```
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, 63.8MB/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/genre-dataset/genre_training_tensor_stack_in
put.pt')

X_test =
torch.load('/kaggle/input/genre-dataset/genre_testing_tensor_stack_inp
ut.pt')

y_train =
torch.load('/kaggle/input/genre-dataset/genre_training_tensor_output.p
t')

y_test =
torch.load('/kaggle/input/genre-dataset/genre_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([15167, 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=10, 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=10, 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: 5331590

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}%")
```

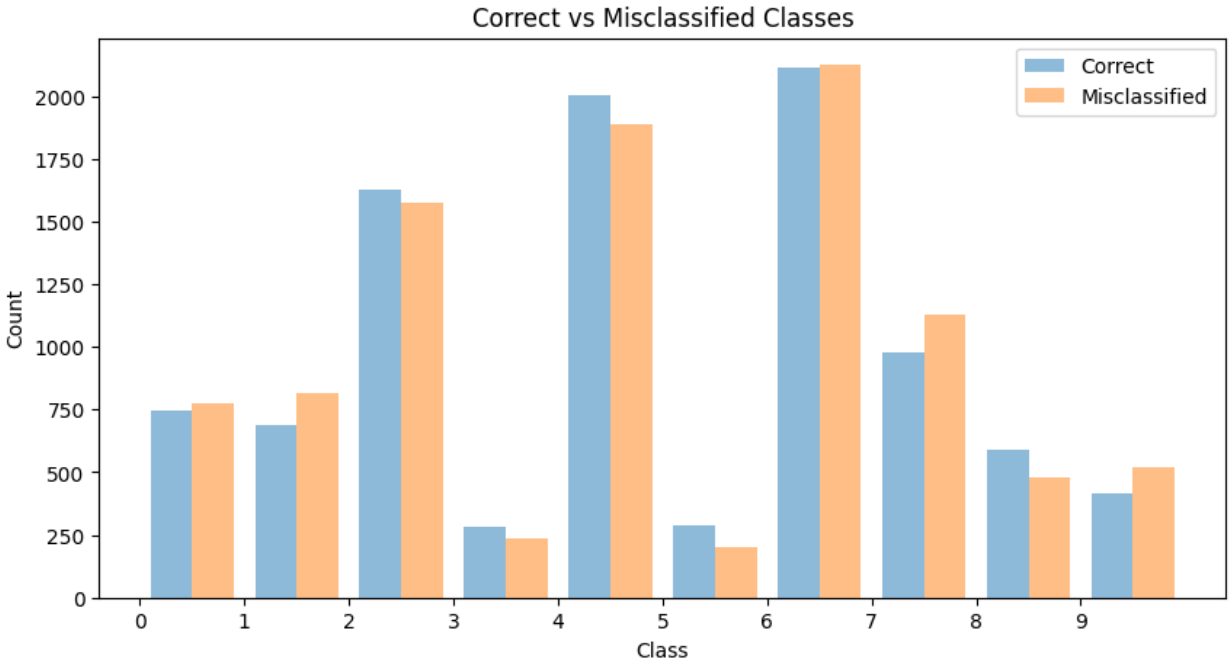
```
Epoch [1/100], Train Loss: 1.1476, Train Accuracy: 62.53%, Test Loss:
1.1433, Test Accuracy: 61.31%
Epoch [2/100], Train Loss: 0.9000, Train Accuracy: 69.54%, Test Loss:
0.9229, Test Accuracy: 67.82%
Epoch [3/100], Train Loss: 0.7905, Train Accuracy: 73.44%, Test Loss:
0.9088, Test Accuracy: 69.56%
Epoch [4/100], Train Loss: 0.7026, Train Accuracy: 75.99%, Test Loss:
0.8492, Test Accuracy: 71.90%
Epoch [5/100], Train Loss: 0.7400, Train Accuracy: 75.95%, Test Loss:
0.8878, Test Accuracy: 70.69%
Epoch [6/100], Train Loss: 0.5690, Train Accuracy: 80.66%, Test Loss:
0.9330, Test Accuracy: 71.29%
Epoch [7/100], Train Loss: 0.5458, Train Accuracy: 82.12%, Test Loss:
0.9577, Test Accuracy: 70.35%
Epoch [8/100], Train Loss: 0.4515, Train Accuracy: 84.55%, Test Loss:
0.9629, Test Accuracy: 71.10%
Epoch [9/100], Train Loss: 0.3788, Train Accuracy: 87.39%, Test Loss:
1.1264, Test Accuracy: 69.13%
Epoch [10/100], Train Loss: 0.3332, Train Accuracy: 88.83%, Test Loss:
1.0635, Test Accuracy: 69.98%
Epoch [11/100], Train Loss: 0.3352, Train Accuracy: 89.05%, Test Loss:
1.1630, Test Accuracy: 69.75%
