model_multipleVideos-class_wise_Implementation-EfficientNet-withLSTM

April 8, 2022

[1]:
```python
import torch
import torchvision
import torchvision.transforms as transforms
import torchvision.models as models
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import time
from itertools import count
import natsort
import datetime
import numpy as np
import os
import math
```

[2]:
```python
from torch.utils.data import Dataset, DataLoader, WeightedRandomSampler
import albumentations as A
from albumentations.pytorch import ToTensorV2
import cv2
import glob
import numpy
import random
import pandas as pd
import tqdm
torch.manual_seed(10)
```

[2]:
```
<torch._C.Generator at 0x18109185130>
```

[3]:
```python
print(f"Is CUDA supported by this system? {torch.cuda.is_available()}")
print(f"CUDA version: {torch.version.cuda}")
# Storing ID of current CUDA device
cuda_id = torch.cuda.current_device()
print(f"ID of current CUDA device: {torch.cuda.current_device()}")
print(f"Name of current CUDA device: {torch.cuda.get_device_name(cuda_id)}")

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

```
print(device)
```

```
Is CUDA supported by this system? True
CUDA version: 11.3
ID of current CUDA device: 0
Name of current CUDA device: NVIDIA GeForce RTX 2070 Super
cuda:0
```

1 Building the dataset

[4]:

```
class SurgicalDataset(Dataset):
    def __init__(self, image_paths, labels, transform=False):
        super(SurgicalDataset, self).__init__()
        self.image_paths = image_paths
        self.labels = labels    #.astype(dtype='int')
        self.transform = transform

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        image_filepath = self.image_paths[idx]
        image = cv2.imread(image_filepath)
        label = self.labels[idx]
        if self.transform is not None:
            image = self.transform(image=image)["image"]

        return image, label
```

[5]:

```
def get_transform(model_name):

    if model_name == 'alexnet':
        transform = A.Compose([
            A.Resize(227, 227),
            A.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
            ToTensorV2(),
        ])

    elif model_name == 'effinet':
        transform = A.Compose([
            A.Resize(224, 224),
            A.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
            ToTensorV2(),
        ])

    else:
        transform = A.Compose([
            A.Resize(224,224),
            A.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5)),
            ToTensorV2(),
        ])

    return transform
```

[6]:

```
# Preparing the datasets
# Get images
train_image_paths = []
train_data_path = r"C:
 ↪\Users\panji\EECS6691_Advanced_DL\Assignment2\training_data_images"
train_image_paths.append(glob.glob(train_data_path + '/*'))
# unpack the listed list
train_image_paths1 = [item for sublist in train_image_paths for item in sublist]
train_image_paths1 = natsort.natsorted(train_image_paths1)
print('len(train_image_paths1)', len(train_image_paths1))

# Get labels
df = pd.read_csv("Processed_data.csv")
df1 = df.loc[:,"Phases"].to_numpy()
df2 = df1.tolist()
print('len(df2)', len(df2))

# Preparing the datasets (images and labels)
dataset_train = pd.DataFrame(
    {'Link': train_image_paths1,
     'Label': df2,
    })
dataset_train1 = dataset_train.sample(frac=1, random_state=1)
train_image_paths = dataset_train1.loc[:,"Link"].to_numpy().tolist()
labels = dataset_train1.loc[:,"Label"].to_numpy().tolist()

# manually split the dataset
train_image_paths, valid_image_paths = train_image_paths[:int(0.
 ↪8*len(train_image_paths))], train_image_paths[int(0.
 ↪8*len(train_image_paths)):]
train_labels, valid_labels = labels[:int(0.8*len(labels))], labels[int(0.
 ↪8*len(labels)):]
print('train_labels', len(train_labels))
print('train_image_paths', len(train_image_paths))
print('train_labels', len(valid_labels))
print('train_image_paths', len(valid_image_paths))

print('label distribution in the training data', np.bincount(train_labels))
```

```
len(train_image_paths1) 215057
len(df2) 215057
train_labels 172045
train_image_paths 172045
train_labels 43012
train_image_paths 43012
label distribution in the training data [  243    73  2308 22305   952  1246
44928  8681 11596 22901   896 41140
  1789 12987]
```

2 Building the classifier class

[7]:

