import libraries

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import torch.backends.cudnn as cudnn
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import numpy as np
import os, sys
import time
import copy
from glob import glob
import imageio
```

Download Intel Image Dataset using Kaggle api

```
!pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages (1.5
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (fr
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (f
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages (fr
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages (
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-package
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-pa
```

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d puneet6060/intel-image-classification
```

```
Downloading intel-image-classification.zip to /content
100% 346M/346M [00:16<00:00, 20.9MB/s]
100% 346M/346M [00:16<00:00, 22.1MB/s]
```

1s completed at 9:32 PM

Data directory set

```
!unzip intel-image-classification.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: seg_train/seg_train/mountain/7506.jpg
  inflating: seg_train/seg_train/mountain/7537.jpg
  inflating: seg_train/seg_train/mountain/7539.jpg
  inflating: seg_train/seg_train/mountain/7551.jpg
  inflating: seg_train/seg_train/mountain/7560.jpg
  inflating: seg_train/seg_train/mountain/7565.jpg
  inflating: seg_train/seg_train/mountain/7578.jpg
  inflating: seg_train/seg_train/mountain/7581.jpg
  inflating: seg_train/seg_train/mountain/7586.jpg
  inflating: seg_train/seg_train/mountain/7647.jpg
  inflating: seg_train/seg_train/mountain/7652.jpg
  inflating: seg_train/seg_train/mountain/7654.jpg
  inflating: seg_train/seg_train/mountain/7662.jpg
  inflating: seg_train/seg_train/mountain/767.jpg
  inflating: seg_train/seg_train/mountain/7672.jpg
  inflating: seg_train/seg_train/mountain/7679.jpg
  inflating: seg_train/seg_train/mountain/7681.jpg
  inflating: seg_train/seg_train/mountain/7693.jpg
  inflating: seg_train/seg_train/mountain/7695.jpg
  inflating: seg_train/seg_train/mountain/7698.jpg
  inflating: seg_train/seg_train/mountain/7700.jpg
  inflating: seg_train/seg_train/mountain/771.jpg
  inflating: seg_train/seg_train/mountain/7715.jpg
  inflating: seg_train/seg_train/mountain/7744.jpg
  inflating: seg_train/seg_train/mountain/7745.jpg
  inflating: seg_train/seg_train/mountain/7751.jpg
  inflating: seg_train/seg_train/mountain/7763.jpg
  inflating: seg_train/seg_train/mountain/7771.jpg
  inflating: seg_train/seg_train/mountain/7780.jpg
  inflating: seg_train/seg_train/mountain/7787.jpg
  inflating: seg_train/seg_train/mountain/7788.jpg
  inflating: seg_train/seg_train/mountain/7813.jpg
  inflating: seg_train/seg_train/mountain/7816.jpg
  inflating: seg_train/seg_train/mountain/7819.jpg
  inflating: seg_train/seg_train/mountain/7820.jpg
  inflating: seg_train/seg_train/mountain/7823.jpg
  inflating: seg_train/seg_train/mountain/7836.jpg
  inflating: seg_train/seg_train/mountain/784.jpg
  inflating: seg_train/seg_train/mountain/7841.jpg
  inflating: seg_train/seg_train/mountain/7842.jpg
  inflating: seg_train/seg_train/mountain/7845.jpg
  inflating: seg_train/seg_train/mountain/7849.jpg
  inflating: seg_train/seg_train/mountain/7851.jpg
  inflating: seg_train/seg_train/mountain/7865.jpg
  inflating: seg_train/seg_train/mountain/7875.jpg
```

```
    inflating: seg_train/seg_train/mountain/7881.jpg
    inflating: seg_train/seg_train/mountain/7885.jpg
    inflating: seg_train/seg_train/mountain/790.jpg
    inflating: seg_train/seg_train/mountain/7908.jpg
    inflating: seg_train/seg_train/mountain/7909.jpg
    inflating: seg_train/seg_train/mountain/7912.jpg
    inflating: seg_train/seg_train/mountain/7922.jpg
    inflating: seg_train/seg_train/mountain/7928.jpg
    inflating: seg_train/seg_train/mountain/7942.jpg
    inflating: seg_train/seg_train/mountain/7946.jpg
    inflating: seg_train/seg_train/mountain/7960.jpg
    inflating: seg_train/seg_train/mountain/7973.jpg
```

```
train = '/content/seg_train/seg_train'
test = '/content/seg_test/seg_test'
```

Data Augmentation

```
# Note: normalize mean and std are standardized for ImageNet

data_transforms = {
    train: transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(0.5),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    test: transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

Load data

```
image_datasets = {x: datasets.ImageFolder(os.path.join(x), data_transforms[x]) for x in [t

dataloader = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=32, shuffle=Tru

dataset_sizes = {x: len(image_datasets[x]) for  x in [train,test]}
```

```
class_names = image_datasets[train].classes

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

```
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarni
  warnings.warn(_create_warning_msg(
```

```
resnet = models.resnet152(pretrained=True)
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:208: UserWarning
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:223: UserWarning
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet152-394f9c45.pth" to /root/.
100%                                   230M/230M [00:00<00:00, 272MB/s]
```

Plot intel image dataset

```
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  # pause a bit so that plots are updated


# Get a batch of training data
inputs, classes = next(iter(dataloader[train]))

# Make a grid from batch
out = torchvision.utils.make_grid(inputs)

imshow(out, title=[class_names[x] for x in classes])
```

```
print(image_datasets[train])
```

```
Dataset ImageFolder
    Number of datapoints: 14034
    Root location: /content/seg_train/seg_train
    StandardTransform
Transform: Compose(
               RandomResizedCrop(size=(224, 224), scale=(0.08, 1.0), ratio=(0.75, 1.
               RandomHorizontalFlip(p=0.5)
               ToTensor()
               Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
           )
```

Freeze CNN part of ResNet152 Model

```
for param in resnet.parameters():
    param.requires_grad = False
```

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

Set fully connected sequential

```
resnet.fc = nn.Sequential(nn.Linear(2048, 1024),
                          nn.ReLU(),
                          nn.Dropout(0.25),
                          nn.Linear(1024, 256),
                          nn.ReLU(),
                          nn.Dropout(0.25),
                          nn.Linear(256, 6),
                          nn.LogSoftmax(dim=1))

resnet = resnet.to(device)
```

Loss function/ Optimizer

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(resnet.parameters(), lr=1e-4)
exp_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
```

Train Model Function