Epoch [12/100], Train Loss: 0.2939, Train Accuracy: 90.57%, Test Loss:
1.2754, Test Accuracy: 68.37%
Epoch [13/100], Train Loss: 0.2047, Train Accuracy: 93.16%, Test Loss:
1.1705, Test Accuracy: 70.05%
Epoch [14/100], Train Loss: 0.3179, Train Accuracy: 89.64%, Test Loss:
```

```
1.1674, Test Accuracy: 69.01%
Epoch [15/100], Train Loss: 0.2702, Train Accuracy: 91.05%, Test Loss:
1.2812, Test Accuracy: 69.00%
Epoch [16/100], Train Loss: 0.1973, Train Accuracy: 93.72%, Test Loss:
1.2944, Test Accuracy: 69.73%
Epoch [17/100], Train Loss: 0.1649, Train Accuracy: 94.42%, Test Loss:
1.2901, Test Accuracy: 69.93%
Epoch [18/100], Train Loss: 0.1340, Train Accuracy: 95.46%, Test Loss:
1.3701, Test Accuracy: 69.83%
Epoch [19/100], Train Loss: 0.1643, Train Accuracy: 94.80%, Test Loss:
1.5545, Test Accuracy: 66.70%
Epoch [20/100], Train Loss: 0.1897, Train Accuracy: 93.76%, Test Loss:
1.4238, Test Accuracy: 69.00%
Epoch [21/100], Train Loss: 0.1634, Train Accuracy: 94.64%, Test Loss:
1.3716, Test Accuracy: 69.58%
Epoch [22/100], Train Loss: 0.1247, Train Accuracy: 95.96%, Test Loss:
1.4101, Test Accuracy: 70.30%
Epoch [23/100], Train Loss: 0.1082, Train Accuracy: 96.29%, Test Loss:
1.5503, Test Accuracy: 69.15%
Epoch [24/100], Train Loss: 0.1209, Train Accuracy: 96.09%, Test Loss:
1.6097, Test Accuracy: 68.86%
Epoch [25/100], Train Loss: 0.1194, Train Accuracy: 96.03%, Test Loss:
1.5228, Test Accuracy: 69.53%
Epoch [26/100], Train Loss: 0.1028, Train Accuracy: 96.64%, Test Loss:
1.5763, Test Accuracy: 69.51%
Epoch [27/100], Train Loss: 0.1694, Train Accuracy: 94.59%, Test Loss:
1.4749, Test Accuracy: 68.55%
Epoch [28/100], Train Loss: 0.1395, Train Accuracy: 95.48%, Test Loss:
1.4771, Test Accuracy: 69.40%
Epoch [29/100], Train Loss: 0.0951, Train Accuracy: 96.82%, Test Loss:
1.4850, Test Accuracy: 69.92%
Epoch [30/100], Train Loss: 0.1031, Train Accuracy: 96.64%, Test Loss:
1.5533, Test Accuracy: 68.90%
Epoch [31/100], Train Loss: 0.1023, Train Accuracy: 96.74%, Test Loss:
1.6169, Test Accuracy: 69.82%
Epoch [32/100], Train Loss: 0.0868, Train Accuracy: 97.07%, Test Loss:
1.6559, Test Accuracy: 68.09%
Epoch [33/100], Train Loss: 0.0933, Train Accuracy: 96.87%, Test Loss:
1.6524, Test Accuracy: 68.69%
Epoch [34/100], Train Loss: 0.1007, Train Accuracy: 96.82%, Test Loss:
1.6826, Test Accuracy: 67.58%
Epoch [35/100], Train Loss: 0.1083, Train Accuracy: 96.55%, Test Loss:
1.7072, Test Accuracy: 68.94%
Epoch [36/100], Train Loss: 0.0810, Train Accuracy: 97.36%, Test Loss:
1.6635, Test Accuracy: 69.41%
Epoch [37/100], Train Loss: 0.1065, Train Accuracy: 96.38%, Test Loss:
1.5606, Test Accuracy: 68.99%
Epoch [38/100], Train Loss: 0.0879, Train Accuracy: 97.05%, Test Loss:
1.6227, Test Accuracy: 69.16%
```

```
Epoch [39/100], Train Loss: 0.1123, Train Accuracy: 96.55%, Test Loss: 2.2619, Test Accuracy: 68.92%
Epoch [40/100], Train Loss: 0.0686, Train Accuracy: 97.82%, Test Loss: 1.6426, Test Accuracy: 69.18%
Epoch [41/100], Train Loss: 0.0891, Train Accuracy: 97.16%, Test Loss: 1.9443, Test Accuracy: 67.24%
Epoch [42/100], Train Loss: 0.0940, Train Accuracy: 97.05%, Test Loss: 3.3406, Test Accuracy: 68.56%
Epoch [43/100], Train Loss: 0.1062, Train Accuracy: 96.61%, Test Loss: 1.8832, Test Accuracy: 67.99%
Epoch [44/100], Train Loss: 0.1039, Train Accuracy: 96.57%, Test Loss: 1.7376, Test Accuracy: 68.37%
Epoch [45/100], Train Loss: 0.0596, Train Accuracy: 98.16%, Test Loss: 2.2769, Test Accuracy: 68.76%
Epoch [46/100], Train Loss: 0.0470, Train Accuracy: 98.39%, Test Loss: 2.3254, Test Accuracy: 69.26%
Epoch [47/100], Train Loss: 0.0695, Train Accuracy: 97.73%, Test Loss: 3.0339, Test Accuracy: 67.46%
Epoch [48/100], Train Loss: 0.0776, Train Accuracy: 97.50%, Test Loss: 2.0568, Test Accuracy: 68.72%
