Lab11 Report

st122314

In this lab, there are three parts to report which are the following:

- Visual Transformer(ViT) from scratch
- Pretrained Pytorch ViT and finetuning

1.Visual Transformer(ViT) from scratch

1.1 Load data

In this part, MNIST dataset from the root directory were downloaded and put into dataloader for both train and test sets.

In [7]:

```
import numpy as np

import torch
import torch.nn as nn
from torch.nn import CrossEntropyLoss
from torch.optim import Adam
from torch.utils.data import DataLoader

from torchvision.datasets.mnist import MNIST
from torchvision.transforms import ToTensor

# Loading data
transform = ToTensor()

train_set = MNIST(root='./../datasets', train=True, download=True, transform=transfo
test_set = MNIST(root='./../datasets', train=False, download=True, transform=transfo

train_loader = DataLoader(train_set, shuffle=True, batch_size=16)
test_loader = DataLoader(test_set, shuffle=False, batch_size=16)
```

1.2 Train and Test functions

Train and test functions were employed for Visual Transformers according to the following cell.

In [8]:

```
def train_ViT_classify(model, optimizer, N_EPOCHS, train_loader, device="cpu"):
    criterion = CrossEntropyLoss()
    for epoch in range(N_EPOCHS):
        train_loss = 0.0
        for batch in train_loader:
            x, y = batch
            x = x.to(device)
            y = y.to(device)
            y_hat = model(x)
            loss = criterion(y_hat, y) / len(x)

            train_loss += loss.item()

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        print(f"Epoch {epoch + 1}/{N_EPOCHS} loss: {train_loss:.2f}")

def test_ViT_classify(model, optimizer, test_loader):
    criterion = CrossEntropyLoss()
    correct, total = 0, 0
    test_loss = 0.0
    for batch in test_loader:
        x, y = batch
        x = x.to(device)
        y = y.to(device)

        y_hat = model(x)
        loss = criterion(y_hat, y) / len(x)
        test_loss += loss

        correct += torch.sum(torch.argmax(y_hat, dim=1) == y).item()
        total += len(x)
    print(f"Test loss: {test_loss:.2f}")
    print(f"Test accuracy: {correct / total * 100:.2f}%")
```

1.3 Multi-head Self Attention (MSA) Model

In [9]:

```
class MSA(nn.Module):
    def __init__(self, d, n_heads=2):
        super(MSA, self).__init__()
        self.d = d
        self.n_heads = n_heads

        assert d % n_heads == 0, f"Can't divide dimension {d} into {n_heads} heads"

        d_head = int(d / n_heads)
        self.q_mappings = [nn.Linear(d_head, d_head) for _ in range(self.n_heads)]
        self.k_mappings = [nn.Linear(d_head, d_head) for _ in range(self.n_heads)]
        self.v_mappings = [nn.Linear(d_head, d_head) for _ in range(self.n_heads)]
        self.d_head = d_head
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, sequences):
        # Sequences has shape (N, seq_length, token_dim)
        # We go into shape    (N, seq_length, n_heads, token_dim / n_heads)
        # And come back to    (N, seq_length, item_dim)  (through concatenation)
        result = []
        for sequence in sequences:
            seq_result = []
            for head in range(self.n_heads):
                q_mapping = self.q_mappings[head]
                k_mapping = self.k_mappings[head]
                v_mapping = self.v_mappings[head]

                seq = sequence[:, head * self.d_head: (head + 1) * self.d_head]
                q, k, v = q_mapping(seq), k_mapping(seq), v_mapping(seq)

                attention = self.softmax(q @ k.T / (self.d_head ** 0.5))
                seq_result.append(attention @ v)
            result.append(torch.hstack(seq_result))
        return torch.cat([torch.unsqueeze(r, dim=0) for r in result])
```

1.4 Position encoding

In [10]:

```
def get_positional_embeddings(sequence_length, d, device="cpu"):
    result = torch.ones(sequence_length, d)
    for i in range(sequence_length):
        for j in range(d):
            result[i][j] = np.sin(i / (10000 ** (j / d))) if j % 2 == 0 else np.cos(
    return result.to(device)
```