```
from datetime import datetime

def train_model(model, criterion, optimizer, train_loader, test_loader, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)

    for it in range(epochs):
        if it == 10 :
          for param in model.parameters():
           param.requires_grad = True
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in dataloader[train]:
            inputs, targets = inputs.to(device), targets.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            loss.backward()
            optimizer.step()
            train_loss.append(loss.item())

        train_loss = np.mean(train_loss)

        test_loss = []
        for inputs, targets in dataloader[test]:
            inputs, targets = inputs.to(device), targets.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            test_loss.append(loss.item())

        test_loss = np.mean(test_loss)

        train_losses[it] = train_loss
        test_losses[it] = test_loss

        dt = datetime.now() - t0

        print(f'Epoch {it+1}/{epochs}, Train_Loss: {train_loss:.4f}, \Test_Loss: {test_los

    return train_losses, test_losses

train_losses, test_losses = train_model(
                            resnet,
                            criterion.
```

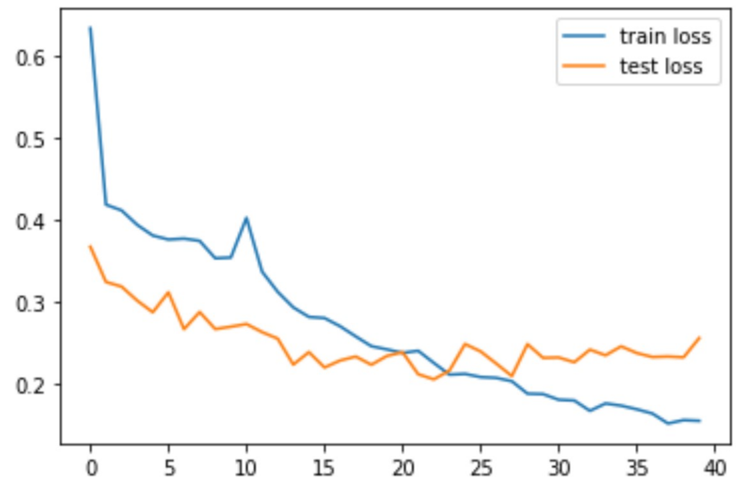
```
                       optimizer,
                       dataloader[train],
                       dataloader[test],
                       epochs=40)
```

```
Epoch 1/40, Train_Loss: 0.6348, \Test_Loss: 0.3671, Duration: 0:02:18.012346
Epoch 2/40, Train_Loss: 0.4187, \Test_Loss: 0.3239, Duration: 0:02:12.348367
Epoch 3/40, Train_Loss: 0.4116, \Test_Loss: 0.3183, Duration: 0:02:12.303492
Epoch 4/40, Train_Loss: 0.3940, \Test_Loss: 0.3011, Duration: 0:02:11.797530
Epoch 5/40, Train_Loss: 0.3809, \Test_Loss: 0.2869, Duration: 0:02:12.182279
Epoch 6/40, Train_Loss: 0.3760, \Test_Loss: 0.3113, Duration: 0:02:12.008638
Epoch 7/40, Train_Loss: 0.3771, \Test_Loss: 0.2661, Duration: 0:02:12.230927
Epoch 8/40, Train_Loss: 0.3743, \Test_Loss: 0.2872, Duration: 0:02:12.286634
Epoch 9/40, Train_Loss: 0.3530, \Test_Loss: 0.2662, Duration: 0:02:13.266736
Epoch 10/40, Train_Loss: 0.3539, \Test_Loss: 0.2692, Duration: 0:02:12.489712
Epoch 11/40, Train_Loss: 0.4025, \Test_Loss: 0.2725, Duration: 0:06:02.206394
Epoch 12/40, Train_Loss: 0.3366, \Test_Loss: 0.2625, Duration: 0:06:00.994119
Epoch 13/40, Train_Loss: 0.3119, \Test_Loss: 0.2546, Duration: 0:06:00.693115
Epoch 14/40, Train_Loss: 0.2928, \Test_Loss: 0.2227, Duration: 0:06:01.171837
Epoch 15/40, Train_Loss: 0.2811, \Test_Loss: 0.2380, Duration: 0:06:00.407107
Epoch 16/40, Train_Loss: 0.2797, \Test_Loss: 0.2192, Duration: 0:06:00.845971
Epoch 17/40, Train_Loss: 0.2697, \Test_Loss: 0.2277, Duration: 0:06:00.459688
Epoch 18/40, Train_Loss: 0.2573, \Test_Loss: 0.2326, Duration: 0:06:00.712149
Epoch 19/40, Train_Loss: 0.2453, \Test_Loss: 0.2225, Duration: 0:06:00.095190
Epoch 20/40, Train_Loss: 0.2413, \Test_Loss: 0.2334, Duration: 0:06:00.221754
Epoch 21/40, Train_Loss: 0.2371, \Test_Loss: 0.2379, Duration: 0:05:59.689844
Epoch 22/40, Train_Loss: 0.2397, \Test_Loss: 0.2109, Duration: 0:05:59.977015
Epoch 23/40, Train_Loss: 0.2248, \Test_Loss: 0.2046, Duration: 0:05:59.677062
Epoch 24/40, Train_Loss: 0.2104, \Test_Loss: 0.2154, Duration: 0:05:59.551734
Epoch 25/40, Train_Loss: 0.2113, \Test_Loss: 0.2479, Duration: 0:05:59.383888
Epoch 26/40, Train_Loss: 0.2074, \Test_Loss: 0.2388, Duration: 0:05:59.620912
Epoch 27/40, Train_Loss: 0.2065, \Test_Loss: 0.2240, Duration: 0:05:59.554906
Epoch 28/40, Train_Loss: 0.2023, \Test_Loss: 0.2087, Duration: 0:05:57.515687
Epoch 29/40, Train_Loss: 0.1871, \Test_Loss: 0.2478, Duration: 0:05:55.495677
Epoch 30/40, Train_Loss: 0.1866, \Test_Loss: 0.2309, Duration: 0:05:55.362354
Epoch 31/40, Train_Loss: 0.1797, \Test_Loss: 0.2314, Duration: 0:05:55.285163
Epoch 32/40, Train_Loss: 0.1787, \Test_Loss: 0.2256, Duration: 0:05:54.940069
Epoch 33/40, Train_Loss: 0.1660, \Test_Loss: 0.2410, Duration: 0:05:56.998170
Epoch 34/40, Train_Loss: 0.1751, \Test_Loss: 0.2341, Duration: 0:05:55.879126
Epoch 35/40, Train_Loss: 0.1724, \Test_Loss: 0.2451, Duration: 0:05:55.100177
Epoch 36/40, Train_Loss: 0.1680, \Test_Loss: 0.2369, Duration: 0:05:54.989463
Epoch 37/40, Train_Loss: 0.1628, \Test_Loss: 0.2321, Duration: 0:05:55.205754
Epoch 38/40, Train_Loss: 0.1506, \Test_Loss: 0.2326, Duration: 0:05:54.941748
Epoch 39/40, Train_Loss: 0.1547, \Test_Loss: 0.2317, Duration: 0:05:54.979370
Epoch 40/40, Train_Loss: 0.1539, \Test_Loss: 0.2552, Duration: 0:05:54.774343
```

Plot Train/Test loss