```
class Classifier():

    def __init__(self, name, model, dataloaders, parameter, use_cuda=False):

        '''
        @name: Experiment name. Will define stored results etc.
        @model: Any models
        @dataloaders: Dictionary with keys train, val and test and␣
 ↪corresponding dataloaders
        @class_names: list of classes, where the idx of class name corresponds␣
 ↪to the label used for it in the data
        @use_cuda: whether or not to use cuda
        '''

        self.name = name
        if use_cuda and not torch.cuda.is_available():
            raise Exception("Asked for CUDA but GPU not found")

        self.use_cuda = use_cuda
        self.epoch = parameter['epochs']
        self.lr = parameter['lr']
        self.batch_size = parameter['batch_size']

        self.model = model.to('cuda' if use_cuda else 'cpu') # model.to('cpu')
        self.criterion = nn.CrossEntropyLoss()
        self.optimizer = optim.Adam(self.model.parameters(), lr=self.lr)
        self.train_loader, self.valid_loader = self.
 ↪get_dataloaders(dataloaders['train_image_paths'],
                                                                              ␣
 ↪dataloaders['train_labels'],
                                                                              ␣
 ↪dataloaders['valid_image_paths'],
                                                                              ␣
 ↪dataloaders['valid_labels'],
                                                                   ␣
↪train_transforms=dataloaders['transforms'],
                                                                    batch_size␣
↪= self.batch_size,
                                                                   ␣
↪shuffle=parameter['shuffle'],
                                                                   sampler =␣
↪dataloaders['sampler'])
        self.class_names = parameter['class_names']

        self.activations_path = os.path.join('activations', self.name)
        self.kernel_path = os.path.join('kernel_viz', self.name)
        save_path = os.path.join(os.getcwd(), 'models', self.name)
        if not os.path.exists(save_path):
            os.makedirs(save_path)

        if not os.path.exists(self.activations_path):
            os.makedirs(self.activations_path)

        if not os.path.exists(self.kernel_path):
            os.makedirs(self.kernel_path)

        self.save_path = save_path

    def train(self, save=True):
        '''
        @epochs: number of epochs to train
        @save: whether or not to save the checkpoints
        '''
        best_val_accuracy = - math.inf

        for epoch in range(self.epoch):  # loop over the dataset multiple times
            self.model.train()
            t = time.time()
            running_loss = 0.0
            train_acc = 0
            val_accuracy = 0
            correct = 0
            total = 0
            count = 0
            loop = tqdm.tqdm(self.train_loader, total = len(self.train_loader),␣
↪leave = True)

            for img, label in loop:
                # get the inputs; data is a list of [inputs, labels]
                inputs, labels = img.to(device), label.to(device) #img.
↪to(device), label.to(device)
```

```
            # zero the parameter gradients
            self.optimizer.zero_grad()

            # forward + backward + optimize
            outputs = self.model(inputs)
            _, predictions = torch.max(outputs, 1)
            loss = self.criterion(outputs, labels)
            loss.backward()
            self.optimizer.step()

            # print statistics
            running_loss += loss.item()
            total += labels.shape[0]
            correct += (predictions == labels).sum().item()

            count += 1
            if count % 2000 == 1999:    # print every 2000 mini-batches
                print(f'[{epoch + 1}, {count + 1:5d}] loss: {running_loss /␣
↪2000:.3f}')
                running_loss = 0.0

        train_acc = 100 * correct / total
        print(f'Epoch:', epoch + 1, f'Training Epoch Accuracy:{train_acc}')

        # evaluate the validation dataset
        self.model.eval()
        correct_pred = {classname: 0 for classname in self.class_names}
        total_pred = {classname: 0 for classname in self.class_names}

        # again no gradients needed
        correct = 0
        total = 0
        with torch.no_grad():
            for data in self.valid_loader:
                images, labels = data[0].to(device), data[1].to(device)␣
↪#data[0], data[1]
                outputs = self.model(images)
                _, predictions = torch.max(outputs, 1)
                # collect the correct predictions for each class
                total += labels.shape[0]
                correct += (predictions == labels).sum().item()

                for label, prediction in zip(labels, predictions):
                    if label == prediction:
                        correct_pred[classes[label]] += 1
                    total_pred[classes[label]] += 1
            val_accuracy = 100 * correct / total
            print(f'Epoch:', epoch + 1, f'Validation Epoch Accuracy:
↪{val_accuracy}')