1.5 ViT Model

Step 1: Patchifying and the linear mapping

Step 2: Adding the classification token

Step 3: Positional encoding

Step 4: LN, MSA, and Residual Connection

Step 5: LN, MLP, and Residual Connection

Step 6: Classification MLP

In [11]:

```
model = ViT((1, 28, 28), n_patches=7, hidden_d=20, n_heads=2, out_d=10)
model = model.to(device)

N_EPOCHS = 5
LR = 0.01
optimizer = Adam(model.parameters(), lr=LR)

train_ViT_classify(model, optimizer, N_EPOCHS, train_loader, device)

test_ViT_classify(model, optimizer, test_loader)
```

After running the model, the following figure is the training loss for each epochs. Then, Test loss is 61.41 and test accuracy is 88.88%

```
st122314@9a1d87bf22c3:~/Lab11$ /bin/python3 /home/st122314/Lab11/train_test.py
st122314@9a1d87bf22c3:~/Lab11$ /bin/python3 /home/st122314/Lab11/msa.py
st122314@9a1d87bf22c3:~/Lab11$ /bin/python3 /home/st122314/Lab11/ViT.py
st122314@9a1d87bf22c3:~/Lab11$ /bin/python3 /home/st122314/Lab11/main.py
Epoch 1/5 loss: 406.25
Epoch 2/5 loss: 378.67
Epoch 3/5 loss: 372.71
Epoch 4/5 loss: 370.53
Epoch 5/5 loss: 369.21
Test loss: 61.41
Test accuracy: 88.88%
st122314@9a1d87bf22c3:~/Lab11$
```

2. Pytorch Pre-trained ViT and fine-tuning

2.1 Pre-trained ViT

In [12]:

```
import numpy as np
import time
import torch
import torch.nn as nn
from torch.nn import CrossEntropyLoss
from torch.optim import Adam
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.datasets.mnist import MNIST
from torchvision import transforms
from vit_pytorch import ViT

device = torch.device("cuda:1" if torch.cuda.is_available() else "cpu")
```

2.1.1 Loading Data

ImageNet from the root path is employed to load as train data and validation data after making image tranformation: resize to 256 pixels, randomcrop and randomhorizontalflip. Then, all were put into DataLoader.

In [ ]:

```
# Load data.

print("Loading ImageNet train ...")
train_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))]
    )

train_set = datasets.ImageNet(root='/Dataset/ImageNet',
#train_set = datasets.ImageNet(root='/home/fidji/mdailey/Datasets/Imagenet',
        split = 'train',
        transform=train_transform)

print("Loading ImageNet val ...")
val_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))]
    )

val_set = datasets.ImageNet(root='/Dataset/ImageNet',
#val_set = datasets.ImageNet(root='/home/fidji/mdailey/Datasets/Imagenet',
        split = 'val',
        transform=val_transform)

train_loader = DataLoader(train_set, shuffle=True, batch_size=48, num_workers=4)
val_loader = DataLoader(val_set, shuffle=True, batch_size=48, num_workers=4)
```

2.1.2 Training function

In [ ]:

```
def train(model, optimizer, N_EPOCHS, device, train_loader):
    criterion = CrossEntropyLoss()
    for epoch in range(N_EPOCHS):
        train_loss = 0.0
        epoch_start = time.time()
        for i, (batch) in enumerate(train_loader):
            model = model.to(device)
            x, y = batch
            x = x.to(device)
            y = y.to(device)
            # print('x shape: ', x.shape)
            y_hat = model(x)
            loss = criterion(y_hat, y) / len(x)

            train_loss += loss.item()

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            if (i % 100 == 0):
                print(f'Iteration {i}: train loss {loss.item()}')
                print(f'Time Elapsed: {time.time() - epoch_start} seconds')
                print("----------------------")

            if (i % 20000 == 0):
                torch.save(model.state_dict(), f'model_epoch_{i}.pt')
        print(f"Epoch {epoch + 1}/{N_EPOCHS} loss: {train_loss/ len(train_loader):.2
```

2.1.3 Testing function

In [ ]:

```
def test(model, device, test_loader):
    criterion = nn.CrossEntropyLoss()
    correct, total = 0, 0
    test_loss = 0.0

    model.eval()
    with torch.no_grad():
        for batch in test_loader:
            x, y = batch
            x = x.to(device)
            y = y.to(device)

            y_hat = model(x)
            loss = criterion(y_hat, y) / len(x)
            test_loss += loss

            correct += torch.sum(torch.argmax(y_hat, dim=1) == y).item()
            total += len(x)
        print(f"Test loss: {test_loss:.2f}")
        print(f"Test accuracy: {correct / total * 100:.2f}%")
```

2.1.4 Pre-trained Visual Transformer(ViT) Model

In this ViT model, the image size of 224 was firstly splitted into patch size 16 and used 1024 dimmension with depth 16 and total 6 multi heads. Moreover, 2048 multi layer percetron were employed before 0.1 drop out rate and embedded dropout.

