```
import time
import copy
import glob
import torch
import torch.nn as nn
import torchvision
from torch.utils.data import DataLoader
from torch.utils.data import dataset
from matplotlib import pyplot as plt
import torchvision.models as models
import torch.optim as optim
import os
import numpy as np
```

```
transform = torchvision.transforms.Compose([
    torchvision.transforms.RandomRotation(10),
    torchvision.transforms.ToTensor()
])
mnist_train = torchvision.datasets.MNIST('./data', train = True, download = True, transform =
train_loader = DataLoader(mnist_train, batch_size=1000, shuffle=True)
```

```
mnist_test = torchvision.datasets.MNIST('./data', train = False, download = True, transform =
test_loader = DataLoader(mnist_test, batch_size=1000, shuffle=True)
```

```
mnist_train
```

```
Dataset MNIST
    Number of datapoints: 60000
    Root location: ./data
    Split: Train
    StandardTransform
Transform: Compose(
               RandomRotation(degrees=[-10.0, 10.0], interpolation=nearest,
expand=False, fill=0)
               ToTensor()
           )
```

Сургалтын датасет нь 60000

```
len(mnist_train)
```

```
60000
```

```
resnet18 = models.resnet18(pretrained=True)
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:209: UserWarning: Th
  f"The parameter '{pretrained_param}' is deprecated since 0.13 and will be removed in 0
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWarning: Ar
  warnings.warn(msg)
```

```
resnet18.conv1 = nn.Conv2d(1, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=Fal
resnet18.fc = nn.Linear(512,10)
```

```
print(resnet18)
```

```
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_st
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_st
      )
    )
    (layer3): Sequential(

      (0): BasicBlock(
        (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), b
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (downsample): Sequential(
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_st
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_st
      )
    )
    (layer4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), b
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=512, out_features=10, bias=True)
  )
```

```
def set_parameter_requires_grad(model, feature_extracting=True):
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad = True

set_parameter_requires_grad(resnet18)
```

```
def train_model(model, train_loader, test_loader, criterion, optimizer, device, scheduler, nu
    since = time.time()

    acc_history = []
    loss_history = []

    best_acc = 0.0
    model.to(device)
    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        running_loss = 0.0
        running_corrects = 0

        model.train()
        # Iterate over data.
        for inputs, labels in train_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            # zero the parameter gradients
            optimizer.zero_grad()

            # forward
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            _, preds = torch.max(outputs, 1)

            # backward
            loss.backward()
            optimizer.step()

            # statistics
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)

        epoch_loss = running_loss / len(train_loader.dataset)
        epoch_acc = running_corrects.double() / len(train_loader.dataset)

        print('Loss: {:.4f} Acc: {:.4f}'.format(epoch_loss, epoch_acc))
        test_model(model, test_loader, criterion, device, scheduler)

        if epoch_acc > best_acc:
            best_acc = epoch_acc

        acc_history.append(epoch_acc.item())
        loss_history.append(epoch_loss)

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60, time_elapsed % 60
    print('Best Acc: {:4f}'.format(best_acc))

    return acc_history, loss_history
def test_model(model, test_loader, criterion, device, scheduler):
  model.eval()
  best_acc = 0.0
  test_loss = 0.0
  test_corrects = 0
  for inputs, labels in test_loader:
    inputs = inputs.to(device)
    labels = labels.to(device)
    outputs = model(inputs)
    _, preds = torch.max(outputs, 1)
    loss = criterion(outputs, labels)
    test_loss += loss.item() * inputs.size(0)
    test_corrects += torch.sum(preds == labels.data)
  epoch_acc = test_corrects.double() / len(test_loader.dataset)
  scheduler.step(test_loss)
  print('Test Acc: {:.4f}'.format(epoch_acc))

  if epoch_acc > best_acc:
      best_acc = epoch_acc
      torch.save(model.state_dict(), './best_model.pt')
```

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

# Set up the loss function
# Setup the loss function
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(resnet18.parameters())
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min')
# Train model
train_acc_hist, train_loss_hist = train_model(resnet18, train_loader, test_loader, criterion,
```

```
Epoch 0/24
----------
Loss: 0.2783 Acc: 0.9123
Test Acc: 0.9698
Epoch 1/24
----------
Loss: 0.0688 Acc: 0.9794
Test Acc: 0.9757
Epoch 2/24
----------
Loss: 0.0496 Acc: 0.9844
Test Acc: 0.9796
Epoch 3/24
----------
Loss: 0.0428 Acc: 0.9867
Test Acc: 0.9824
Epoch 4/24
----------
Loss: 0.0370 Acc: 0.9886
Test Acc: 0.9865
Epoch 5/24
----------
Loss: 0.0303 Acc: 0.9904
Test Acc: 0.9867
Epoch 6/24
----------
Loss: 0.0269 Acc: 0.9916
Test Acc: 0.9852
Epoch 7/24
----------
Loss: 0.0256 Acc: 0.9923
Test Acc: 0.9862
Epoch 8/24
----------
Loss: 0.0225 Acc: 0.9930
Test Acc: 0.9864
Epoch 9/24
----------
Loss: 0.0209 Acc: 0.9929
Test Acc: 0.9885
Epoch 10/24
----------
Loss: 0.0204 Acc: 0.9935
Test Acc: 0.9904
Epoch 11/24
----------
Loss: 0.0184 Acc: 0.9942
Test Acc: 0.9857
Epoch 12/24
```

```
----------
Loss: 0.0179 Acc: 0.9941
Test Acc: 0.9897
Epoch 13/24
----------
Loss: 0.0167 Acc: 0.9945
Test Acc: 0.9796
Epoch 14/24
----------
```