            # print the summary for each class
            print('Epoch:', epoch + 1, 'Correct predictions', correct_pred)
            print('Epoch:', epoch + 1, 'Total predictions', total_pred)
            print('Epoch:', epoch + 1, 'Correct predictions', correct_pred)
            print('Epoch:', epoch + 1, 'Total predictions', total_pred)

            # inspect the time taken to train one epoch
            d = time.time()-t
            print('Fininsh Trainig Epoch', epoch, '!', 'Time used:', d)

            if save:
                torch.save(self.model.state_dict(), os.path.join(self.
↪save_path, f'epoch_{epoch}.pt'))
                if val_accuracy > best_val_accuracy:
                    torch.save(self.model.state_dict(), os.path.join(self.
↪save_path, 'best.pt'))
                    best_val_accuracy = val_accuracy

        print('Done training!')


    def evaluate(self):
        # for evaluating the test dataset if there were any.
        try:
            assert os.path.exists(os.path.join(self.save_path, 'best.pt'))

        except:
            print('Please train first')
            return

        self.model.load_state_dict(torch.load(os.path.join(self.save_path,␣
↪'best.pt')))
        self.model.eval()


    def get_dataloaders(self, train_image_paths, train_labels,␣
↪valid_image_paths, valid_labels, train_transforms=False, batch_size=32,␣
↪shuffle=True, sampler = None):
        train_dataset = SurgicalDataset(train_image_paths,train_labels,␣
↪train_transforms)
```

```
        val_dataset = SurgicalDataset(valid_image_paths,valid_labels,␣
↪train_transforms)
        train_loader = DataLoader(train_dataset, batch_size, shuffle = False)
        valid_loader = DataLoader(val_dataset, batch_size, shuffle = False)

        return train_loader, valid_loader


    def grad_cam_on_input(self, img):

        try:
            assert os.path.exists(os.path.join(self.save_path, 'best.pt'))

        except:
            print('It appears you are testing the model without training.␣
↪Please train first')
            return

        self.model.load_state_dict(torch.load(os.path.join(self.save_path,␣
↪'best.pt')))


        self.model.eval()
        img = img.to('cuda' if self.use_cuda else 'cpu')


        out = self.model(img)

        _, pred = torch.max(out, 1)

        predicted_class = self.class_names[int(pred)]
        print(f'Predicted class was {predicted_class}')

        out[:, pred].backward()
        gradients = self.model.get_gradient_activations()

        print('Gradients shape: ', f'{gradients.shape}')

        mean_gradients = torch.mean(gradients, [0, 2, 3]).cpu()
        activations = self.model.get_final_conv_layer(img).detach().cpu()

        print('Activations shape: ', f'{activations.shape}')

        for idx in range(activations.shape[1]):
            activations[:, idx, :, :] *= mean_gradients[idx]

        final_heatmap = np.maximum(torch.mean(activations, dim=1).squeeze(), 0)
        final_heatmap /= torch.max(final_heatmap)

        return final_heatmap

    def trained_kernel_viz(self):

        all_layers = [0, 3]
        all_filters = []
        for layer in all_layers:

            filters = self.model.conv_model[layer].weight
            all_filters.append(filters.detach().cpu().clone()[:8, :8, :, :])

        for filter_idx in range(len(all_filters)):

            filter = all_filters[filter_idx]
            print(filter.shape)
            filter = filter.contiguous().view(-1, 1, filter.shape[2], filter.shape[3])
            image = show_img(make_grid(filter))
            image = 255 * image
            cv2.imwrite(os.path.join(self.kernel_path, f'filter_layer{all_layers[filter_idx]}.jpg'), image)


    def activations_on_input(self, img):

        img = img.to('cuda' if self.use_cuda else 'cpu')

        all_layers = [0,3,6,8,10]
        all_viz = []