```
plt.plot(train_losses, label='train loss')
plt.plot(test_losses, label='test loss')
plt.legend()
```

```
plt.legend()
plt.show()
```



Accuracy on Test data

```
from tqdm.autonotebook import tqdm
```

```
train_losses=[]
test_losses=[]
def accuracy(loader, model):
    num_corrects = 0
    num_samples = 0
    model.eval()
    loop = tqdm(loader)
    with torch.no_grad():
        for x, y in loop:
            x = x.to(device)
            y = y.to(device)
            scores = model(x)
            test_losses.append(scores.data)
            _, prediction = scores.max(1)
            num_corrects += (prediction == y).sum()
            num_samples += prediction.size(0)
            acc = (num_corrects/num_samples) * 100
            loop.set_postfix(acc=acc.item())
        print(f'Got {num_corrects}/{num_samples} with accuracy {acc:.4f}')
```

```
accuracy(dataloader[test], resnet)
```

```
100%                                          94/94 [00:21<00:00, 4.93it/s, acc=93.1]
Got 2793/3000 with accuracy 93.1000
```

Train dataset with Wide Residual Networks (WRNs)

```
wrn = models.wide_resnet101_2(pretrained=True)
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:223: UserWarning
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/wide_resnet101_2-32ee1156.pth" to
100%                                                243M/243M [00:01<00:00, 276MB/s]
```

```
for param in wrn.parameters():
    param.requires_grad = False
```

```
wrn.fc = nn.Sequential(nn.Linear(2048, 1024),
                          nn.ReLU(),
                          nn.Dropout(0.5),
                          nn.Linear(1024, 6),
                          nn.LogSoftmax(dim=1))

wrn = wrn.to(device)
```

```
data_transforms = {
    train: transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(0.5),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    test: transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

```
image_datasets = {x: datasets.ImageFolder(os.path.join(x), data_transforms[x]) for x in [t

dataloader = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=32, shuffle=Tru

dataset_sizes = {x: len(image_datasets[x]) for  x in [train,test]}

class_names = image_datasets[train].classes
```

```
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarni
  warnings.warn(_create_warning_msg(
```

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(wrn.parameters(), lr=1e-3, betas=(0.9, 0.9))
```

```
import torchsummary as summary
```

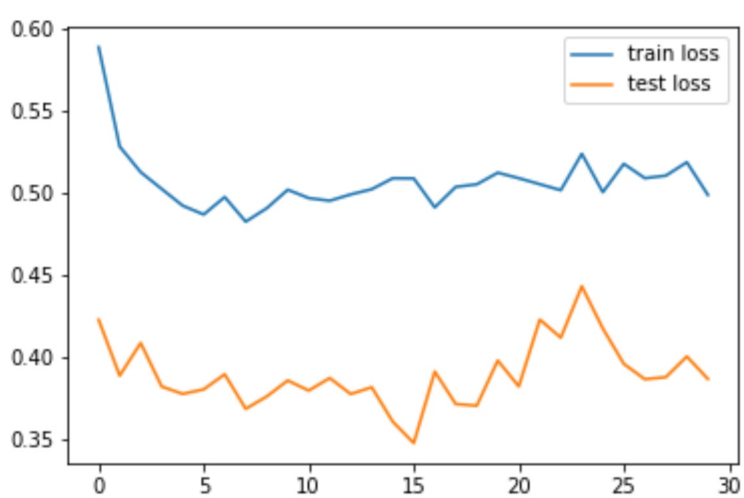
```
def train_model(model, criterion, optimizer, train_loader, test_loader, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)

    for it in range(epochs):
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in dataloader[train]:
            inputs, targets = inputs.to(device), targets.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            loss.backward()
            optimizer.step()
            train_loss.append(loss.item())

        train_loss = np.mean(train_loss)

        test_loss = []
        for inputs, targets in dataloader[test]:
            inputs, targets = inputs.to(device), targets.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            test_loss.append(loss.item())

        test_loss = np.mean(test_loss)

        train_losses[it] = train_loss
        test_losses[it] = test_loss

        dt = datetime.now() - t0

        print(f'Epoch {it+1}/{epochs}, Train_Loss: {train_loss:.4f}, \Test_Loss: {test_los

    return train_losses, test_losses
```

```
train_losses, test_losses = train_model(
                            wrn,
                            criterion,
                            optimizer,
```

```
                       dataloader[train],
                       dataloader[test],
                       epochs=30)
```