In [ ]:

```
model = ViT(
    image_size = 224,
    patch_size = 16,
    num_classes = 1000,
    dim = 1024,
    depth = 6,
    heads = 16,
    mlp_dim = 2048,
    dropout = 0.1,
    emb_dropout = 0.1
).to(device)
```

2.1.5 Training

In training part, total 5 epochs, 0.001 learning rate and Adam optimizer were used. Then, training model started after counting time and model was saved.

In [ ]:

```
N_EPOCHS = 5
LR = 0.001
optimizer = Adam(model.parameters(), lr=LR)


start_training = time.time()
train(model, optimizer, N_EPOCHS, device, train_loader)

torch.save(model.state_dict(), 'trained-pytorch-vit-imagenet.pt')
print(f"Total Training Time: {time.time() - start_training} seconds")
```

The follwing cell is the reult of training which is total 5 epochs and 26,000 iterations per epoch. Futhermore, it takes around 15 hrs (52038.67340183258 seconds).

```
st122314@a5cc46d8d889:~/Lab11$ python3 pretrained_ViT.py
Loading ImageNet train ...
Loading ImageNet val ...
Iteration 0: train loss 0.14633233845233917
Time Elapsed: 2.359689474105835 seconds
----------------------
Iteration 1000: train loss 0.13888031244277954
Time Elapsed: 402.00834488868713 seconds
----------------------
Iteration 2000: train loss 0.1384449005126953
Time Elapsed: 788.3315346240997 seconds
----------------------
Iteration 3000: train loss 0.1388528048992157
Time Elapsed: 1172.0676605701447 seconds
----------------------
Iteration 4000: train loss 0.13784384727478027
Time Elapsed: 1559.4002575874329 seconds
```

```
----------------------
Iteration 5000: train loss 0.14386095106601715
Time Elapsed: 1997.2671530246735 seconds
----------------------
Iteration 6000: train loss 0.14059361815452576
Time Elapsed: 2375.6388494968414 seconds
----------------------
Iteration 7000: train loss 0.14187641441822052
Time Elapsed: 2757.6220197677612 seconds
----------------------
Iteration 8000: train loss 0.14215943217277527
Time Elapsed: 3138.0855412483215 seconds
----------------------
Iteration 9000: train loss 0.1420522928237915
Time Elapsed: 3519.9479072093964 seconds
----------------------
Iteration 10000: train loss 0.13842365145683289
Time Elapsed: 3895.891693353653 seconds
----------------------
Iteration 11000: train loss 0.14233674108982086
Time Elapsed: 4271.892471551895 seconds
----------------------
Iteration 12000: train loss 0.13715839385986328
Time Elapsed: 4647.771924972534 seconds
----------------------
Iteration 13000: train loss 0.13894450664520264
Time Elapsed: 5023.998697042465 seconds
----------------------
Iteration 14000: train loss 0.14175936579704285
Time Elapsed: 5398.050559997559 seconds
----------------------
Iteration 15000: train loss 0.1397249847650528
Time Elapsed: 5772.897041082382 seconds
----------------------
Iteration 16000: train loss 0.1367110311985016
Time Elapsed: 6147.266947031021 seconds
----------------------
Iteration 17000: train loss 0.13902634382247925
Time Elapsed: 6527.137417078018 seconds
----------------------
Iteration 18000: train loss 0.1418648362159729
Time Elapsed: 6902.311444759369 seconds
----------------------
Iteration 19000: train loss 0.14178717136383057
Time Elapsed: 7277.655646085739 seconds
----------------------
Iteration 20000: train loss 0.1368142068386078
Time Elapsed: 7649.758494138718 seconds
----------------------
Iteration 21000: train loss 0.14399056136608124
Time Elapsed: 8028.297704935074 seconds
----------------------
Iteration 22000: train loss 0.14148715138435364
Time Elapsed: 8401.40327000618 seconds
----------------------
Iteration 23000: train loss 0.13877590000629425
Time Elapsed: 8774.038024663925 seconds
----------------------
Iteration 24000: train loss 0.1418556123971939
Time Elapsed: 9152.340377807617 seconds
----------------------
```

```
Iteration 25000: train loss 0.13843068480491638
Time Elapsed: 9528.682741641998 seconds
----------------------
Iteration 26000: train loss 0.14166519045829773
Time Elapsed: 9902.710470676422 seconds