        # looking at the outputs of the relu
        for each in all_layers:

            current_model = self.model.conv_model[:each+1]
            current_out = current_model(img)
            all_viz.append(current_out.detach().cpu().clone()[:, :64, :, :])

        for viz_idx in range(len(all_viz)):

            viz = all_viz[viz_idx]
            viz = viz.view(-1, 1, viz.shape[2], viz.shape[3])
            image = show_img(make_grid(viz))
            image = 255 * image
            cv2.imwrite(os.path.join(self.activations_path,␣
↪f'sample_layer{all_layers[viz_idx]}.jpg'), image)
```

3 Build and train models

[8]:
```
example_model = models.efficientnet_b7(pretrained=True)
print(example_model)
```

```
EfficientNet(
  (features): Sequential(
    (0): ConvNormActivation(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (1): BatchNorm2d(64, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=64, bias=False)
            (1): BatchNorm2d(64, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(64, 16, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(16, 64, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (2): ConvNormActivation(
            (0): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.0, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=32, bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.0036363636363636364, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=32, bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.007272727272727273, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=32, bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(32, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.01090909090909091, mode=row)
      )
    )
    (2): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(192, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), groups=192, bias=False)
            (1): BatchNorm2d(192, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(192, 8, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(8, 192, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(192, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
```

```
          )
        )
        (stochastic_depth): StochasticDepth(p=0.014545454545454545, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.01818181818181818, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.02181818181818182, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.025454545454545455, mode=row)
      )
      (4): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.02909090909090909, mode=row)
      )
      (5): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.03272727272727273, mode=row)
      )
      (6): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(48, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.03636363636363636, mode=row)
      )
    )
    (3): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(48, 288, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
```

```
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(288, 288, kernel_size=(5, 5), stride=(2, 2), padding=(2,
2), groups=288, bias=False)
            (1): BatchNorm2d(288, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(288, 12, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(12, 288, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(288, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.04, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.04363636363636364, mode=row)
   )
   (2): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
      (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
      (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.04727272727272727, mode=row)
   )
   (3): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
      (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.05090909090909091, mode=row)
      )
      (4): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.05454545454545454, mode=row)
      )
      (5): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.05818181818181818, mode=row)
      )
      (6): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.06181818181818183, mode=row)
      )
    )
    (4): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), groups=480, bias=False)
            (1): BatchNorm2d(480, eps=0.001, momentum=0.01, 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): ConvNormActivation(
            (0): Conv2d(480, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.06545454545454546, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.06909090909090909, mode=row)
   )
   (2): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
```