```
Epoch 1/30, Train_Loss: 0.5889, \Test_Loss: 0.4229, Duration: 0:02:47.483557
Epoch 2/30, Train_Loss: 0.5284, \Test_Loss: 0.3889, Duration: 0:02:53.000433
Epoch 3/30, Train_Loss: 0.5128, \Test_Loss: 0.4088, Duration: 0:02:55.257686
Epoch 4/30, Train_Loss: 0.5026, \Test_Loss: 0.3822, Duration: 0:02:56.183388
Epoch 5/30, Train_Loss: 0.4924, \Test_Loss: 0.3778, Duration: 0:02:55.113532
Epoch 6/30, Train_Loss: 0.4870, \Test_Loss: 0.3805, Duration: 0:02:54.972858
Epoch 7/30, Train_Loss: 0.4976, \Test_Loss: 0.3897, Duration: 0:02:55.538814
Epoch 8/30, Train_Loss: 0.4827, \Test_Loss: 0.3687, Duration: 0:02:55.844843
Epoch 9/30, Train_Loss: 0.4908, \Test_Loss: 0.3762, Duration: 0:02:56.095436
Epoch 10/30, Train_Loss: 0.5021, \Test_Loss: 0.3859, Duration: 0:02:55.548505
Epoch 11/30, Train_Loss: 0.4970, \Test_Loss: 0.3798, Duration: 0:02:55.285072
Epoch 12/30, Train_Loss: 0.4954, \Test_Loss: 0.3874, Duration: 0:02:55.819190
Epoch 13/30, Train_Loss: 0.4992, \Test_Loss: 0.3777, Duration: 0:02:55.143227
Epoch 14/30, Train_Loss: 0.5024, \Test_Loss: 0.3818, Duration: 0:02:55.304871
Epoch 15/30, Train_Loss: 0.5091, \Test_Loss: 0.3609, Duration: 0:02:55.338294
Epoch 16/30, Train_Loss: 0.5090, \Test_Loss: 0.3478, Duration: 0:02:56.312939
Epoch 17/30, Train_Loss: 0.4913, \Test_Loss: 0.3913, Duration: 0:02:55.157252
Epoch 18/30, Train_Loss: 0.5038, \Test_Loss: 0.3716, Duration: 0:02:55.323865
Epoch 19/30, Train_Loss: 0.5054, \Test_Loss: 0.3706, Duration: 0:02:55.515696
Epoch 20/30, Train_Loss: 0.5124, \Test_Loss: 0.3981, Duration: 0:02:55.673086
Epoch 21/30, Train_Loss: 0.5091, \Test_Loss: 0.3823, Duration: 0:02:55.518970
Epoch 22/30, Train_Loss: 0.5055, \Test_Loss: 0.4230, Duration: 0:02:55.496015
Epoch 23/30, Train_Loss: 0.5018, \Test_Loss: 0.4120, Duration: 0:02:55.743689
Epoch 24/30, Train_Loss: 0.5240, \Test_Loss: 0.4433, Duration: 0:02:55.354273
Epoch 25/30, Train_Loss: 0.5006, \Test_Loss: 0.4176, Duration: 0:02:55.864043
Epoch 26/30, Train_Loss: 0.5179, \Test_Loss: 0.3961, Duration: 0:02:56.052837
Epoch 27/30, Train_Loss: 0.5091, \Test_Loss: 0.3867, Duration: 0:02:55.890951
Epoch 28/30, Train_Loss: 0.5107, \Test_Loss: 0.3880, Duration: 0:02:55.322546
Epoch 29/30, Train_Loss: 0.5188, \Test_Loss: 0.4006, Duration: 0:02:55.645223
Epoch 30/30, Train_Loss: 0.4990, \Test_Loss: 0.3869, Duration: 0:02:55.779137
```

```
plt.plot(train_losses, label='train loss')
plt.plot(test_losses, label='test loss')
plt.legend()
plt.show()
```



```
n_correct = 0
n_total = 0
for inputs, targets in dataloader[train]:
  inputs, targets = inputs.to(device), targets.to(device)
  outputs = wrn(inputs)
  _, predictions = torch.max(outputs, -1)
  n_correct += (predictions == targets).sum().item()
  n_total += targets.shape[0]
train_acc = n_correct/n_total

n_correct = 0
n_total = 0
for inputs, targets in dataloader[test]:
  inputs, targets = inputs.to(device), targets.to(device)
  outputs = wrn(inputs)
  _, predictions = torch.max(outputs, -1)
  n_correct += (predictions == targets).sum().item()
  n_total += targets.shape[0]
test_acc = n_correct/n_total
print(f'Train acc: {train_acc:.4f}, Test acc: {test_acc:.4f}')
```

```
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarni
  warnings.warn(_create_warning_msg(
Train acc: 0.8312, Test acc: 0.8653
```

```
train_losses=[]
test_losses=[]
def accuracy(loader, model):
    num_corrects = 0
    num_samples = 0
    model.eval()
    loop = tqdm(loader)
    with torch.no_grad():
        for x, y in loop:
            x = x.to(device)
            y = y.to(device)
            scores = model(x)
            test_losses.append(scores.data)
            _, prediction = scores.max(1)
            num_corrects += (prediction == y).sum()
            num_samples += prediction.size(0)
            acc = (num_corrects/num_samples) * 100
            loop.set_postfix(acc=acc.item())
        print(f'Got {num_corrects}/{num_samples} with accuracy {acc:.4f}')
```

```
accuracy(dataloader[test], wrn)
```

```
100%                                        94/94 [00:29<00:00, 3.54it/s, acc=89.5]
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarni
  warnings.warn(_create_warning_msg(
Got 2685/3000 with accuracy 89.5000
```

Train intel-image using Inception-ResNet

```
model = models.inception_v3(pretrained=True)
```

```
def get_model():
    model = models.inception_v3(pretrained=True)
    for param in model.parameters():
        param.requires_grad = False #Freezing all the layers and changing only the below l
    model.avgpool = nn.AdaptiveAvgPool2d(output_size=(1,1))
    model.fc = nn.Sequential(nn.Flatten(),
                        nn.Linear(2048,1024),
                        nn.ReLU(),
                        nn.Dropout(0.3),
                        nn.Linear(1024,256),
                        nn.ReLU(),
                        nn.Dropout(0.3),
                        nn.Linear(256,6),
                        nn.LogSoftmax(dim=1))
    model.aux_logits = False
    loss_fn = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=1e-4, betas=(0.9, 0.9))
    return model.to(device), loss_fn, optimizer
```

```
!pip install torchsummary
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: torchsummary in /usr/local/lib/python3.8/dist-package
```