----------------------

Epoch 1/5 loss: 0.14


Iteration 0: train loss 0.141191303730011
Time Elapsed: 3.058993339538574 seconds
----------------------
Iteration 1000: train loss 0.14147967100143433
Time Elapsed: 404.7766182422638 seconds
----------------------
Iteration 2000: train loss 0.14243097603321075
Time Elapsed: 781.2967851161957 seconds
----------------------
Iteration 3000: train loss 0.13947610557079315
Time Elapsed: 1156.4267480373383 seconds
----------------------
Iteration 4000: train loss 0.13932348787784576
Time Elapsed: 1532.1910614967346 seconds
----------------------
Iteration 5000: train loss 0.14481492340564728
Time Elapsed: 1912.774705171585 seconds
----------------------
Iteration 6000: train loss 0.13876263797283173
Time Elapsed: 2287.7863137722015 seconds
----------------------
Iteration 7000: train loss 0.14442986249923706
Time Elapsed: 2688.7298197746277 seconds
----------------------
Iteration 8000: train loss 0.14299368858337402
Time Elapsed: 3099.3477005958557 seconds
----------------------
Iteration 9000: train loss 0.14187254011631012
Time Elapsed: 3514.6221961975098 seconds
----------------------
Iteration 10000: train loss 0.14141899347305298
Time Elapsed: 4426.271542310715 seconds
----------------------
Iteration 11000: train loss 0.13928182423114777
Time Elapsed: 4932.9295744895935 seconds
----------------------
Iteration 12000: train loss 0.1398921012878418
Time Elapsed: 5317.819102048874 seconds
----------------------
Iteration 13000: train loss 0.1417018175125122
Time Elapsed: 5848.137082576752 seconds
----------------------
Iteration 14000: train loss 0.13926757872104645
Time Elapsed: 6238.944685459137 seconds
----------------------
Iteration 15000: train loss 0.14091581106185913
Time Elapsed: 6617.137130975723 seconds
----------------------
Iteration 15000: train loss 0.1397249847650528
Time Elapsed: 5772.897041082382 seconds
----------------------
```

```
Iteration 16000: train loss 0.1367110311985016
Time Elapsed: 6147.266947031021 seconds
----------------------
Iteration 17000: train loss 0.13902634382247925
Time Elapsed: 6527.137417078018 seconds
----------------------
Iteration 18000: train loss 0.1418648362159729
Time Elapsed: 6902.311444759369 seconds
----------------------
Iteration 19000: train loss 0.14178717136383057
Time Elapsed: 7277.655646085739 seconds
----------------------
Iteration 20000: train loss 0.1368142068386078
Time Elapsed: 7649.758494138718 seconds
----------------------
Iteration 21000: train loss 0.14399056136608124
Time Elapsed: 8028.297704935074 seconds
----------------------
Iteration 22000: train loss 0.14148715138435364
Time Elapsed: 8401.40327000618 seconds
----------------------
Iteration 23000: train loss 0.13877590000629425
Time Elapsed: 8774.038024663925 seconds
----------------------
Iteration 24000: train loss 0.1418556123971939
Time Elapsed: 9152.340377807617 seconds
----------------------
Iteration 25000: train loss 0.13843068480491638
Time Elapsed: 9528.682741641998 seconds
----------------------
Iteration 26000: train loss 0.14166519045829773
Time Elapsed: 9902.710470676422 seconds
----------------------
Epoch 1/5 loss: 0.14
Iteration 0: train loss 0.141191303730011
Time Elapsed: 3.058993339538574 seconds
----------------------
Iteration 1000: train loss 0.14147967100143433
Time Elapsed: 404.7766182422638 seconds
----------------------
Iteration 2000: train loss 0.14243097603321075
Time Elapsed: 781.2967851161957 seconds
----------------------
Iteration 3000: train loss 0.13947610557079315
Time Elapsed: 1156.4267480373383 seconds
----------------------
Iteration 4000: train loss 0.13932348787784576
Time Elapsed: 1532.1910614967346 seconds
----------------------
Iteration 5000: train loss 0.14481492340564728
Time Elapsed: 1912.774705171585 seconds
----------------------
Iteration 6000: train loss 0.13876263797283173
Time Elapsed: 2287.7863137722015 seconds
----------------------
Iteration 7000: train loss 0.14442986249923706
