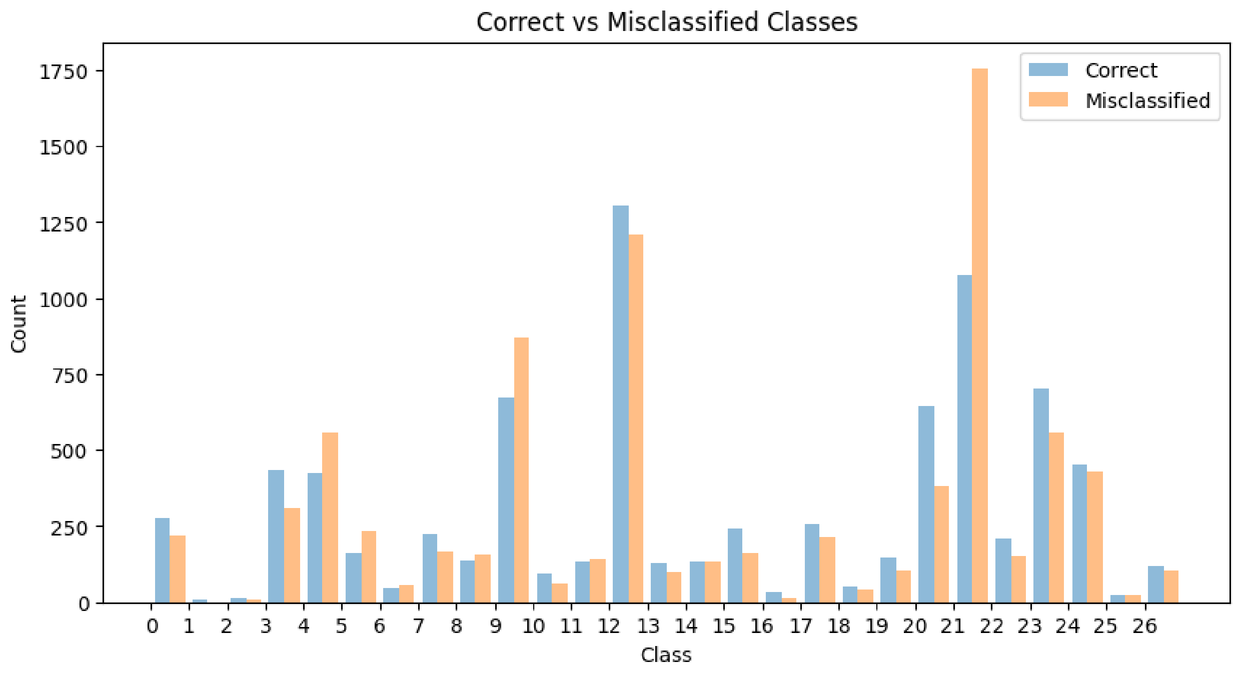
```
Epoch [1/100], Train Loss: 2.2053, Train Accuracy: 33.85%, Test Loss:
1.8535, Test Accuracy: 41.25%
Epoch [2/100], Train Loss: 1.8116, Train Accuracy: 42.98%, Test Loss:
1.8900, Test Accuracy: 44.37%
Epoch [3/100], Train Loss: 1.6742, Train Accuracy: 46.67%, Test Loss:
1.7670, Test Accuracy: 44.36%
Epoch [4/100], Train Loss: 1.5526, Train Accuracy: 49.58%, Test Loss:
2.0142, Test Accuracy: 44.82%
Epoch [5/100], Train Loss: 1.4127, Train Accuracy: 53.40%, Test Loss:
1.7440, Test Accuracy: 46.38%
Epoch [6/100], Train Loss: 1.2044, Train Accuracy: 60.10%, Test Loss:
1.7434, Test Accuracy: 47.87%
Epoch [7/100], Train Loss: 1.1016, Train Accuracy: 63.51%, Test Loss:
1.7406, Test Accuracy: 47.91%
Epoch [8/100], Train Loss: 0.9345, Train Accuracy: 68.65%, Test Loss:
1.8503, Test Accuracy: 46.87%
Epoch [9/100], Train Loss: 0.8659, Train Accuracy: 71.02%, Test Loss:
1.9778, Test Accuracy: 46.36%
Epoch [10/100], Train Loss: 0.6966, Train Accuracy: 76.54%, Test Loss:
2.0204, Test Accuracy: 46.70%
Epoch [11/100], Train Loss: 0.5695, Train Accuracy: 80.97%, Test Loss:
2.4306, Test Accuracy: 46.00%
Epoch [12/100], Train Loss: 0.5000, Train Accuracy: 83.60%, Test Loss:
2.4142, Test Accuracy: 46.47%
Epoch [13/100], Train Loss: 0.5198, Train Accuracy: 83.16%, Test Loss:
2.1500, Test Accuracy: 47.66%
Epoch [14/100], Train Loss: 0.4181, Train Accuracy: 86.36%, Test Loss:
2.3738, Test Accuracy: 48.38%
Epoch [15/100], Train Loss: 0.3582, Train Accuracy: 88.27%, Test Loss:
```

```
2.4306, Test Accuracy: 45.99%
Epoch [16/100], Train Loss: 0.3488, Train Accuracy: 88.68%, Test Loss:
2.5619, Test Accuracy: 46.08%
Epoch [17/100], Train Loss: 0.3583, Train Accuracy: 88.19%, Test Loss:
2.5225, Test Accuracy: 46.17%
Epoch [18/100], Train Loss: 0.2761, Train Accuracy: 91.09%, Test Loss:
2.5920, Test Accuracy: 47.98%
Epoch [19/100], Train Loss: 0.2976, Train Accuracy: 90.34%, Test Loss:
2.9521, Test Accuracy: 46.42%
Epoch [20/100], Train Loss: 0.3217, Train Accuracy: 89.48%, Test Loss:
2.6823, Test Accuracy: 46.15%
Epoch [21/100], Train Loss: 0.2513, Train Accuracy: 92.00%, Test Loss:
2.6881, Test Accuracy: 46.49%
Epoch [22/100], Train Loss: 0.2505, Train Accuracy: 92.09%, Test Loss:
2.7531, Test Accuracy: 46.27%
Epoch [23/100], Train Loss: 0.2232, Train Accuracy: 93.29%, Test Loss:
2.7540, Test Accuracy: 46.09%
Epoch [24/100], Train Loss: 0.2561, Train Accuracy: 91.85%, Test Loss:
2.9242, Test Accuracy: 44.90%
Epoch [25/100], Train Loss: 0.2353, Train Accuracy: 92.38%, Test Loss:
2.7447, Test Accuracy: 46.51%
Epoch [26/100], Train Loss: 0.2183, Train Accuracy: 92.80%, Test Loss:
3.0100, Test Accuracy: 45.71%
Epoch [27/100], Train Loss: 0.2137, Train Accuracy: 93.40%, Test Loss:
2.9245, Test Accuracy: 46.65%
Epoch [28/100], Train Loss: 0.2225, Train Accuracy: 92.88%, Test Loss:
2.8782, Test Accuracy: 45.89%
Epoch [29/100], Train Loss: 0.2093, Train Accuracy: 93.29%, Test Loss:
2.7460, Test Accuracy: 46.24%
Epoch [30/100], Train Loss: 0.1938, Train Accuracy: 93.94%, Test Loss:
2.9166, Test Accuracy: 46.85%
Epoch [31/100], Train Loss: 0.1873, Train Accuracy: 94.26%, Test Loss:
2.9475, Test Accuracy: 44.34%
Epoch [32/100], Train Loss: 0.1769, Train Accuracy: 94.33%, Test Loss:
3.0480, Test Accuracy: 45.46%
Epoch [33/100], Train Loss: 0.1577, Train Accuracy: 94.96%, Test Loss:
2.7760, Test Accuracy: 47.12%
Epoch [34/100], Train Loss: 0.1611, Train Accuracy: 94.88%, Test Loss:
2.9508, Test Accuracy: 46.94%
Epoch [35/100], Train Loss: 0.1823, Train Accuracy: 94.30%, Test Loss:
2.9182, Test Accuracy: 45.88%
Epoch [36/100], Train Loss: 0.1651, Train Accuracy: 94.62%, Test Loss:
3.0523, Test Accuracy: 44.97%
Epoch [37/100], Train Loss: 0.1799, Train Accuracy: 94.40%, Test Loss:
2.9615, Test Accuracy: 45.66%
Epoch [38/100], Train Loss: 0.1547, Train Accuracy: 94.99%, Test Loss:
2.9554, Test Accuracy: 47.17%
Epoch [39/100], Train Loss: 0.1705, Train Accuracy: 94.67%, Test Loss:
2.9916, Test Accuracy: 46.35%
```

```
Epoch [40/100], Train Loss: 0.1515, Train Accuracy: 95.17%, Test Loss:
2.9890, Test Accuracy: 46.11%
Epoch [41/100], Train Loss: 0.1359, Train Accuracy: 95.66%, Test Loss:
3.2237, Test Accuracy: 46.20%