```
from torchsummary import summary

input_shape = (3,300,300)
summary(model.to(device), input_shape)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 149, 149]             864
       BatchNorm2d-2         [-1, 32, 149, 149]              64
       BasicConv2d-3         [-1, 32, 149, 149]               0
            Conv2d-4         [-1, 32, 147, 147]           9,216
       BatchNorm2d-5         [-1, 32, 147, 147]              64
       BasicConv2d-6         [-1, 32, 147, 147]               0
            Conv2d-7         [-1, 64, 147, 147]          18,432
       BatchNorm2d-8         [-1, 64, 147, 147]             128
       BasicConv2d-9         [-1, 64, 147, 147]               0
        MaxPool2d-10           [-1, 64, 73, 73]               0
           Conv2d-11           [-1, 80, 73, 73]           5,120
      BatchNorm2d-12           [-1, 80, 73, 73]             160
      BasicConv2d-13           [-1, 80, 73, 73]               0
           Conv2d-14          [-1, 192, 71, 71]         138,240
      BatchNorm2d-15          [-1, 192, 71, 71]             384
      BasicConv2d-16          [-1, 192, 71, 71]               0
        MaxPool2d-17          [-1, 192, 35, 35]               0
           Conv2d-18           [-1, 64, 35, 35]          12,288
      BatchNorm2d-19           [-1, 64, 35, 35]             128
      BasicConv2d-20           [-1, 64, 35, 35]               0
           Conv2d-21           [-1, 48, 35, 35]           9,216
      BatchNorm2d-22           [-1, 48, 35, 35]              96
      BasicConv2d-23           [-1, 48, 35, 35]               0
           Conv2d-24           [-1, 64, 35, 35]          76,800
      BatchNorm2d-25           [-1, 64, 35, 35]             128
      BasicConv2d-26           [-1, 64, 35, 35]               0
           Conv2d-27           [-1, 64, 35, 35]          12,288
      BatchNorm2d-28           [-1, 64, 35, 35]             128
      BasicConv2d-29           [-1, 64, 35, 35]               0
           Conv2d-30           [-1, 96, 35, 35]          55,296
      BatchNorm2d-31           [-1, 96, 35, 35]             192
      BasicConv2d-32           [-1, 96, 35, 35]               0
           Conv2d-33           [-1, 96, 35, 35]          82,944
      BatchNorm2d-34           [-1, 96, 35, 35]             192
      BasicConv2d-35           [-1, 96, 35, 35]               0
           Conv2d-36           [-1, 32, 35, 35]           6,144
      BatchNorm2d-37           [-1, 32, 35, 35]              64
      BasicConv2d-38           [-1, 32, 35, 35]               0
       InceptionA-39          [-1, 256, 35, 35]               0
           Conv2d-40           [-1, 64, 35, 35]          16,384
      BatchNorm2d-41           [-1, 64, 35, 35]             128
      BasicConv2d-42           [-1, 64, 35, 35]               0
           Conv2d-43           [-1, 48, 35, 35]          12,288
      BatchNorm2d-44           [-1, 48, 35, 35]              96
      BasicConv2d-45           [-1, 48, 35, 35]               0
           Conv2d-46           [-1, 64, 35, 35]          76,800
      BatchNorm2d-47           [-1, 64, 35, 35]             128
      BasicConv2d-48           [-1, 64, 35, 35]               0
           Conv2d-49           [-1, 64, 35, 35]          16,384
      BatchNorm2d-50           [-1, 64, 35, 35]             128
      BasicConv2d-51           [-1, 64, 35, 35]               0
           Conv2d-52           [-1, 96, 35, 35]          55,296
```

```
      BatchNorm2d-53          [-1, 96, 35, 35]             192
      BasicConv2d-54          [-1, 96, 35, 35]               0
           Conv2d-55          [-1, 96, 35, 35]          82,944
```

```
data_transforms = {
    train: transforms.Compose([
        transforms.Resize((300,300)),
        transforms.RandomRotation(degrees=(0, 180)),
        transforms.RandomHorizontalFlip(0.5),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    test: transforms.Compose([
        transforms.Resize((300,300)),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}


image_datasets = {x: datasets.ImageFolder(os.path.join(x), data_transforms[x]) for x in [t

dataloader = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=32, shuffle=Tru

dataset_sizes = {x: len(image_datasets[x]) for  x in [train,test]}

class_names = image_datasets[train].classes

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

```
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarni
  warnings.warn(_create_warning_msg(
```

```
def train_batch(x, y, model, opt, loss_fn):
    output = model(x)
#     print(f"type of output - {type(output)}")
    batch_loss = loss_fn(output, y)
    batch_loss.backward()
    optimizer.step()
    optimizer.zero_grad()
    return batch_loss.item()

@torch.no_grad()
def accuracy(x, y, model):
    model.eval()
    prediction = model(x)
    max_values, argmaxes = prediction.max(-1)
    is_correct = argmaxes == y
    return is_correct.cpu().numpy().tolist()
```

```
@torch.no_grad()
def val_loss(x, y, model):
    model.eval()
    prediction = model(x)
    val_loss = loss_fn(prediction, y)
    return val_loss.item()
```

```
model, loss_fn, optimizer = get_model()
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:208: UserWarning
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:223: UserWarning
  warnings.warn(msg)
```

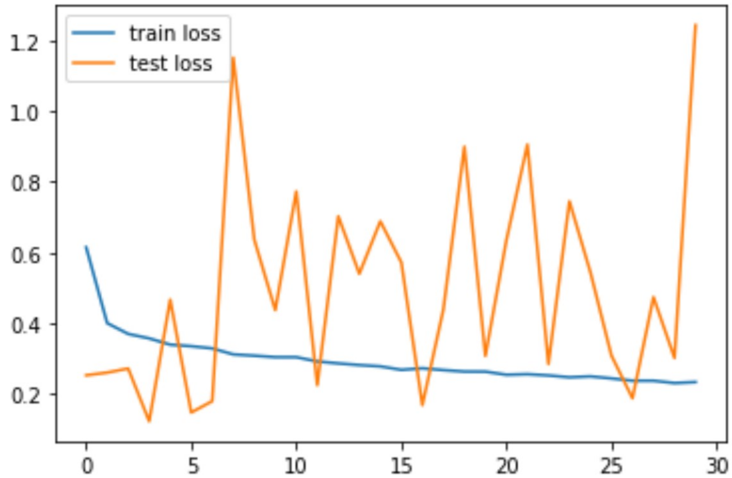
```
train_losses, train_accuracies = [], []
val_losses, val_accuracies = [], []
for epoch in range(1,31):
    d0 = datetime.now()
    print(epoch)
    train_epoch_losses, train_epoch_accuracies = [], []
    for ix, batch in enumerate(iter(dataloader[train])):
#         print(f"ix - {ix}, {batch}")
        x, y = batch
#         print(f"type of x - {type(x)}, type of y - {type(y)}")
        x, y= x.to(device), y.to(device)
        batch_loss = train_batch(x, y, model, optimizer, loss_fn)
        is_correct = accuracy(x, y, model)
        train_epoch_accuracies.extend(is_correct)
        train_epoch_losses.append(batch_loss)
    train_epoch_loss = np.array(train_epoch_losses).mean()
    train_epoch_accuracy = np.mean(train_epoch_accuracies)
    print('Epoch:',epoch,'Train Loss:',train_epoch_loss,'Train Accuracy:',train_epoch_accu

    for ix, batch in enumerate(iter(dataloader[test])):
        x, y = batch
        x, y= x.to(device), y.to(device)
        val_is_correct = accuracy(x, y, model)
        validation_loss = val_loss(x, y, model)
        val_epoch_accuracy = np.mean(val_is_correct)
    dt = datetime.now() - d0
    print('Epoch:',epoch,'Validation Loss:',validation_loss,'Validation Accuracy:',val_epo

    train_losses.append(train_epoch_loss)
    train_accuracies.append(train_epoch_accuracy)
    val_losses.append(validation_loss)
    val_accuracies.append(val_epoch_accuracy)
```

```
1
Epoch: 1 Train Loss: 0.6147843960816877 Train Accuracy: 0.7997719823286304
```