Time Elapsed: 2688.7298197746277 seconds
----------------------
Iteration 8000: train loss 0.14299368858337402
Time Elapsed: 3099.3477005958557 seconds
----------------------
```

```
Iteration 9000: train loss 0.14187254011631012
Time Elapsed: 3514.6221961975098 seconds
----------------------
Iteration 10000: train loss 0.14141899347305298
Time Elapsed: 4426.271542310715 seconds
----------------------
Iteration 11000: train loss 0.13928182423114777
Time Elapsed: 4932.9295744895935 seconds
----------------------
Iteration 12000: train loss 0.1398921012878418
Time Elapsed: 5317.819102048874 seconds
----------------------
Iteration 13000: train loss 0.1417018175125122
Time Elapsed: 5848.137082576752 seconds
----------------------
Iteration 14000: train loss 0.13926757872104645
Time Elapsed: 6238.944685459137 seconds
----------------------
Iteration 15000: train loss 0.14091581106185913
Time Elapsed: 6617.137130975723 seconds
----------------------
Iteration 16000: train loss 0.13938488066196442
Time Elapsed: 6993.011977434158 seconds
----------------------
Iteration 17000: train loss 0.13915009796619415
Time Elapsed: 7365.753441810608 seconds
----------------------
Iteration 18000: train loss 0.13628077507019043
Time Elapsed: 7746.385533332825 seconds
----------------------
Iteration 19000: train loss 0.143539160490036
Time Elapsed: 8124.784602403641 seconds
----------------------
Iteration 20000: train loss 0.14154307544231415
Time Elapsed: 8497.999198198318 seconds
----------------------
Iteration 21000: train loss 0.1427953839302063
Time Elapsed: 8873.80286359787 seconds
----------------------
Iteration 22000: train loss 0.14059235155582428
Time Elapsed: 9245.829558610916 seconds
----------------------
Iteration 23000: train loss 0.1397273987531662
Time Elapsed: 9626.873769283295 seconds
----------------------
Iteration 24000: train loss 0.14075049757957458
Time Elapsed: 10000.698282003403 seconds
----------------------
Iteration 25000: train loss 0.143774151802063
Time Elapsed: 10379.361604690552 seconds
----------------------
Iteration 26000: train loss 0.1405758559703827
Time Elapsed: 10850.103506803513 seconds
----------------------

Epoch 2/5 loss: 0.14



Iteration 0: train loss 0.14230649173259735
Time Elapsed: 3.2797293663024902 seconds
```

```
----------------------
Iteration 1000: train loss 0.14046867191791534
Time Elapsed: 427.59963631629944 seconds
----------------------
Iteration 2000: train loss 0.14118683338165283
Time Elapsed: 865.9122269153595 seconds
----------------------
Iteration 3000: train loss 0.13967327773571014
Time Elapsed: 1297.7534620761871 seconds
----------------------
Iteration 4000: train loss 0.1395128071308136
Time Elapsed: 1735.960110425949 seconds
----------------------
Iteration 5000: train loss 0.14029833674430847
Time Elapsed: 2201.578524827957 seconds
----------------------
Iteration 6000: train loss 0.13769793510437012
Time Elapsed: 2646.6980974674225 seconds
----------------------
Iteration 7000: train loss 0.14209549129009247
Time Elapsed: 3183.757510662079 seconds
----------------------
Iteration 8000: train loss 0.141939178109169
Time Elapsed: 3590.8853464126587 seconds
----------------------
Iteration 9000: train loss 0.1397198885679245
Time Elapsed: 4063.868819475174 seconds
----------------------
Iteration 10000: train loss 0.1406521052122116
Time Elapsed: 4468.993368864059 seconds
----------------------
Iteration 11000: train loss 0.14215531945228577
Time Elapsed: 4855.493980646133 seconds
----------------------
Iteration 12000: train loss 0.14571499824523926
Time Elapsed: 5227.557518959045 seconds
----------------------
Iteration 13000: train loss 0.1400294303894043
Time Elapsed: 5604.141076803207 seconds
----------------------
Iteration 14000: train loss 0.1402643918991089
Time Elapsed: 5982.2826561927795 seconds
----------------------
Iteration 15000: train loss 0.1412220001220703
Time Elapsed: 6368.187423944473 seconds