Epoch [49/100], Train Loss: 0.0560, Train Accuracy: 98.15%, Test Loss: 1.9578, Test Accuracy: 68.73%
Epoch [50/100], Train Loss: 0.0601, Train Accuracy: 97.90%, Test Loss: 1.9795, Test Accuracy: 67.45%
Epoch [51/100], Train Loss: 0.0720, Train Accuracy: 97.58%, Test Loss: 1.7999, Test Accuracy: 68.01%
Epoch [52/100], Train Loss: 0.1110, Train Accuracy: 96.49%, Test Loss: 1.7741, Test Accuracy: 67.62%
Epoch [53/100], Train Loss: 0.0855, Train Accuracy: 97.21%, Test Loss: 1.9442, Test Accuracy: 69.20%
Epoch [54/100], Train Loss: 0.0328, Train Accuracy: 98.93%, Test Loss: 1.8769, Test Accuracy: 70.12%
Epoch [55/100], Train Loss: 0.1044, Train Accuracy: 96.62%, Test Loss: 1.7961, Test Accuracy: 67.94%
Epoch [56/100], Train Loss: 0.0528, Train Accuracy: 98.26%, Test Loss: 2.6770, Test Accuracy: 69.24%
Epoch [57/100], Train Loss: 0.0436, Train Accuracy: 98.60%, Test Loss: 1.8448, Test Accuracy: 69.61%
Epoch [58/100], Train Loss: 0.0407, Train Accuracy: 98.58%, Test Loss: 1.8917, Test Accuracy: 69.54%
Epoch [59/100], Train Loss: 0.0804, Train Accuracy: 97.55%, Test Loss: 1.8253, Test Accuracy: 68.09%
Epoch [60/100], Train Loss: 0.0870, Train Accuracy: 97.11%, Test Loss: 1.7296, Test Accuracy: 69.82%
Epoch [61/100], Train Loss: 0.0609, Train Accuracy: 97.87%, Test Loss: 1.8483, Test Accuracy: 69.51%
Epoch [62/100], Train Loss: 0.0473, Train Accuracy: 98.42%, Test Loss: 2.8624, Test Accuracy: 69.96%
Epoch [63/100], Train Loss: 0.0468, Train Accuracy: 98.40%, Test Loss:
```

```
2.9524, Test Accuracy: 69.20%
Epoch [64/100], Train Loss: 0.0409, Train Accuracy: 98.66%, Test Loss:
1.9867, Test Accuracy: 69.21%
Epoch [65/100], Train Loss: 0.0530, Train Accuracy: 98.34%, Test Loss:
1.9888, Test Accuracy: 69.63%
Epoch [66/100], Train Loss: 0.0548, Train Accuracy: 98.17%, Test Loss:
1.8254, Test Accuracy: 68.08%
Epoch [67/100], Train Loss: 0.0583, Train Accuracy: 98.16%, Test Loss:
1.9531, Test Accuracy: 69.05%
Epoch [68/100], Train Loss: 0.0536, Train Accuracy: 98.11%, Test Loss:
1.8477, Test Accuracy: 69.55%
Epoch [69/100], Train Loss: 0.0518, Train Accuracy: 98.29%, Test Loss:
2.0816, Test Accuracy: 68.83%
Epoch [70/100], Train Loss: 0.0398, Train Accuracy: 98.65%, Test Loss:
1.9433, Test Accuracy: 69.75%
Epoch [71/100], Train Loss: 0.0395, Train Accuracy: 98.69%, Test Loss:
1.8616, Test Accuracy: 69.69%
Epoch [72/100], Train Loss: 0.0516, Train Accuracy: 98.26%, Test Loss:
2.1152, Test Accuracy: 67.27%
Epoch [73/100], Train Loss: 0.1056, Train Accuracy: 96.68%, Test Loss:
1.7836, Test Accuracy: 68.69%
Epoch [74/100], Train Loss: 0.0344, Train Accuracy: 98.88%, Test Loss:
1.8247, Test Accuracy: 69.22%
Epoch [75/100], Train Loss: 0.0208, Train Accuracy: 99.29%, Test Loss:
2.1945, Test Accuracy: 68.82%
Epoch [76/100], Train Loss: 0.0350, Train Accuracy: 98.79%, Test Loss:
3.2570, Test Accuracy: 68.48%
Epoch [77/100], Train Loss: 0.0519, Train Accuracy: 98.34%, Test Loss:
2.2731, Test Accuracy: 63.60%
Epoch [78/100], Train Loss: 0.1007, Train Accuracy: 96.74%, Test Loss:
1.8263, Test Accuracy: 69.23%
Epoch [79/100], Train Loss: 0.0360, Train Accuracy: 98.85%, Test Loss:
1.8474, Test Accuracy: 70.10%
Epoch [80/100], Train Loss: 0.0282, Train Accuracy: 99.00%, Test Loss:
2.0654, Test Accuracy: 68.68%
Epoch [81/100], Train Loss: 0.0359, Train Accuracy: 98.79%, Test Loss:
2.0298, Test Accuracy: 68.98%
Epoch [82/100], Train Loss: 0.0418, Train Accuracy: 98.65%, Test Loss:
2.0374, Test Accuracy: 69.65%
Epoch [83/100], Train Loss: 0.0364, Train Accuracy: 98.76%, Test Loss:
2.1284, Test Accuracy: 69.15%
Epoch [84/100], Train Loss: 0.0707, Train Accuracy: 97.65%, Test Loss:
2.0023, Test Accuracy: 68.56%
Epoch [85/100], Train Loss: 0.0457, Train Accuracy: 98.56%, Test Loss:
2.0438, Test Accuracy: 69.17%
Epoch [86/100], Train Loss: 0.0254, Train Accuracy: 99.22%, Test Loss:
2.0625, Test Accuracy: 69.60%
Epoch [87/100], Train Loss: 0.0420, Train Accuracy: 98.65%, Test Loss:
2.0328, Test Accuracy: 67.88%
```

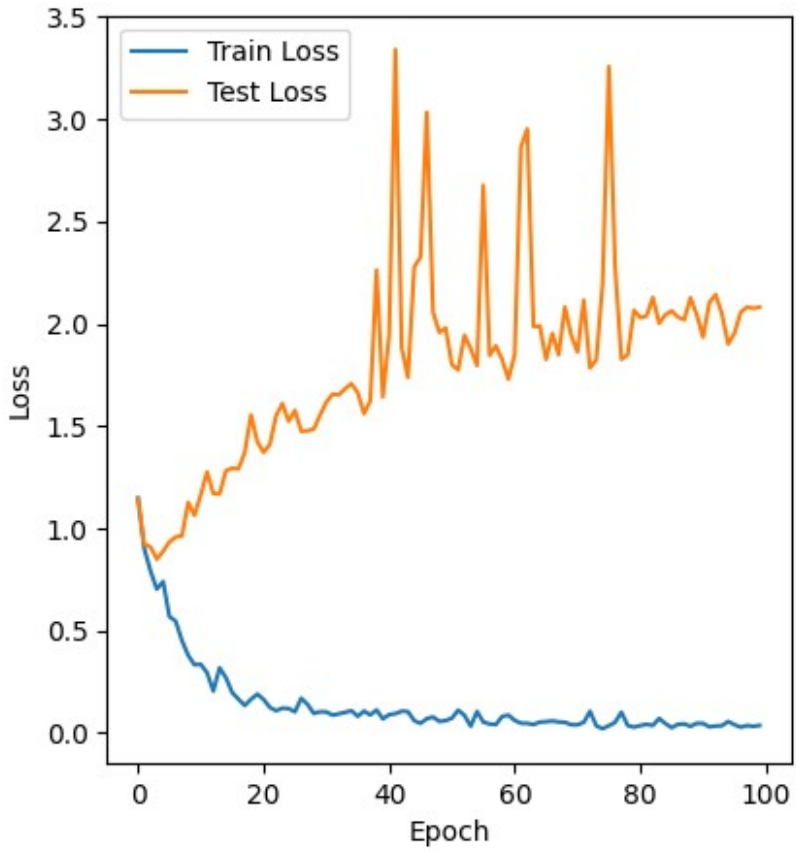
```
Epoch [88/100], Train Loss: 0.0439, Train Accuracy: 98.64%, Test Loss:
2.0195, Test Accuracy: 69.11%
Epoch [89/100], Train Loss: 0.0322, Train Accuracy: 98.88%, Test Loss:
2.1271, Test Accuracy: 68.29%
Epoch [90/100], Train Loss: 0.0468, Train Accuracy: 98.33%, Test Loss:
2.0440, Test Accuracy: 68.62%
Epoch [91/100], Train Loss: 0.0465, Train Accuracy: 98.47%, Test Loss:
1.9344, Test Accuracy: 69.25%
Epoch [92/100], Train Loss: 0.0297, Train Accuracy: 98.90%, Test Loss:
2.1033, Test Accuracy: 68.17%
Epoch [93/100], Train Loss: 0.0337, Train Accuracy: 98.93%, Test Loss:
2.1427, Test Accuracy: 68.85%
Epoch [94/100], Train Loss: 0.0352, Train Accuracy: 98.93%, Test Loss:
2.0494, Test Accuracy: 69.45%
Epoch [95/100], Train Loss: 0.0557, Train Accuracy: 98.22%, Test Loss:
1.9008, Test Accuracy: 67.25%
Epoch [96/100], Train Loss: 0.0399, Train Accuracy: 98.69%, Test Loss:
1.9538, Test Accuracy: 69.55%
Epoch [97/100], Train Loss: 0.0280, Train Accuracy: 99.06%, Test Loss:
2.0563, Test Accuracy: 68.80%
Epoch [98/100], Train Loss: 0.0360, Train Accuracy: 98.74%, Test Loss:
2.0810, Test Accuracy: 69.22%
Epoch [99/100], Train Loss: 0.0324, Train Accuracy: 98.83%, Test Loss:
2.0758, Test Accuracy: 69.33%
Epoch [100/100], Train Loss: 0.0367, Train Accuracy: 98.78%, Test
Loss: 2.0810, Test Accuracy: 69.39%
```



```
Train Loss: 0.0372, Train Accuracy: 98.81%, Test Loss: 1.9752, Test
Accuracy: 68.29%
```

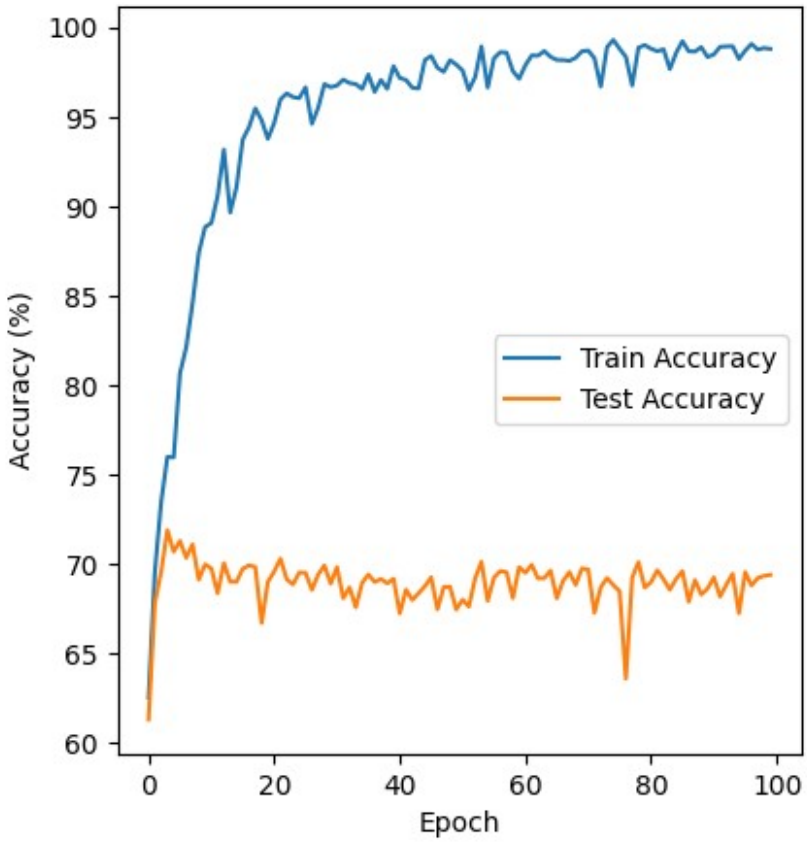
```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7d23ff8ac6d0>
```




```
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 0x7d23ff4a7910>
```




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