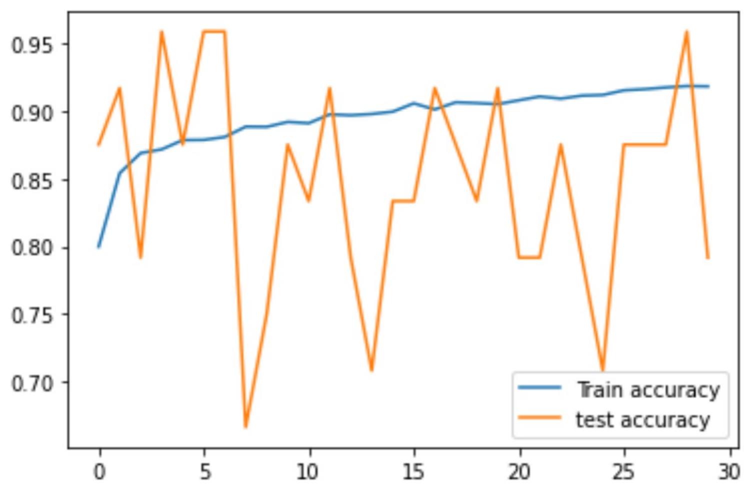
```
Epoch: 1 Validation Loss: 0.2536163628101349 Validation Accuracy: 0.875 Duration: 0:
2
Epoch: 2 Train Loss: 0.4008501869263573 Train Accuracy: 0.8539261792788941
Epoch: 2 Validation Loss: 0.26119133830070496 Validation Accuracy: 0.916666666666666
3
Epoch: 3 Train Loss: 0.3703767183260114 Train Accuracy: 0.8687473279179136
Epoch: 3 Validation Loss: 0.27270570397377014 Validation Accuracy: 0.791666666666666
4
Epoch: 4 Train Loss: 0.35774958668306367 Train Accuracy: 0.8714550377654268
Epoch: 4 Validation Loss: 0.1237589493393898 Validation Accuracy: 0.9583333333333334
5
Epoch: 5 Train Loss: 0.3400826733856375 Train Accuracy: 0.8782955679065128
Epoch: 5 Validation Loss: 0.4670526683330536 Validation Accuracy: 0.875 Duration: 0:
6
Epoch: 6 Train Loss: 0.33567620564362455 Train Accuracy: 0.8784380789511187
Epoch: 6 Validation Loss: 0.14849501848220825 Validation Accuracy: 0.958333333333333
7
Epoch: 7 Train Loss: 0.3294972089672958 Train Accuracy: 0.880575744620208
Epoch: 7 Validation Loss: 0.18056625127792358 Validation Accuracy: 0.958333333333333
8
Epoch: 8 Train Loss: 0.31252267434005043 Train Accuracy: 0.8882000855066268
Epoch: 8 Validation Loss: 1.1506531238555908 Validation Accuracy: 0.6666666666666666
9
Epoch: 9 Train Loss: 0.30893714871927924 Train Accuracy: 0.8880575744620208
Epoch: 9 Validation Loss: 0.6368729472160339 Validation Accuracy: 0.75 Duration: 0:0
10
Epoch: 10 Train Loss: 0.304772251932659 Train Accuracy: 0.8916916060994727
Epoch: 10 Validation Loss: 0.4377513825893402 Validation Accuracy: 0.875 Duration: 0
11
Epoch: 11 Train Loss: 0.3046975946966092 Train Accuracy: 0.890765284309534
Epoch: 11 Validation Loss: 0.772612988948822 Validation Accuracy: 0.8333333333333334
12
Epoch: 12 Train Loss: 0.2926434572523031 Train Accuracy: 0.897249536839105
Epoch: 12 Validation Loss: 0.22557474672794342 Validation Accuracy: 0.91666666666666
13
Epoch: 13 Train Loss: 0.2871898828596216 Train Accuracy: 0.8966082371383782
Epoch: 13 Validation Loss: 0.7025075554847717 Validation Accuracy: 0.791666666666666
14
Epoch: 14 Train Loss: 0.28246640831503617 Train Accuracy: 0.8975345589283169
Epoch: 14 Validation Loss: 0.5403130650520325 Validation Accuracy: 0.708333333333333
15
Epoch: 15 Train Loss: 0.2787607992089148 Train Accuracy: 0.8992446914635884
Epoch: 15 Validation Loss: 0.6883854269981384 Validation Accuracy: 0.833333333333333
16
Epoch: 16 Train Loss: 0.26896397121131826 Train Accuracy: 0.9053726663816446
Epoch: 16 Validation Loss: 0.5723026394844055 Validation Accuracy: 0.833333333333333
17
Epoch: 17 Train Loss: 0.2733323503463849 Train Accuracy: 0.9008835684765569
Epoch: 17 Validation Loss: 0.1693018078804016 Validation Accuracy: 0.916666666666666
18
Epoch: 18 Train Loss: 0.2682374722678596 Train Accuracy: 0.9061564771269773
Epoch: 18 Validation Loss: 0.4393166016578674 Validation Accuracy: 0.875 Duration: 
19
Epoch: 19 Train Loss: 0.2641515208913145 Train Accuracy: 0.9055864329485536
Epoch: 19 Validation Loss: 0.8991274833679199 Validation Accuracy: 0.833333333333333
```

20

```
plt.plot(train_losses, label='train loss')
plt.plot(val_losses, label='test loss')
plt.legend()
plt.show()
```




```
plt.plot(train_accuracies, label='Train accuracy')
plt.plot(val_accuracies, label='test accuracy')
plt.legend()
plt.show()
```




```
n_correct = 0
n_total = 0
for inputs, targets in dataloader[train]:
  inputs, targets = inputs.to(device), targets.to(device)
  outputs = model(inputs)
  _, predictions = torch.max(outputs, -1)
  n_correct += (predictions == targets).sum().item()
  n_total += targets.shape[0]
train_acc = n_correct/n_total
```

```
n_correct = 0
n_total = 0
for inputs, targets in dataloader[test]:
  inputs, targets = inputs.to(device), targets.to(device)
  outputs = model(inputs)
  _, predictions = torch.max(outputs, -1)
  n_correct += (predictions == targets).sum().item()
  n_total += targets.shape[0]
test_acc = n_correct/n_total
print(f'Train acc: {train_acc:.4f}, Test acc: {test_acc:.4f}')
```

```
Train acc: 0.9111, Test acc: 0.8193
```

```
train_losses=[]
test_losses=[]
def accuracy(loader, model):
    num_corrects = 0
    num_samples = 0
    model.eval()
    loop = tqdm(loader)
    with torch.no_grad():
        for x, y in loop:
            x = x.to(device)
            y = y.to(device)
            scores = model(x)
            test_losses.append(scores.data)
            _, prediction = scores.max(1)
            num_corrects += (prediction == y).sum()
            num_samples += prediction.size(0)
            acc = (num_corrects/num_samples) * 100
            loop.set_postfix(acc=acc.item())
        print(f'Got {num_corrects}/{num_samples} with accuracy {acc:.4f}')
```

```
accuracy(dataloader[test], model)
```

100% 94/94 [00:10<00:00, 12.14it/s, acc=87]

```
Got 2609/3000 with accuracy 86.9667
```