```
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.07272727272727272, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.07636363636363637, mode=row)
      )
      (4): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.08, mode=row)
      )
      (5): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.08363636363636365, mode=row)
      )
      (6): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.08727272727272728, mode=row)
      )
      (7): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.09090909090909091, mode=row)
      )
      (8): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.09454545454545454, mode=row)
      )
      (9): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.09818181818181819, mode=row)
      )
    )
    (5): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(960, 960, kernel_size=(5, 5), stride=(1, 1), padding=(2,
2), groups=960, bias=False)
            (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(960, 40, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(40, 960, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(960, 224, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.10181818181818182, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.10545454545454547, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.10909090909090909, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.11272727272727273, mode=row)
      )
      (4): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.11636363636363636, mode=row)
      )
      (5): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.12000000000000001, mode=row)
      )
      (6): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.12363636363636366, mode=row)
      )
      (7): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.12727272727272726, mode=row)
      )
      (8): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.13090909090909092, mode=row)
      )
      (9): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 224, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(224, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.13454545454545455, mode=row)
      )
    )
    (6): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(224, 1344, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(1344, 1344, kernel_size=(5, 5), stride=(2, 2),
padding=(2, 2), groups=1344, bias=False)
            (1): BatchNorm2d(1344, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(1344, 56, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(56, 1344, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(1344, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.13818181818181818, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.14181818181818184, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.14545454545454545, mode=row)
   )
   (3): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
      (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.1490909090909091, mode=row)
   )
   (4): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.15272727272727274, mode=row)
      )
      (5): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.15636363636363634, mode=row)
   )
   (6): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
      (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.16, mode=row)
   )
   (7): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.16363636363636364, mode=row)
      )
      (8): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.1672727272727273, mode=row)
   )
   (9): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (1): ConvNormActivation(
      (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
      (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
     )
     (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
     )
     (3): ConvNormActivation(
      (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
     )
    )
    (stochastic_depth): StochasticDepth(p=0.17090909090909093, mode=row)
   )
   (10): MBConv(
    (block): Sequential(
     (0): ConvNormActivation(
      (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.17454545454545456, mode=row)
      )
      (11): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.1781818181818182, mode=row)
      )
      (12): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(384, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.18181818181818182, mode=row)
      )
    )
    (7): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(384, 2304, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(2304, 2304, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=2304, bias=False)
            (1): BatchNorm2d(2304, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(2304, 96, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(96, 2304, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(2304, 640, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(640, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.18545454545454548, mode=row)
      )
      (1): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(640, 3840, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(3840, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(3840, 3840, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=3840, bias=False)
            (1): BatchNorm2d(3840, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(3840, 160, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(160, 3840, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(3840, 640, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(640, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.1890909090909091, mode=row)
      )
      (2): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(640, 3840, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(3840, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(3840, 3840, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=3840, bias=False)
            (1): BatchNorm2d(3840, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(3840, 160, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(160, 3840, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(3840, 640, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(640, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.19272727272727275, mode=row)
      )
      (3): MBConv(
        (block): Sequential(
          (0): ConvNormActivation(
            (0): Conv2d(640, 3840, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(3840, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): ConvNormActivation(
            (0): Conv2d(3840, 3840, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=3840, bias=False)
            (1): BatchNorm2d(3840, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (2): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(3840, 160, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(160, 3840, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (3): ConvNormActivation(
            (0): Conv2d(3840, 640, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (1): BatchNorm2d(640, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.19636363636363638, mode=row)
      )
    )
    (8): ConvNormActivation(
      (0): Conv2d(640, 2560, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(2560, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True)
      (2): SiLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=1)
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=True)
    (1): Linear(in_features=2560, out_features=1000, bias=True)
  )
)
```

[9]:

```
print(example_model.classifier)
print(example_model.avgpool)
```

```
Sequential(
  (0): Dropout(p=0.5, inplace=True)
  (1): Linear(in_features=2560, out_features=1000, bias=True)
)
AdaptiveAvgPool2d(output_size=1)
```

[10]:

```python
class TransferEffiNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.base_effi_net = models.efficientnet_b7(pretrained=True)
        self.conv_model = self.get_conv_layers()
        self.avg_pool = self.transition_layer()
        self.activate_training_layers()

        self.dropout = nn.Dropout(p=0.5, inplace=False)
        self.fc1 = nn.Linear(in_features=2560, out_features=1024, bias=True)
        self.bn1 = nn.BatchNorm1d(1024)
        #nn.Dropout(p=0.5, inplace=False)
        self.fc2 = nn.Linear(in_features=1024, out_features=512, bias=True)
        self.bn2 = nn.BatchNorm1d(512)
        self.lstm = nn.LSTM(512, 128)
        self.fc3 = nn.Linear(in_features=128, out_features=14, bias=True)

    def activate_training_layers(self):
        for name, param in self.conv_model.named_parameters():
            number = int(name.split('.')[1])
            # for all layers except the last conv layer, set param.
↪requires_grad = False
            if number == 8:
                param.requires_grad = True
            else:
                param.requires_grad = False

    def get_conv_layers(self):
        return self.base_effi_net.features

    def transition_layer(self):
        return self.base_effi_net.avgpool

    def forward(self, x):
        x = self.conv_model(x)   #call the conv layers
        x = self.avg_pool(x)  #call the avg pool layer
        x = torch.flatten(x, 1)
        # start of the classifier
        x = self.dropout(x)
        x = self.fc1(x)
        x = F.relu(self.bn1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        x = F.relu(self.bn2(x))
        x, _ = self.lstm(x)
        x = self.fc3(x)
        return x
```

[11]:

```
torch.cuda.empty_cache()
```

[12]:

```
# Train a transfer learning model with Alexnet
name = 'TransferEffiNet-LSTM'
classes = [i for i in range(15)]
transforms = get_transform('effinet')
dataloaders = {'train_image_paths': train_image_paths, 'train_labels' :␣
 ↪train_labels, 'valid_image_paths': valid_image_paths, 'valid_labels':
 ↪valid_labels, 'transforms':transforms, 'sampler':None}
parameters = {'lr': 0.0001, 'epochs' : 10, 'batch_size':32, 'shuffle':False,␣
 ↪'class_names':classes}

model = TransferEffiNet()
classifier = Classifier(name, model, dataloaders, parameters, use_cuda=True)
classifier.train()
```

```
 37%|
| 1999/5377 [22:11<2:05:18,  2.23s/it]

[1,  2000] loss: 0.925

 74%|
| 3999/5377 [47:32<14:51,  1.55it/s]

[1,  4000] loss: 0.585

100%|
 | 5377/5377 [1:04:05<00:00,  1.40it/s]

Epoch: 1 Training Epoch Accuracy:78.0104042547008
Epoch: 1 Validation Epoch Accuracy:89.45178089835395
Epoch: 1 Correct predictions {0: 0, 1: 0, 2: 496, 3: 5242, 4: 17, 5: 272, 6:
10901, 7: 2028, 8: 1934, 9: 5185, 10: 124, 11: 9980, 12: 184, 13: 2112, 14: 0}
Epoch: 1 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 1 Correct predictions {0: 0, 1: 0, 2: 496, 3: 5242, 4: 17, 5: 272, 6:
10901, 7: 2028, 8: 1934, 9: 5185, 10: 124, 11: 9980, 12: 184, 13: 2112, 14: 0}
Epoch: 1 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 0 ! Time used: 4571.735313653946

 37%|
| 1999/5377 [17:59<29:24,  1.91it/s]
```

```
[2,  2000] loss: 0.415

 74%|
| 3999/5377 [35:48<11:59,  1.92it/s]

[2,  4000] loss: 0.373

100%|
  | 5377/5377 [47:48<00:00,  1.87it/s]

Epoch: 2 Training Epoch Accuracy:87.53175041413584
Epoch: 2 Validation Epoch Accuracy:92.88338138193993
Epoch: 2 Correct predictions {0: 1, 1: 0, 2: 520, 3: 5295, 4: 87, 5: 279, 6:
10916, 7: 2074, 8: 2221, 9: 5450, 10: 235, 11: 10164, 12: 388, 13: 2321, 14: 0}
Epoch: 2 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 2 Correct predictions {0: 1, 1: 0, 2: 520, 3: 5295, 4: 87, 5: 279, 6:
10916, 7: 2074, 8: 2221, 9: 5450, 10: 235, 11: 10164, 12: 388, 13: 2321, 14: 0}
Epoch: 2 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 1 ! Time used: 3385.8801736831665

 37%|
| 1999/5377 [17:18<29:05,  1.93it/s]

[3,  2000] loss: 0.294

 74%|
| 3999/5377 [34:37<12:05,  1.90it/s]

[3,  4000] loss: 0.273

100%|
  | 5377/5377 [46:31<00:00,  1.93it/s]

Epoch: 3 Training Epoch Accuracy:90.85994943183469
Epoch: 3 Validation Epoch Accuracy:94.20394308565052
Epoch: 3 Correct predictions {0: 22, 1: 0, 2: 545, 3: 5276, 4: 97, 5: 279, 6:
10978, 7: 2105, 8: 2400, 9: 5540, 10: 241, 11: 10242, 12: 388, 13: 2406, 14: 0}
Epoch: 3 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 3 Correct predictions {0: 22, 1: 0, 2: 545, 3: 5276, 4: 97, 5: 279, 6:
10978, 7: 2105, 8: 2400, 9: 5540, 10: 241, 11: 10242, 12: 388, 13: 2406, 14: 0}
Epoch: 3 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 2 ! Time used: 3304.413810968399

 37%|
| 1999/5377 [17:23<29:12,  1.93it/s]

[4,  2000] loss: 0.227

 74%|
| 3999/5377 [34:44<11:51,  1.94it/s]
```

```
[4,  4000] loss: 0.217

100%|
  | 5377/5377 [46:44<00:00,  1.92it/s]

