Tanawin-st123975-dlcv-cloth_color-classify

November 23, 2023

[ ]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

[ ]:
```
# from google.colab import files
# files.upload()
```

[ ]:
```
!nvidia-smi
```

```
Wed Nov 22 20:20:33 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2…  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   38C    P0    33W / 300W |      0MiB / 16384MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

[ ]:
```
# !unzip /content/drive/MyDrive/AIT/ISE/computer-vision/exam/
  ↪cloth_color_dataset/cloth_color_1200.zip -d data
```

[ ]:
```
!pip install efficientnet-pytorch
```

```
Requirement already satisfied: efficientnet-pytorch in
/usr/local/lib/python3.10/dist-packages (0.7.1)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(from efficientnet-pytorch) (2.1.0+cu118)
```

```
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch->efficientnet-pytorch) (3.13.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch->efficientnet-pytorch)
(4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch->efficientnet-pytorch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch->efficientnet-pytorch) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch->efficientnet-pytorch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch->efficientnet-pytorch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-
packages (from torch->efficientnet-pytorch) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch->efficientnet-
pytorch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch->efficientnet-pytorch) (1.3.0)
```

[ ]:
```
# !mkdir /content/drive/MyDrive/AIT/ISE/computer-vision/exam/report/
  ↪train-2000images-b7-v2-report
```

[ ]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision.datasets import ImageFolder
from torchvision.transforms import transforms
from efficientnet_pytorch import EfficientNet
# import pytorch_lightning as pl
# from pytorch_lightning.callbacks import ModelCheckpoint
```

[ ]:
```
# Set the device for training
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

[ ]:
```
device(type='cuda')
```

[ ]:
```
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

```
# Define the transformation applied to each image
# transform = transforms.Compose([
#     transforms.RandomResizedCrop(224),
#     transforms.RandomHorizontalFlip(),
#     transforms.RandomVerticalFlip(),
#     transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4,␣
 ↪hue=0.1),
#     transforms.RandomRotation(30),
#     transforms.RandomAffine(degrees=0, translate=(0.1, 0.1), scale=(0.9, 1.
 ↪1), shear=10),
#     transforms.ToTensor(),
#     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225])
# ])
```

[ ]:
```
train_data = "/content/data/cloth_color_1200/train"
val_data = "/content/data/cloth_color_1200/test"
```

[ ]:
```
# Load the training and validation datasets
train_dataset = ImageFolder(train_data, transform=transform)
valid_dataset = ImageFolder(val_data, transform=transform)
```

[ ]:
```
batch_size = 16
# Define the data loaders
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True,␣
 ↪num_workers=4)
valid_loader = DataLoader(valid_dataset, batch_size=batch_size, shuffle=False,␣
 ↪num_workers=4)
```

[ ]:
```
# Load the pre-trained EfficientNet model
model = EfficientNet.from_pretrained('efficientnet-b3', num_classes=8)
model.to(device)
```

```
Loaded pretrained weights for efficientnet-b3
```