----------------------
Iteration 16000: train loss 0.14206013083457947
Time Elapsed: 6765.824393749237 seconds
----------------------
Iteration 17000: train loss 0.14092840254306793
Time Elapsed: 7142.28927731514 seconds
----------------------
Iteration 18000: train loss 0.13903546333312988
Time Elapsed: 7530.887738227844 seconds
----------------------
Iteration 19000: train loss 0.14067259430885315
Time Elapsed: 7928.022391080856 seconds
----------------------
Iteration 20000: train loss 0.1408890038728714
Time Elapsed: 8303.78990983963 seconds
----------------------
```

```
Iteration 21000: train loss 0.13912290334701538
Time Elapsed: 8688.493919372559 seconds
----------------------
Iteration 22000: train loss 0.14106181263923645
Time Elapsed: 9102.556811094284 seconds
----------------------
Iteration 23000: train loss 0.13867609202861786
Time Elapsed: 9541.839179754257 seconds
----------------------
Iteration 24000: train loss 0.14323124289512634
Time Elapsed: 9909.784101963043 seconds
----------------------
Iteration 25000: train loss 0.1409144103527069
Time Elapsed: 10275.991936206818 seconds
----------------------
Iteration 26000: train loss 0.13815458118915558
Time Elapsed: 10637.631856441498 seconds
----------------------
Epoch 3/5 loss: 0.14


Iteration 0: train loss 0.14076286554336548
Time Elapsed: 2.6273787021636963 seconds
----------------------
Iteration 25000: train loss 0.1409144103527069
Time Elapsed: 10275.991936206818 seconds
----------------------
Iteration 26000: train loss 0.13815458118915558
Time Elapsed: 10637.631856441498 seconds
----------------------
Epoch 3/5 loss: 0.14


Iteration 0: train loss 0.14076286554336548
Time Elapsed: 2.6273787021636963 seconds
----------------------
Iteration 1000: train loss 0.14225144684314728
Time Elapsed: 364.0714337825775 seconds
----------------------
Iteration 2000: train loss 0.1402924805879593
Time Elapsed: 723.0902264118195 seconds
----------------------
Iteration 3000: train loss 0.14215335249900818
Time Elapsed: 1083.5055882930756 seconds
----------------------
Iteration 4000: train loss 0.14263972640037537
Time Elapsed: 1441.438039779663 seconds
----------------------
Iteration 5000: train loss 0.1423766314983368
Time Elapsed: 1801.8044893741608 seconds
----------------------
Iteration 6000: train loss 0.14168091118335724
Time Elapsed: 2160.047431945801 seconds
----------------------
Iteration 7000: train loss 0.13670742511749268
Time Elapsed: 2517.8502576351166 seconds
----------------------
Iteration 8000: train loss 0.1366393268108368
Time Elapsed: 2875.488919019699 seconds
----------------------
Iteration 9000: train loss 0.1421920210123062
```

```
Time Elapsed: 3234.008144378662 seconds
----------------------
Iteration 10000: train loss 0.13721930980682373
Time Elapsed: 3591.7157940864563 seconds
----------------------
Iteration 11000: train loss 0.14233118295669556
Time Elapsed: 3950.9266481399536 seconds
----------------------
Iteration 12000: train loss 0.13793689012527466
Time Elapsed: 4309.980562448502 seconds
----------------------
Iteration 13000: train loss 0.1333065927028656
Time Elapsed: 4675.496218681335 seconds
----------------------
Iteration 14000: train loss 0.14200982451438904
Time Elapsed: 5042.962399721146 seconds
----------------------
Iteration 15000: train loss 0.1431654691696167
Time Elapsed: 5402.57327914238 seconds
----------------------
Iteration 16000: train loss 0.14178785681724548
Time Elapsed: 5761.523490190506 seconds
----------------------
Iteration 17000: train loss 0.14224034547805786
Time Elapsed: 6120.8950934410095 seconds
----------------------
Iteration 18000: train loss 0.13997632265090942
Time Elapsed: 6492.152825832367 seconds
----------------------
Iteration 19000: train loss 0.1416899561882019
Time Elapsed: 6853.009024143219 seconds
----------------------
Iteration 20000: train loss 0.14050476253032684
Time Elapsed: 7212.794869184494 seconds
----------------------