Epoch [42/100], Train Loss: 0.1478, Train Accuracy: 95.17%, Test Loss:
3.2023, Test Accuracy: 44.86%
Epoch [43/100], Train Loss: 0.1421, Train Accuracy: 95.45%, Test Loss:
3.1402, Test Accuracy: 45.09%
Epoch [44/100], Train Loss: 0.1331, Train Accuracy: 95.77%, Test Loss:
3.2053, Test Accuracy: 45.68%
Epoch [45/100], Train Loss: 0.1333, Train Accuracy: 95.67%, Test Loss:
3.1953, Test Accuracy: 45.94%
Epoch [46/100], Train Loss: 0.1397, Train Accuracy: 95.29%, Test Loss:
3.1538, Test Accuracy: 44.55%
Epoch [47/100], Train Loss: 0.1331, Train Accuracy: 95.66%, Test Loss:
3.0936, Test Accuracy: 46.19%
Epoch [48/100], Train Loss: 0.1217, Train Accuracy: 96.02%, Test Loss:
3.3370, Test Accuracy: 44.61%
Epoch [49/100], Train Loss: 0.1178, Train Accuracy: 95.92%, Test Loss:
3.3533, Test Accuracy: 44.23%
Epoch [50/100], Train Loss: 0.1701, Train Accuracy: 94.47%, Test Loss:
3.1770, Test Accuracy: 44.23%
Epoch [51/100], Train Loss: 0.1455, Train Accuracy: 95.15%, Test Loss:
2.9849, Test Accuracy: 46.58%
Epoch [52/100], Train Loss: 0.1036, Train Accuracy: 96.54%, Test Loss:
3.1972, Test Accuracy: 45.41%
Epoch [53/100], Train Loss: 0.1086, Train Accuracy: 96.45%, Test Loss:
3.1375, Test Accuracy: 46.59%
Epoch [54/100], Train Loss: 0.1072, Train Accuracy: 96.39%, Test Loss:
3.1897, Test Accuracy: 46.80%
Epoch [55/100], Train Loss: 0.1001, Train Accuracy: 96.77%, Test Loss:
3.3808, Test Accuracy: 46.24%
Epoch [56/100], Train Loss: 0.1142, Train Accuracy: 96.21%, Test Loss:
3.4037, Test Accuracy: 44.42%
Epoch [57/100], Train Loss: 0.1152, Train Accuracy: 96.18%, Test Loss:
3.2763, Test Accuracy: 45.57%
Epoch [58/100], Train Loss: 0.1144, Train Accuracy: 96.29%, Test Loss:
3.3249, Test Accuracy: 45.74%
Epoch [59/100], Train Loss: 0.1158, Train Accuracy: 96.09%, Test Loss:
3.2801, Test Accuracy: 44.55%
Epoch [60/100], Train Loss: 0.0999, Train Accuracy: 96.81%, Test Loss:
3.5549, Test Accuracy: 44.18%
Epoch [61/100], Train Loss: 0.1094, Train Accuracy: 96.28%, Test Loss:
3.4604, Test Accuracy: 43.83%
Epoch [62/100], Train Loss: 0.1216, Train Accuracy: 96.01%, Test Loss:
3.3056, Test Accuracy: 45.11%
Epoch [63/100], Train Loss: 0.0944, Train Accuracy: 96.73%, Test Loss:
3.2515, Test Accuracy: 45.92%
Epoch [64/100], Train Loss: 0.1016, Train Accuracy: 96.54%, Test Loss:
```

```
3.4849, Test Accuracy: 45.79%
Epoch [65/100], Train Loss: 0.0937, Train Accuracy: 97.00%, Test Loss:
3.4148, Test Accuracy: 44.85%
Epoch [66/100], Train Loss: 0.0986, Train Accuracy: 96.60%, Test Loss:
3.2140, Test Accuracy: 46.05%
Epoch [67/100], Train Loss: 0.0814, Train Accuracy: 97.17%, Test Loss:
3.4786, Test Accuracy: 45.13%
Epoch [68/100], Train Loss: 0.0965, Train Accuracy: 96.82%, Test Loss:
3.3151, Test Accuracy: 45.29%
Epoch [69/100], Train Loss: 0.1120, Train Accuracy: 96.41%, Test Loss:
3.3206, Test Accuracy: 44.83%
Epoch [70/100], Train Loss: 0.0862, Train Accuracy: 97.06%, Test Loss:
3.4173, Test Accuracy: 44.82%
Epoch [71/100], Train Loss: 0.1019, Train Accuracy: 96.68%, Test Loss:
3.4831, Test Accuracy: 44.55%
Epoch [72/100], Train Loss: 0.0903, Train Accuracy: 96.95%, Test Loss:
3.3935, Test Accuracy: 45.07%
Epoch [73/100], Train Loss: 0.0817, Train Accuracy: 97.22%, Test Loss:
3.3721, Test Accuracy: 45.85%
Epoch [74/100], Train Loss: 0.0920, Train Accuracy: 96.88%, Test Loss:
3.4417, Test Accuracy: 44.68%
Epoch [75/100], Train Loss: 0.0840, Train Accuracy: 97.08%, Test Loss:
3.5810, Test Accuracy: 43.41%
Epoch [76/100], Train Loss: 0.0908, Train Accuracy: 97.00%, Test Loss:
3.5253, Test Accuracy: 45.14%
Epoch [77/100], Train Loss: 0.0799, Train Accuracy: 97.25%, Test Loss:
3.6178, Test Accuracy: 44.55%
Epoch [78/100], Train Loss: 0.0952, Train Accuracy: 96.89%, Test Loss:
3.4741, Test Accuracy: 45.68%
Epoch [79/100], Train Loss: 0.0772, Train Accuracy: 97.29%, Test Loss:
3.4584, Test Accuracy: 45.35%
Epoch [80/100], Train Loss: 0.0744, Train Accuracy: 97.52%, Test Loss:
3.5108, Test Accuracy: 44.82%
Epoch [81/100], Train Loss: 0.0765, Train Accuracy: 97.40%, Test Loss:
3.4846, Test Accuracy: 46.01%
Epoch [82/100], Train Loss: 0.0882, Train Accuracy: 97.02%, Test Loss:
3.6539, Test Accuracy: 45.22%
Epoch [83/100], Train Loss: 0.0826, Train Accuracy: 97.24%, Test Loss:
3.6073, Test Accuracy: 44.37%