[ ]:
```
EfficientNet(
  (_conv_stem): Conv2dStaticSamePadding(
    3, 40, kernel_size=(3, 3), stride=(2, 2), bias=False
    (static_padding): ZeroPad2d((0, 1, 0, 1))
  )
  (_bn0): BatchNorm2d(40, eps=0.001, momentum=0.010000000000000009, affine=True,
track_running_stats=True)
  (_blocks): ModuleList(
    (0): MBConvBlock(
      (_depthwise_conv): Conv2dStaticSamePadding(
        40, 40, kernel_size=(3, 3), stride=[1, 1], groups=40, bias=False
        (static_padding): ZeroPad2d((1, 1, 1, 1))
```

```
      )
      (_bn1): BatchNorm2d(40, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        40, 10, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        10, 40, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        40, 24, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(24, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (1): MBConvBlock(
      (_depthwise_conv): Conv2dStaticSamePadding(
        24, 24, kernel_size=(3, 3), stride=(1, 1), groups=24, bias=False
        (static_padding): ZeroPad2d((1, 1, 1, 1))
      )
      (_bn1): BatchNorm2d(24, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        24, 6, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        6, 24, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        24, 24, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(24, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (2): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
```

```
      (_bn0): BatchNorm2d(144, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        144, 144, kernel_size=(3, 3), stride=[2, 2], groups=144, bias=False
        (static_padding): ZeroPad2d((0, 1, 0, 1))
      )
      (_bn1): BatchNorm2d(144, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        144, 6, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        6, 144, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        144, 32, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(32, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (3-4): 2 x MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(192, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        192, 192, kernel_size=(3, 3), stride=(1, 1), groups=192, bias=False
        (static_padding): ZeroPad2d((1, 1, 1, 1))
      )
      (_bn1): BatchNorm2d(192, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        192, 8, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        8, 192, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False
```

```
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(32, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (5): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(192, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        192, 192, kernel_size=(5, 5), stride=[2, 2], groups=192, bias=False
        (static_padding): ZeroPad2d((2, 2, 2, 2))
      )
      (_bn1): BatchNorm2d(192, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        192, 8, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        8, 192, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        192, 48, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(48, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (6-7): 2 x MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(288, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        288, 288, kernel_size=(5, 5), stride=(1, 1), groups=288, bias=False
        (static_padding): ZeroPad2d((2, 2, 2, 2))
      )
      (_bn1): BatchNorm2d(288, eps=0.001, momentum=0.010000000000000009,
```

```
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        288, 12, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        12, 288, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(48, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (8): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(288, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        288, 288, kernel_size=(3, 3), stride=[2, 2], groups=288, bias=False
        (static_padding): ZeroPad2d((0, 1, 0, 1))
      )
      (_bn1): BatchNorm2d(288, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        288, 12, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        12, 288, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        288, 96, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(96, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (9-12): 4 x MBConvBlock(
```

```
      (_expand_conv): Conv2dStaticSamePadding(
        96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(576, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        576, 576, kernel_size=(3, 3), stride=(1, 1), groups=576, bias=False
        (static_padding): ZeroPad2d((1, 1, 1, 1))
      )
      (_bn1): BatchNorm2d(576, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        576, 24, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        24, 576, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        576, 96, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(96, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (13): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(576, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        576, 576, kernel_size=(5, 5), stride=[1, 1], groups=576, bias=False
        (static_padding): ZeroPad2d((2, 2, 2, 2))
      )
      (_bn1): BatchNorm2d(576, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        576, 24, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        24, 576, kernel_size=(1, 1), stride=(1, 1)
```

```
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        576, 136, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(136, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (14-17): 4 x MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        136, 816, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(816, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        816, 816, kernel_size=(5, 5), stride=(1, 1), groups=816, bias=False
        (static_padding): ZeroPad2d((2, 2, 2, 2))
      )
      (_bn1): BatchNorm2d(816, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        816, 34, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        34, 816, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        816, 136, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(136, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (18): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        136, 816, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(816, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
```

```
        816, 816, kernel_size=(5, 5), stride=[2, 2], groups=816, bias=False
        (static_padding): ZeroPad2d((2, 2, 2, 2))
      )
      (_bn1): BatchNorm2d(816, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        816, 34, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        34, 816, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        816, 232, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(232, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (19-23): 5 x MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        232, 1392, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(1392, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        1392, 1392, kernel_size=(5, 5), stride=(1, 1), groups=1392, bias=False
        (static_padding): ZeroPad2d((2, 2, 2, 2))
      )
      (_bn1): BatchNorm2d(1392, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        1392, 58, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        58, 1392, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        1392, 232, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(232, eps=0.001, momentum=0.010000000000000009,
```

```
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (24): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        232, 1392, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(1392, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        1392, 1392, kernel_size=(3, 3), stride=[1, 1], groups=1392, bias=False
        (static_padding): ZeroPad2d((1, 1, 1, 1))
      )
      (_bn1): BatchNorm2d(1392, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        1392, 58, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        58, 1392, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        1392, 384, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(384, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
    (25): MBConvBlock(
      (_expand_conv): Conv2dStaticSamePadding(
        384, 2304, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn0): BatchNorm2d(2304, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_depthwise_conv): Conv2dStaticSamePadding(
        2304, 2304, kernel_size=(3, 3), stride=(1, 1), groups=2304, bias=False
        (static_padding): ZeroPad2d((1, 1, 1, 1))
      )
      (_bn1): BatchNorm2d(2304, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_se_reduce): Conv2dStaticSamePadding(
        2304, 96, kernel_size=(1, 1), stride=(1, 1)
```

```
        (static_padding): Identity()
      )
      (_se_expand): Conv2dStaticSamePadding(
        96, 2304, kernel_size=(1, 1), stride=(1, 1)
        (static_padding): Identity()
      )
      (_project_conv): Conv2dStaticSamePadding(
        2304, 384, kernel_size=(1, 1), stride=(1, 1), bias=False
        (static_padding): Identity()
      )
      (_bn2): BatchNorm2d(384, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
      (_swish): MemoryEfficientSwish()
    )
  )
  (_conv_head): Conv2dStaticSamePadding(
    384, 1536, kernel_size=(1, 1), stride=(1, 1), bias=False
    (static_padding): Identity()
  )
  (_bn1): BatchNorm2d(1536, eps=0.001, momentum=0.010000000000000009,
affine=True, track_running_stats=True)
  (_avg_pooling): AdaptiveAvgPool2d(output_size=1)
  (_dropout): Dropout(p=0.3, inplace=False)
  (_fc): Linear(in_features=1536, out_features=8, bias=True)
  (_swish): MemoryEfficientSwish()
)
```