Iteration 21000: train loss 0.13802208006381989
Time Elapsed: 7575.417325258255 seconds
----------------------
Iteration 22000: train loss 0.13910914957523346
Time Elapsed: 7935.391286611557 seconds
----------------------
Iteration 23000: train loss 0.13988158106803894
Time Elapsed: 8302.859107732773 seconds
----------------------
Iteration 24000: train loss 0.13844701647758484
Time Elapsed: 8676.489864349365 seconds
----------------------
Iteration 25000: train loss 0.13971760869026184
Time Elapsed: 9049.52786064148 seconds
----------------------
Iteration 26000: train loss 0.14092251658439636
Time Elapsed: 9422.370479345322 seconds
----------------------

Epoch 4/5 loss: 0.14


Iteration 0: train loss 0.14375083148479462
Time Elapsed: 1.8132450580596924 seconds
----------------------
Iteration 1000: train loss 0.14266356825828552
```

```
Time Elapsed: 378.12688636779785 seconds
----------------------
Iteration 2000: train loss 0.13987651467323303
Time Elapsed: 750.8175268173218 seconds
----------------------
Iteration 3000: train loss 0.1403343379497528
Time Elapsed: 1122.0159420967102 seconds
----------------------
Iteration 4000: train loss 0.14143559336662292
Time Elapsed: 1492.5349776744843 seconds
----------------------
Iteration 5000: train loss 0.14119303226470947
Time Elapsed: 1862.5048587322235 seconds
----------------------
Iteration 6000: train loss 0.14255750179290771
Time Elapsed: 2232.581345796585 seconds
----------------------
Iteration 7000: train loss 0.1392657458782196
Time Elapsed: 2603.45166182518 seconds
----------------------
Iteration 8000: train loss 0.14294540882110596
Time Elapsed: 2988.97735953331 seconds
----------------------
Iteration 9000: train loss 0.14005693793296814
Time Elapsed: 3359.426027774811 seconds
----------------------
Iteration 10000: train loss 0.13867586851119995
Time Elapsed: 3729.6206040382385 seconds
----------------------
Iteration 11000: train loss 0.13798613846302032
Time Elapsed: 4102.208824634552 seconds
----------------------
Iteration 12000: train loss 0.13896112143993378
Time Elapsed: 4475.889963150024 seconds
----------------------
Iteration 13000: train loss 0.1396407037973404
Time Elapsed: 4849.203073978424 seconds
----------------------
Iteration 14000: train loss 0.14337459206581116
Time Elapsed: 5242.488445043564 seconds
----------------------
Iteration 15000: train loss 0.14029014110565186
Time Elapsed: 5620.69154047966 seconds
----------------------
Iteration 16000: train loss 0.14033591747283936
Time Elapsed: 6030.106435537338 seconds
----------------------
Iteration 17000: train loss 0.1372278332710266
Time Elapsed: 6424.6762952804565 seconds
----------------------
Iteration 18000: train loss 0.1406635046005249
Time Elapsed: 6807.158838272095 seconds
----------------------
Iteration 19000: train loss 0.14044691622257233
Time Elapsed: 7186.720580339432 seconds
----------------------
Iteration 20000: train loss 0.13896596431732178
Time Elapsed: 7566.396906375885 seconds
----------------------
Iteration 21000: train loss 0.13737471401691437
Time Elapsed: 7953.720011949539 seconds
```

```
-----------------------
Iteration 22000: train loss 0.1397884488105774
Time Elapsed: 8336.67820262909 seconds
-----------------------
Iteration 23000: train loss 0.14313817024230957
Time Elapsed: 8719.253033399582 seconds
-----------------------
Iteration 24000: train loss 0.139867901802063
Time Elapsed: 9107.383109092712 seconds
-----------------------
Iteration 25000: train loss 0.13503572344779968
Time Elapsed: 9500.890689134598 seconds
-----------------------
Iteration 26000: train loss 0.13769620656967163
Time Elapsed: 9888.158396482468 seconds
-----------------------
Epoch 5/5 loss: 0.14
Total Training Time: 52038.67340183258 seconds
st122314@a5cc46d8d889:~/Lab11$
```

```
Epoch 1/5 loss: 0.14
Epoch 2/5 loss: 0.14
Epoch 3/5 loss: 0.14
Epoch 4/5 loss: 0.14
Epoch 5/5 loss: 0.14

Total Training Time: 52038.67340183258 seconds
```