Epoch [84/100], Train Loss: 0.0808, Train Accuracy: 97.08%, Test Loss:
3.4670, Test Accuracy: 45.49%
Epoch [85/100], Train Loss: 0.0832, Train Accuracy: 97.15%, Test Loss:
3.5541, Test Accuracy: 45.23%
Epoch [86/100], Train Loss: 0.0692, Train Accuracy: 97.57%, Test Loss:
3.4871, Test Accuracy: 45.24%
Epoch [87/100], Train Loss: 0.0881, Train Accuracy: 96.84%, Test Loss:
3.6760, Test Accuracy: 44.77%
Epoch [88/100], Train Loss: 0.0684, Train Accuracy: 97.62%, Test Loss:
3.6744, Test Accuracy: 45.28%
```

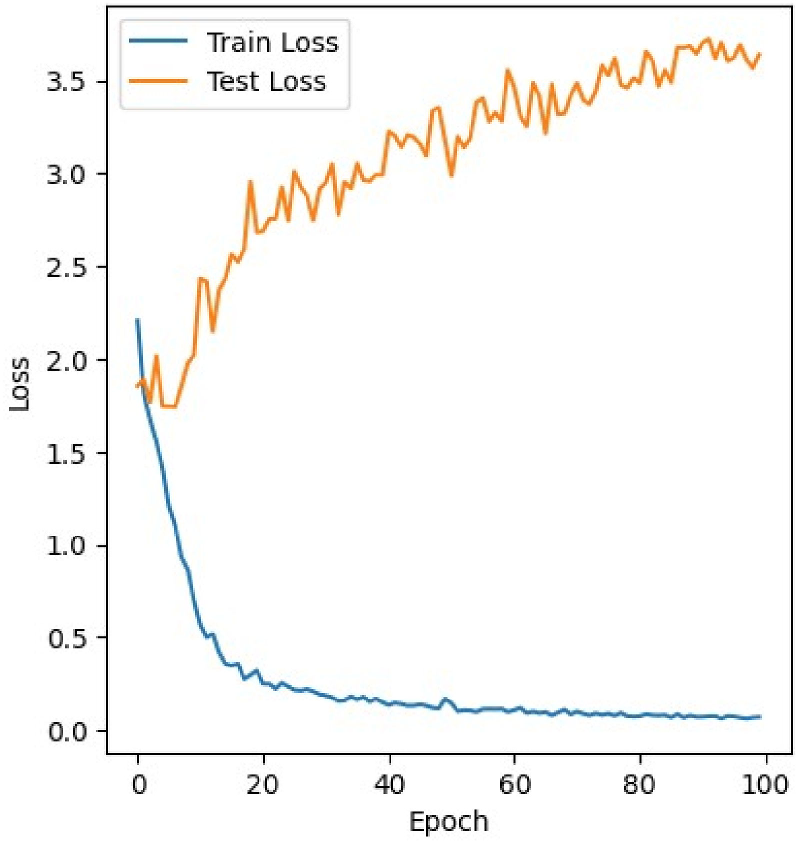
```
Epoch [89/100], Train Loss: 0.0807, Train Accuracy: 97.20%, Test Loss:
3.6822, Test Accuracy: 45.02%
Epoch [90/100], Train Loss: 0.0727, Train Accuracy: 97.37%, Test Loss:
3.6421, Test Accuracy: 45.40%
Epoch [91/100], Train Loss: 0.0731, Train Accuracy: 97.44%, Test Loss:
3.7020, Test Accuracy: 44.63%
Epoch [92/100], Train Loss: 0.0763, Train Accuracy: 97.31%, Test Loss:
3.7225, Test Accuracy: 44.56%
Epoch [93/100], Train Loss: 0.0773, Train Accuracy: 97.28%, Test Loss:
3.6171, Test Accuracy: 45.82%
Epoch [94/100], Train Loss: 0.0635, Train Accuracy: 97.88%, Test Loss:
3.7022, Test Accuracy: 45.53%
Epoch [95/100], Train Loss: 0.0780, Train Accuracy: 97.39%, Test Loss:
3.6034, Test Accuracy: 45.07%
Epoch [96/100], Train Loss: 0.0755, Train Accuracy: 97.45%, Test Loss:
3.6218, Test Accuracy: 45.30%
Epoch [97/100], Train Loss: 0.0689, Train Accuracy: 97.39%, Test Loss:
3.6894, Test Accuracy: 45.09%
Epoch [98/100], Train Loss: 0.0636, Train Accuracy: 97.80%, Test Loss:
3.6096, Test Accuracy: 46.26%
Epoch [99/100], Train Loss: 0.0699, Train Accuracy: 97.59%, Test Loss:
3.5652, Test Accuracy: 44.99%
Epoch [100/100], Train Loss: 0.0724, Train Accuracy: 97.40%, Test
Loss: 3.6371, Test Accuracy: 46.08%
```



```
Train Loss: 0.0656, Train Accuracy: 97.65%, Test Loss: 3.8095, Test
Accuracy: 44.97%
```

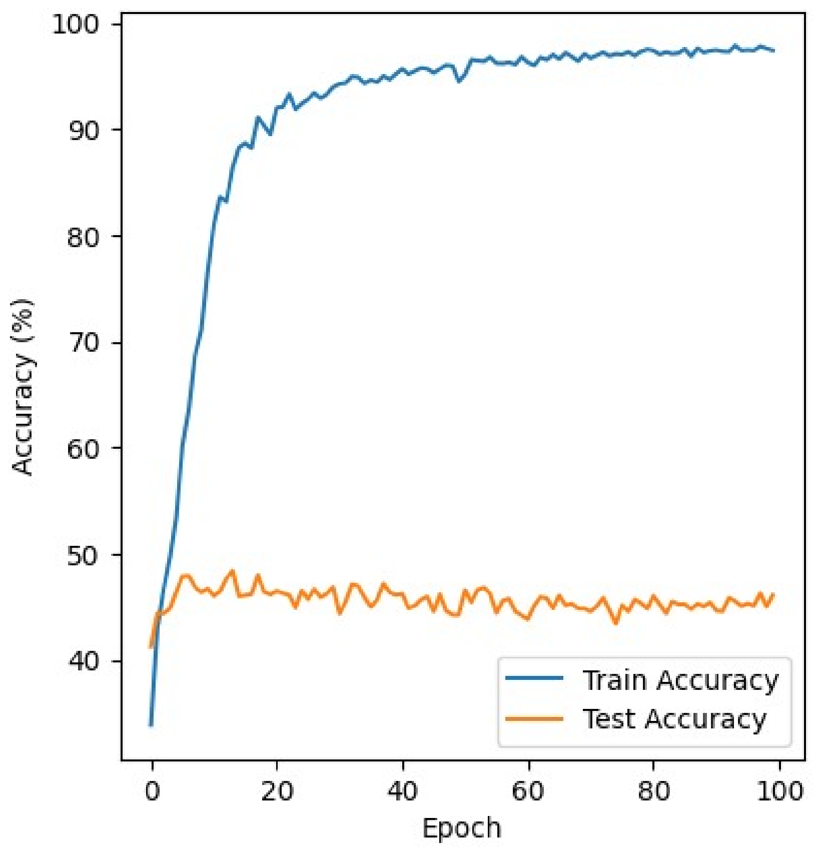
```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x79326d72ea10>
```



```
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 0x793264e9f070>
```



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