[ ]:
```
# Print the number of layers
num_layers = len(list(model.parameters()))
print("Number of layers:", num_layers)
```

```
Number of layers: 340
```

[ ]:
```
# Add dropout layer to the model
model._dropout = nn.Dropout(p=0.5)
```

[ ]:
```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
# optimizer = optim.Adam(model.parameters(), lr=0.000001)
# , weight_decay=1e-4
# optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
# optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9,␣
 ↪weight_decay=1e-4)
```

[ ]:
```
# Define the learning rate scheduler
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
```

[ ]:
```
import os

def create_folder(folder_path):
    try:
        # Create the folder
        os.makedirs(folder_path)
        print(f"Folder '{folder_path}' created successfully.")
    except FileExistsError:
        print(f"Folder '{folder_path}' already exists.")

# Example usage:
folder_to_create = f'/content/drive/MyDrive/AIT/ISE/computer-vision/exam/report/
 ↪cloth_color/without_testset/
 ↪train-v4-ADAM-1200images-notestset-b3-{str(batch_size)}-report'

create_folder(folder_to_create)
```

```
Folder '/content/drive/MyDrive/AIT/ISE/computer-
vision/exam/report/cloth_color/without_testset/train-v4-ADAM-1200images-
notestset-b3-16-report' created successfully.
```

[ ]:
```
report_path = folder_to_create
```

[ ]:
```
model_name =␣
 ↪"torch-cloth-color-efficientnet-ADAM-b3-SGD-batch"+str(batch_size)+"-v4-23-nov-23"
# Training loop
num_epochs = 7
train_loss_list = []
valid_loss_list = []
train_accuracy_list = []
valid_accuracy_list = []


for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        images = images.to(device)
        labels = 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 = correct / total

    # Validation loop
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in valid_loader:
            images = images.to(device)
            labels = 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()

    valid_loss = running_loss / len(valid_loader)
    valid_accuracy = correct / total

    # Store loss and accuracy values
    train_loss_list.append(train_loss)
    valid_loss_list.append(valid_loss)
    train_accuracy_list.append(train_accuracy)
    valid_accuracy_list.append(valid_accuracy)

    # Print the training/validation statistics
    print(f"Epoch: {epoch+1}/{num_epochs} | "
          f"Train Loss: {train_loss:.4f} | Train Acc: {train_accuracy:.4f} | "
          f"Valid Loss: {valid_loss:.4f} | Valid Acc: {valid_accuracy:.4f}")


    # Update the learning rate
    scheduler.step()
```

```
# Save the trained model
torch.save(model.state_dict(), report_path+"/"+model_name+".pth")
```

```
Epoch: 1/7 | Train Loss: 0.6813 | Train Acc: 0.7855 | Valid Loss: 0.5994 | Valid
Acc: 0.8161
Epoch: 2/7 | Train Loss: 0.3750 | Train Acc: 0.8751 | Valid Loss: 0.5507 | Valid
Acc: 0.8349
Epoch: 3/7 | Train Loss: 0.2739 | Train Acc: 0.9082 | Valid Loss: 0.2941 | Valid
Acc: 0.9135
Epoch: 4/7 | Train Loss: 0.2325 | Train Acc: 0.9230 | Valid Loss: 0.3097 | Valid
Acc: 0.9005
Epoch: 5/7 | Train Loss: 0.1951 | Train Acc: 0.9354 | Valid Loss: 0.2610 | Valid
Acc: 0.9266
Epoch: 6/7 | Train Loss: 0.0836 | Train Acc: 0.9703 | Valid Loss: 0.1179 | Valid
Acc: 0.9635
Epoch: 7/7 | Train Loss: 0.0475 | Train Acc: 0.9855 | Valid Loss: 0.1096 | Valid
Acc: 0.9682
```

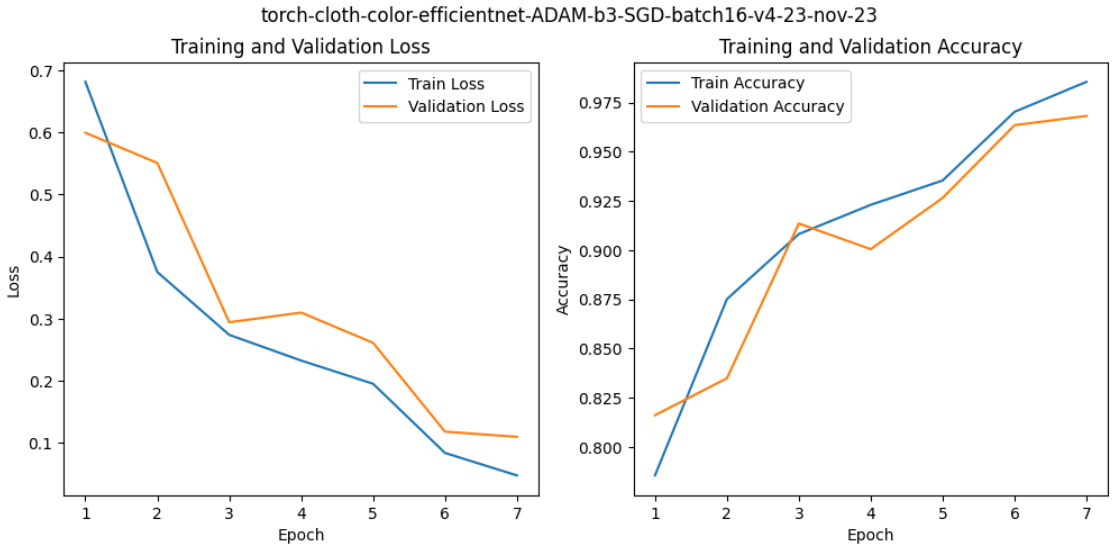
[ ]:
```
# Initialize lists to store true labels and predicted labels
true_labels = []
predicted_labels = []

# ...

# Open a file in write mode
file_path = report_path+"/"+model_name+".text"
file = open(file_path, "w")

# Validation loop
model.eval()
with torch.no_grad():
    for images, labels in valid_loader:
        images = images.to(device)
        labels = 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()

        # Append true labels and predicted labels
        true_labels.extend(labels.cpu().numpy())
        predicted_labels.extend(predicted.cpu().numpy())
```

```
valid_loss = running_loss / len(valid_loader)
valid_accuracy = correct / total

# ...

# Print the training/validation statistics
print(f"Epoch: {epoch+1}/{num_epochs} | "
      f"Train Loss: {train_loss:.4f} | Train Acc: {train_accuracy:.4f} | "
      f"Valid Loss: {valid_loss:.4f} | Valid Acc: {valid_accuracy:.4f}")

file.write(model_name+"\n")
file.write(str(criterion)+"\n")
file.write(str(optimizer)+"\n")
file.write(str(scheduler)+"\n")
file.write(str(model._dropout)+"\n")
# Write text to the file
file.write(f"Epoch: {epoch+1}/{num_epochs} | "
      f"Train Loss: {train_loss:.4f} | Train Acc: {train_accuracy:.4f} | "
      f"Valid Loss: {valid_loss:.4f} | Valid Acc: {valid_accuracy:.4f}"+"\n")

# Close the file
file.close()
```

```
Epoch: 7/7 | Train Loss: 0.0475 | Train Acc: 0.9855 | Valid Loss: 0.2192 | Valid
Acc: 0.9682
```

[ ]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
# Plot training and validation loss
plt.plot(range(1, num_epochs+1), train_loss_list, label='Train Loss')
plt.plot(range(1, num_epochs+1), valid_loss_list, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()

plt.subplot(1, 2, 2)
# Plot training and validation accuracy
plt.plot(range(1, num_epochs+1), train_accuracy_list, label='Train Accuracy')
plt.plot(range(1, num_epochs+1), valid_accuracy_list, label='Validation␣
 ↪Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
```