Epoch: 4 Training Epoch Accuracy:92.89371966636635
Epoch: 4 Validation Epoch Accuracy:94.9432716451223
Epoch: 4 Correct predictions {0: 37, 1: 1, 2: 541, 3: 5309, 4: 110, 5: 279, 6:
10915, 7: 2085, 8: 2517, 9: 5580, 10: 243, 11: 10263, 12: 396, 13: 2561, 14: 0}
Epoch: 4 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 4 Correct predictions {0: 37, 1: 1, 2: 541, 3: 5309, 4: 110, 5: 279, 6:
10915, 7: 2085, 8: 2517, 9: 5580, 10: 243, 11: 10263, 12: 396, 13: 2561, 14: 0}
Epoch: 4 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 3 ! Time used: 3321.863705635071

 37%|
| 1999/5377 [17:26<29:54,  1.88it/s]

[5,  2000] loss: 0.182

 74%|
| 3999/5377 [34:46<11:55,  1.93it/s]

[5,  4000] loss: 0.175

100%|
  | 5377/5377 [46:58<00:00,  1.91it/s]

Epoch: 5 Training Epoch Accuracy:94.18989217937168
Epoch: 5 Validation Epoch Accuracy:95.47103134009113
Epoch: 5 Correct predictions {0: 43, 1: 7, 2: 554, 3: 5335, 4: 110, 5: 279, 6:
10966, 7: 2113, 8: 2501, 9: 5629, 10: 242, 11: 10272, 12: 393, 13: 2620, 14: 0}
Epoch: 5 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 5 Correct predictions {0: 43, 1: 7, 2: 554, 3: 5335, 4: 110, 5: 279, 6:
10966, 7: 2113, 8: 2501, 9: 5629, 10: 242, 11: 10272, 12: 393, 13: 2620, 14: 0}
Epoch: 5 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 4 ! Time used: 3344.631583213806

 37%|
| 1999/5377 [17:23<29:14,  1.93it/s]

[6,  2000] loss: 0.151

 74%|
| 3999/5377 [34:47<12:00,  1.91it/s]

[6,  4000] loss: 0.149

100%|
  | 5377/5377 [46:50<00:00,  1.91it/s]
```

```
Epoch: 6 Training Epoch Accuracy:95.13266877851724
Epoch: 6 Validation Epoch Accuracy:95.66632567655537
Epoch: 6 Correct predictions {0: 41, 1: 4, 2: 547, 3: 5333, 4: 117, 5: 281, 6:
11006, 7: 2132, 8: 2502, 9: 5631, 10: 243, 11: 10207, 12: 395, 13: 2709, 14: 0}
Epoch: 6 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 6 Correct predictions {0: 41, 1: 4, 2: 547, 3: 5333, 4: 117, 5: 281, 6:
11006, 7: 2132, 8: 2502, 9: 5631, 10: 243, 11: 10207, 12: 395, 13: 2709, 14: 0}
Epoch: 6 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 5 ! Time used: 3332.6721127033234

 37%|
| 1999/5377 [18:52<33:17,  1.69it/s]

[7,  2000] loss: 0.129

 74%|
| 3999/5377 [36:36<11:50,  1.94it/s]

[7,  4000] loss: 0.127

100%|
  | 5377/5377 [48:34<00:00,  1.85it/s]

Epoch: 7 Training Epoch Accuracy:95.8266732540905
Epoch: 7 Validation Epoch Accuracy:96.07318887752255
Epoch: 7 Correct predictions {0: 44, 1: 11, 2: 545, 3: 5352, 4: 153, 5: 281, 6:
10993, 7: 2108, 8: 2548, 9: 5687, 10: 242, 11: 10256, 12: 399, 13: 2704, 14: 0}
Epoch: 7 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 7 Correct predictions {0: 44, 1: 11, 2: 545, 3: 5352, 4: 153, 5: 281, 6:
10993, 7: 2108, 8: 2548, 9: 5687, 10: 242, 11: 10256, 12: 399, 13: 2704, 14: 0}
Epoch: 7 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 6 ! Time used: 3430.037877559662

 37%|
| 1999/5377 [17:25<29:09,  1.93it/s]

[8,  2000] loss: 0.112

 74%|
| 3999/5377 [34:51<11:58,  1.92it/s]

[8,  4000] loss: 0.112

100%|
  | 5377/5377 [46:56<00:00,  1.91it/s]

Epoch: 8 Training Epoch Accuracy:96.34281728617513
Epoch: 8 Validation Epoch Accuracy:96.049939551753
Epoch: 8 Correct predictions {0: 45, 1: 12, 2: 555, 3: 5364, 4: 134, 5: 283, 6:
11046, 7: 2145, 8: 2527, 9: 5617, 10: 245, 11: 10261, 12: 399, 13: 2680, 14: 0}
```

```
Epoch: 8 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 8 Correct predictions {0: 45, 1: 12, 2: 555, 3: 5364, 4: 134, 5: 283, 6:
11046, 7: 2145, 8: 2527, 9: 5617, 10: 245, 11: 10261, 12: 399, 13: 2680, 14: 0}
Epoch: 8 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 7 ! Time used: 3331.0325458049774

 37%|
| 1999/5377 [17:21<29:27,  1.91it/s]

[9,  2000] loss: 0.098

 74%|
| 3999/5377 [34:45<11:51,  1.94it/s]

[9,  4000] loss: 0.100

100%|
  | 5377/5377 [46:43<00:00,  1.92it/s]

Epoch: 9 Training Epoch Accuracy:96.73631898631172
Epoch: 9 Validation Epoch Accuracy:96.13596205710034
Epoch: 9 Correct predictions {0: 47, 1: 16, 2: 555, 3: 5321, 4: 153, 5: 279, 6:
10951, 7: 2121, 8: 2608, 9: 5620, 10: 242, 11: 10317, 12: 396, 13: 2724, 14: 0}
Epoch: 9 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 9 Correct predictions {0: 47, 1: 16, 2: 555, 3: 5321, 4: 153, 5: 279, 6:
10951, 7: 2121, 8: 2608, 9: 5620, 10: 242, 11: 10317, 12: 396, 13: 2724, 14: 0}
Epoch: 9 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 8 ! Time used: 3322.9786179065704

 37%|
| 1999/5377 [17:30<30:36,  1.84it/s]

[10,  2000] loss: 0.089

 74%|
| 3999/5377 [35:06<11:55,  1.93it/s]

[10,  4000] loss: 0.091

100%|
  | 5377/5377 [47:06<00:00,  1.90it/s]

Epoch: 10 Training Epoch Accuracy:96.98276613676654
Epoch: 10 Validation Epoch Accuracy:96.31963173067982
Epoch: 10 Correct predictions {0: 47, 1: 16, 2: 556, 3: 5360, 4: 132, 5: 279, 6:
11005, 7: 2131, 8: 2656, 9: 5682, 10: 241, 11: 10253, 12: 398, 13: 2673, 14: 0}
Epoch: 10 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Epoch: 10 Correct predictions {0: 47, 1: 16, 2: 556, 3: 5360, 4: 132, 5: 279, 6:
11005, 7: 2131, 8: 2656, 9: 5682, 10: 241, 11: 10253, 12: 398, 13: 2673, 14: 0}
```

```
Epoch: 10 Total predictions {0: 52, 1: 18, 2: 587, 3: 5470, 4: 213, 5: 288, 6:
11135, 7: 2177, 8: 3001, 9: 5799, 10: 247, 11: 10405, 12: 404, 13: 3216, 14: 0}
Fininsh Trainig Epoch 9 ! Time used: 3344.4643886089325
Done training!
```

[13]:

```
# !pip install GPUtil

# import torch
# from GPUtil import showUtilization as gpu_usage
# from numba import cuda

# def free_gpu_cache():
#     print("Initial GPU Usage")
#     gpu_usage()

#     torch.cuda.empty_cache()

#     cuda.select_device(0)
#     cuda.close()
#     cuda.select_device(0)

#     print("GPU Usage after emptying the cache")
#     gpu_usage()

# free_gpu_cache()
```

4 Data Augmentations for scarse data

[14]:

```
# find out the ratio of different labels/data

# we decide to augment the scarser data in the following scheme: label 0, +␣
 ↪400, label 14 + 500, label 6 + 200
```