Models summary

```
from torchsummary import summary

input_shape = (3,300,300)
summary(resnet.to(device), input_shape)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 150, 150]           9,408
       BatchNorm2d-2         [-1, 64, 150, 150]             128
              ReLU-3         [-1, 64, 150, 150]               0
         MaxPool2d-4           [-1, 64, 75, 75]               0
            Conv2d-5           [-1, 64, 75, 75]           4,096
       BatchNorm2d-6           [-1, 64, 75, 75]             128
              ReLU-7           [-1, 64, 75, 75]               0
            Conv2d-8           [-1, 64, 75, 75]          36,864
       BatchNorm2d-9           [-1, 64, 75, 75]             128
             ReLU-10           [-1, 64, 75, 75]               0
           Conv2d-11          [-1, 256, 75, 75]          16,384
      BatchNorm2d-12          [-1, 256, 75, 75]             512
           Conv2d-13          [-1, 256, 75, 75]          16,384
      BatchNorm2d-14          [-1, 256, 75, 75]             512
             ReLU-15          [-1, 256, 75, 75]               0
       Bottleneck-16          [-1, 256, 75, 75]               0
           Conv2d-17           [-1, 64, 75, 75]          16,384
      BatchNorm2d-18           [-1, 64, 75, 75]             128
             ReLU-19           [-1, 64, 75, 75]               0
           Conv2d-20           [-1, 64, 75, 75]          36,864
      BatchNorm2d-21           [-1, 64, 75, 75]             128
             ReLU-22           [-1, 64, 75, 75]               0
           Conv2d-23          [-1, 256, 75, 75]          16,384
      BatchNorm2d-24          [-1, 256, 75, 75]             512
             ReLU-25          [-1, 256, 75, 75]               0
       Bottleneck-26          [-1, 256, 75, 75]               0
           Conv2d-27           [-1, 64, 75, 75]          16,384
      BatchNorm2d-28           [-1, 64, 75, 75]             128
             ReLU-29           [-1, 64, 75, 75]               0
           Conv2d-30           [-1, 64, 75, 75]          36,864
      BatchNorm2d-31           [-1, 64, 75, 75]             128
             ReLU-32           [-1, 64, 75, 75]               0
           Conv2d-33          [-1, 256, 75, 75]          16,384
      BatchNorm2d-34          [-1, 256, 75, 75]             512
             ReLU-35          [-1, 256, 75, 75]               0
       Bottleneck-36          [-1, 256, 75, 75]               0
           Conv2d-37          [-1, 128, 75, 75]          32,768
      BatchNorm2d-38          [-1, 128, 75, 75]             256
             ReLU-39          [-1, 128, 75, 75]               0
           Conv2d-40          [-1, 128, 38, 38]         147,456
      BatchNorm2d-41          [-1, 128, 38, 38]             256
             ReLU-42          [-1, 128, 38, 38]               0
           Conv2d-43          [-1, 512, 38, 38]          65,536
      BatchNorm2d-44          [-1, 512, 38, 38]           1,024
           Conv2d-45          [-1, 512, 38, 38]         131,072
      BatchNorm2d-46          [-1, 512, 38, 38]           1,024
             ReLU-47          [-1, 512, 38, 38]               0
       Bottleneck-48          [-1, 512, 38, 38]               0
           Conv2d-49          [-1, 128, 38, 38]          65,536
      BatchNorm2d-50          [-1, 128, 38, 38]             256
             ReLU-51          [-1, 128, 38, 38]               0
           Conv2d-52          [-1, 128, 38, 38]         147,456
```

```
    BatchNorm2d-53          [-1, 128, 38, 38]             256
           ReLU-54          [-1, 128, 38, 38]               0
         Conv2d-55          [-1, 512, 38, 38]          65,536
```

```
from torchsummary import summary

input_shape = (3,300,300)
summary(wrn.to(device), input_shape)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 64, 150, 150]           9,408
       BatchNorm2d-2        [-1, 64, 150, 150]             128
              ReLU-3        [-1, 64, 150, 150]               0
         MaxPool2d-4          [-1, 64, 75, 75]               0
            Conv2d-5         [-1, 128, 75, 75]           8,192
       BatchNorm2d-6         [-1, 128, 75, 75]             256
              ReLU-7         [-1, 128, 75, 75]               0
            Conv2d-8         [-1, 128, 75, 75]         147,456
       BatchNorm2d-9         [-1, 128, 75, 75]             256
             ReLU-10         [-1, 128, 75, 75]               0
           Conv2d-11         [-1, 256, 75, 75]          32,768
      BatchNorm2d-12         [-1, 256, 75, 75]             512
           Conv2d-13         [-1, 256, 75, 75]          16,384
      BatchNorm2d-14         [-1, 256, 75, 75]             512
             ReLU-15         [-1, 256, 75, 75]               0
       Bottleneck-16         [-1, 256, 75, 75]               0
           Conv2d-17         [-1, 128, 75, 75]          32,768
      BatchNorm2d-18         [-1, 128, 75, 75]             256
             ReLU-19         [-1, 128, 75, 75]               0
           Conv2d-20         [-1, 128, 75, 75]         147,456
      BatchNorm2d-21         [-1, 128, 75, 75]             256
             ReLU-22         [-1, 128, 75, 75]               0
           Conv2d-23         [-1, 256, 75, 75]          32,768
      BatchNorm2d-24         [-1, 256, 75, 75]             512
             ReLU-25         [-1, 256, 75, 75]               0
       Bottleneck-26         [-1, 256, 75, 75]               0
           Conv2d-27         [-1, 128, 75, 75]          32,768
      BatchNorm2d-28         [-1, 128, 75, 75]             256
             ReLU-29         [-1, 128, 75, 75]               0
           Conv2d-30         [-1, 128, 75, 75]         147,456
      BatchNorm2d-31         [-1, 128, 75, 75]             256
             ReLU-32         [-1, 128, 75, 75]               0
           Conv2d-33         [-1, 256, 75, 75]          32,768
      BatchNorm2d-34         [-1, 256, 75, 75]             512
             ReLU-35         [-1, 256, 75, 75]               0
       Bottleneck-36         [-1, 256, 75, 75]               0
           Conv2d-37         [-1, 256, 75, 75]          65,536
      BatchNorm2d-38         [-1, 256, 75, 75]             512
             ReLU-39         [-1, 256, 75, 75]               0
           Conv2d-40         [-1, 256, 38, 38]         589,824
      BatchNorm2d-41         [-1, 256, 38, 38]             512
             ReLU-42         [-1, 256, 38, 38]               0
           Conv2d-43         [-1, 512, 38, 38]         131,072
```

```
         Conv2d-43          [-1, 512, 38, 38]         131,072
    BatchNorm2d-44          [-1, 512, 38, 38]           1,024
         Conv2d-45          [-1, 512, 38, 38]         131,072
    BatchNorm2d-46          [-1, 512, 38, 38]           1,024
           ReLU-47          [-1, 512, 38, 38]               0
     Bottleneck-48          [-1, 512, 38, 38]               0
         Conv2d-49          [-1, 256, 38, 38]         131,072
    BatchNorm2d-50          [-1, 256, 38, 38]             512
           ReLU-51          [-1, 256, 38, 38]               0
         Conv2d-52          [-1, 256, 38, 38]         589,824
    BatchNorm2d-53          [-1, 256, 38, 38]             512
           ReLU-54          [-1, 256, 38, 38]               0
         Conv2d-55          [-1, 512, 38, 38]         131,072
```

```
from torchsummary import summary

input_shape = (3,300,300)
summary(model.to(device), input_shape)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 149, 149]             864
       BatchNorm2d-2         [-1, 32, 149, 149]              64
      BasicConv2d-3         [-1, 32, 149, 149]               0
            Conv2d-4         [-1, 32, 147, 147]           9,216
       BatchNorm2d-5         [-1, 32, 147, 147]              64
      BasicConv2d-6         [-1, 32, 147, 147]               0
            Conv2d-7         [-1, 64, 147, 147]          18,432
       BatchNorm2d-8         [-1, 64, 147, 147]             128
      BasicConv2d-9         [-1, 64, 147, 147]               0
       MaxPool2d-10           [-1, 64, 73, 73]               0
          Conv2d-11           [-1, 80, 73, 73]           5,120
     BatchNorm2d-12           [-1, 80, 73, 73]             160
     BasicConv2d-13           [-1, 80, 73, 73]               0
          Conv2d-14          [-1, 192, 71, 71]         138,240
     BatchNorm2d-15          [-1, 192, 71, 71]             384
     BasicConv2d-16          [-1, 192, 71, 71]               0
       MaxPool2d-17          [-1, 192, 35, 35]               0
          Conv2d-18           [-1, 64, 35, 35]          12,288
     BatchNorm2d-19           [-1, 64, 35, 35]             128
     BasicConv2d-20           [-1, 64, 35, 35]               0
          Conv2d-21           [-1, 48, 35, 35]           9,216
     BatchNorm2d-22           [-1, 48, 35, 35]              96
     BasicConv2d-23           [-1, 48, 35, 35]               0
          Conv2d-24           [-1, 64, 35, 35]          76,800
     BatchNorm2d-25           [-1, 64, 35, 35]             128
     BasicConv2d-26           [-1, 64, 35, 35]               0
          Conv2d-27           [-1, 64, 35, 35]          12,288
     BatchNorm2d-28           [-1, 64, 35, 35]             128
     BasicConv2d-29           [-1, 64, 35, 35]               0
          Conv2d-30           [-1, 96, 35, 35]          55,296
     BatchNorm2d-31           [-1, 96, 35, 35]             192
     BasicConv2d-32           [-1, 96, 35, 35]               0
          Conv2d-33           [-1, 96, 35, 35]          82,944
```

```
    BatchNorm2d-34          [-1, 96, 35, 35]             192
    BasicConv2d-35          [-1, 96, 35, 35]               0
         Conv2d-36          [-1, 32, 35, 35]           6,144
    BatchNorm2d-37          [-1, 32, 35, 35]              64
    BasicConv2d-38          [-1, 32, 35, 35]               0
     InceptionA-39         [-1, 256, 35, 35]               0
         Conv2d-40          [-1, 64, 35, 35]          16,384
    BatchNorm2d-41          [-1, 64, 35, 35]             128
    BasicConv2d-42          [-1, 64, 35, 35]               0
         Conv2d-43          [-1, 48, 35, 35]          12,288
    BatchNorm2d-44          [-1, 48, 35, 35]              96
    BasicConv2d-45          [-1, 48, 35, 35]               0
         Conv2d-46          [-1, 64, 35, 35]          76,800
    BatchNorm2d-47          [-1, 64, 35, 35]             128
    BasicConv2d-48          [-1, 64, 35, 35]               0
         Conv2d-49          [-1, 64, 35, 35]          16,384
    BatchNorm2d-50          [-1, 64, 35, 35]             128
    BasicConv2d-51          [-1, 64, 35, 35]               0
         Conv2d-52          [-1, 96, 35, 35]          55,296
    BatchNorm2d-53          [-1, 96, 35, 35]             192
    BasicConv2d-54          [-1, 96, 35, 35]               0
         Conv2d-55          [-1, 96, 35, 35]          82,944
```