```
plt.suptitle(model_name)
plt.savefig(report_path+'/'+model_name+"_loss_acc.jpg")
plt.show()
```



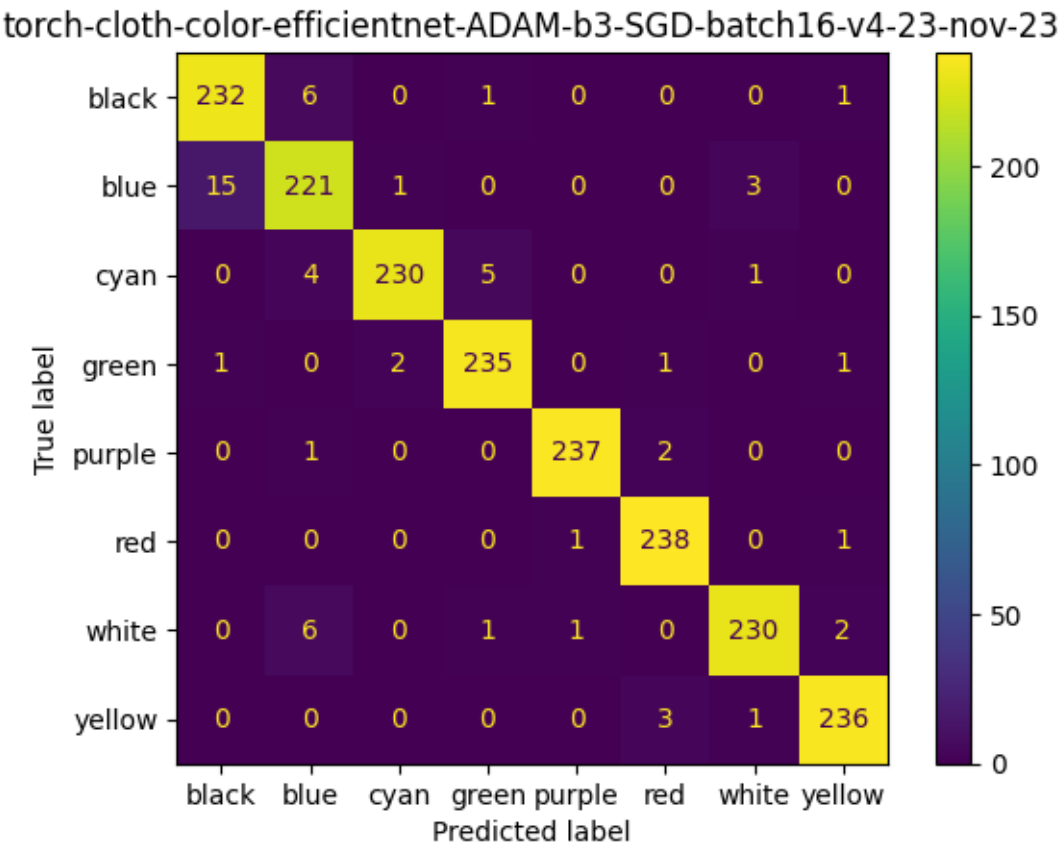
[ ]:
```
from sklearn.metrics import classification_report

# Generate classification report
file_path = report_path+"/"+model_name+"_f1_score.text"
file = open(file_path, "w")
classification_rep = classification_report(true_labels, predicted_labels)
print("Classification Report:")
print(classification_rep)

file.write("Classification Report:"+"\n")
file.write(classification_rep+"\n")

file.close()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       240
           1       0.93      0.92      0.92       240
           2       0.99      0.96      0.97       240
           3       0.97      0.98      0.98       240
           4       0.99      0.99      0.99       240
           5       0.98      0.99      0.98       240
           6       0.98      0.96      0.97       240
```

```
           7       0.98      0.98      0.98       240

    accuracy                           0.97      1920
   macro avg       0.97      0.97      0.97      1920
weighted avg       0.97      0.97      0.97      1920
```

[ ]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)
cm_display = ConfusionMatrixDisplay(confusion_matrix=cm,␣
 ↪display_labels=valid_dataset.classes)
cm_display.plot()

plt.title(model_name)

# Show the plot
plt.savefig(report_path+"/"+model_name+"_ConfusionMatrixDisplay.jpg")


plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```


torch-cloth-color-efficientnet-ADAM-b3-SGD-batch16-v4-23-nov-23
black 232 6 0 1 0 0 0 1
blue 15 221 1 0 0 0 3 0
cyan 0 4 230 5 0 0 1 0
green 1 0 2 235 0 1 0 1
purple 0 1 0 0 237 2 0 0
red 0 0 0 0 1 238 0 1
white 0 6 0 1 1 0 230 2
yellow 0 0 0 0 0 3 1 236
True label
black blue cyan green purple red white yellow
Predicted label
200
150
100
50
0