According to the above train loss result, the pre-trained model I applied does not seem learning.

2.1.5 Testing

In [ ]:

```python
test(model, device, test_loader)
print(len(val_loader))

# print("Loading Model Checkpoint ...")
# model.load_state_dict(torch.load('trained-vit.pt'))

test(model, device, val_loader)
```

The following pic shows the test result.

```
st122314@a5cc46d8d889:~/Lab11$ python3 pretrained_ViT.py
Loading ImageNet train ...
Loading ImageNet val ...
Loading Model Checkpoint ...
Test loss: 145.73
Test accuracy: 0.58%
```

2.2 fine-tuning with CIFAR10

In [ ]:

```
import numpy as np
import time
import torch
import torch.nn as nn
from torch.nn import CrossEntropyLoss
from torch.optim import Adam
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.datasets.mnist import MNIST
from torchvision import transforms
from vit_pytorch import ViT

device = torch.device("cuda:3" if torch.cuda.is_available() else "cpu")

# Load data.

print("Loading CIFAR10 train ...")
train_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))]
    )

train_set = datasets.CIFAR100(root='/Dataset/CIFAR10',
                            train = True, download= True,
                            transform=train_transform)


val_set = datasets.CIFAR100(root='/Dataset/CIFAR10',
                            train = False, download= True,
                            transform=train_transform)


train_loader = DataLoader(train_set, shuffle=True, batch_size=48, num_workers=4)
print(len(train_loader))
val_loader = DataLoader(val_set, shuffle=True, batch_size=48, num_workers=4)

model = ViT(
    image_size = 224,
    patch_size = 16,
    num_classes = 1000,
    dim = 1024,
    depth = 6,
    heads = 16,
    mlp_dim = 2048,
    dropout = 0.1,
    emb_dropout = 0.1
).to(device)


print("Loading Model Checkpoint ...")
model.load_state_dict(torch.load('trained-pytorch-vit-imagenet.pt'))

def train(model, optimizer, N_EPOCHS, device, train_loader):
    criterion = CrossEntropyLoss()
    for epoch in range(N_EPOCHS):
```

```
        train_loss = 0.0
        epoch_start = time.time()
        for i, (batch) in enumerate(train_loader):
            model = model.to(device)
            x, y = batch
            x = x.to(device)
            y = y.to(device)
            # print('x shape: ', x.shape)
            y_hat = model(x)
            loss = criterion(y_hat, y) / len(x)

            train_loss += loss.item()

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            if (i % 100 == 0):
                print(f'Iteration {i}: train loss {loss.item()}')
                print(f'Time Elapsed: {time.time() - epoch_start} seconds')
                print("----------------------")

            if (i % 20000 == 0):
                torch.save(model.state_dict(), f'model_epoch_{i}.pt')
        print(f"Epoch {epoch + 1}/{N_EPOCHS} loss: {train_loss/ len(train_loader):.2


def test(model, device, test_loader):
    criterion = nn.CrossEntropyLoss()
    correct, total = 0, 0
    test_loss = 0.0

    model.eval()
    with torch.no_grad():
        for batch in test_loader:
            x, y = batch
            x = x.to(device)
            y = y.to(device)

            y_hat = model(x)
            loss = criterion(y_hat, y) / len(x)
            test_loss += loss

            correct += torch.sum(torch.argmax(y_hat, dim=1) == y).item()
            total += len(x)
        print(f"Test loss: {test_loss:.2f}")
        print(f"Test accuracy: {correct / total * 100:.2f}%")


N_EPOCHS = 5
LR = 0.001
optimizer = Adam(model.parameters(), lr=LR)

model.mlp_head[1] = nn.Linear(in_features = 1024, out_features = 10, bias = True)

start_training = time.time()
train(model, optimizer, N_EPOCHS, device, train_loader)

print(f"Total Training Time: {time.time() - start_training} seconds")

test(model, device, val_loader)
```

The following is the testing result on CIFAR10.

```
st122314@3bc5602f36d6:~/Lab11$ python3 finetune.py
Loading CIFAR10 train ...
1042
Loading Model Checkpoint ...
Test loss: 156.50
Test accuracy: 0.04%
```

In [ ]:

In [ ]:

In [ ]: