# mini project 3

ResNet 모델을 전이학습으로 하는 방법을 사용하여, 현업에서 분류가 필요한 영상들을 학습 데이터로 하여 모델을 학습시키기

산업인공지능학과

2020254018

강 윤 구

# 1. 실습 프로그램#1 (google drive에 Hardware_data를 만들어 사진 파일 사용)

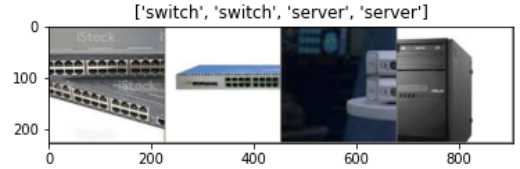
```
%matplotlib inline
from __future__ import print_function, division

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import numpy as np
import torchvision
from  torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import time
import os
import copy
plt.ion()   #interactive mode

from google.colab import drive
drive.mount('/content/gdrive')

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

data_dir = '/content/gdrive/MyDrive/Hardware_data'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), data_transforms[x]) for x in ['train', 'val']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=4, shuffle=True,
                                              num_workers=4) for x in ['train', 'val']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
class_names = image_datasets['train'].classes
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

def imshow(inp, title=None):
  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)
```

```
inputs, classes = next(iter(dataloaders['train']))
out = torchvision.utils.make_grid(inputs)

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

def train_model(model, criterion, optimizer, scheduler, num_epochs = 25):
  since = time.time()
  best_model_wts = copy.deepcopy(model.state_dict())
  best_acc=0.0

  for epoch in range(num_epochs):
    print('Epoch {}/{}'.format(epoch, num_epochs - 1))
    print('-' * 10)
    for phase in ['train', 'val']:
      if phase == 'train':
        scheduler.step()
        model.train()
      else:
        model.eval()
      running_loss = 0.0
      running_corrects = 0
      for inputs, labels in dataloaders[phase]:
        inputs = inputs.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        with torch.set_grad_enabled(phase == 'train'):
          outputs = model(inputs)
          _, preds = torch.max(outputs, 1)
          loss = criterion(outputs, labels)
          if phase == 'train':
            loss.backward()
            optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)
      epoch_loss = running_loss / dataset_sizes[phase]
      epoch_acc = running_corrects.double() / dataset_sizes[phase]
      print('{} Loss: {:.4f} Acc: {:.4f}'.format(phase, epoch_loss, epoch_acc))
      if phase == 'val' and epoch_acc > best_acc:
        best_acc = epoch_acc
        best_model_wts = copy.deepcopy(model.state_dict())
    print()

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

def visualize_model(model, num_images=6):
  was_training = model.training
  model.eval()
  images_so_far = 0
  fig = plt.figure()
```

```
  with torch.no_grad():
    for i, (inputs, labels) in enumerate(dataloaders['val']):
      inputs = inputs.to(device)
      labels = labels.to(device)

      outputs = model(inputs)
      _, preds = torch.max(outputs, 1)

      for j in range(inputs.size()[0]):
        images_so_far += 1
        ax = plt.subplot(num_images//2, 2, images_so_far)
        ax.axis('off')
        ax.set_title('predicted: {}'.format(class_names[preds[j]]))
        imshow(inputs.cpu().data[j])

        if images_so_far == num_images:
          model.train(mode=was_training)
          return
    model.train(mode=was_training)

model_ft = models.resnet18(pretrained=True)  #사전 학습된 ResNet18 가져오기
num_ftrs = model_ft.fc.in_features  #모델에서 feature extraction 후 FC 층에 입력되는 특징수
model_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)  # cpu나 GPU에 model_ft를 할당
criterion = nn.CrossEntropyLoss()

#모든 파라미터를 학습
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

#매 7 에포크 마다 학습율 0.1배 감소
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler, num_epochs=25)

visualize_model(model_ft)
```

# 2. 결과#1

```
Mounted at /content/gdrive
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserWarning: Thi
  cpuset_checked))
```




```
Downloading: "https://download.pytorch.org/models/resnet18-5c106cde.pth" to /root/.cache/t
100%  44.7M/44.7M [01:16<00:00, 610kB/s]

Epoch 0/24
----------
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Detect
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
train Loss: 0.6700 Acc: 0.6667
val Loss: 0.2687 Acc: 0.9000

Epoch 1/24
----------
train Loss: 0.7136 Acc: 0.7595
val Loss: 0.7407 Acc: 0.7167

Epoch 2/24
----------
train Loss: 0.5706 Acc: 0.7975
val Loss: 0.1723 Acc: 0.9167

                    .
                    .
                    .

Epoch 23/24
----------
train Loss: 0.2727 Acc: 0.8987
val Loss: 0.1258 Acc: 0.9833

Epoch 24/24
----------
train Loss: 0.2376 Acc: 0.8861
val Loss: 0.1918 Acc: 0.8667

Training complete in 1m 5s
Best val Acc: 0.983333
```

predicted: server

predicted: server

predicted: server

predicted: switch

predicted: switch

predicted: switch

# 3. 실습 프로그램#2

```
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters():
  param.requires_grad = False  #사전 학습된 모델의 가중치를 상수로 고정. 학습시키지 않음

#새로 생성된 모듈의 파라미터는 기본적으로 requires_grad=True
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)

model_conv = model_conv.to(device)
criterion = nn.CrossEntropyLoss()

#마지막 층의 파라미터만 학습
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
model_conv = train_model(model_conv, criterion, optimizer_conv, exp_lr_scheduler, num_epochs=25)

visualize_model(model_conv)

plt.ioff()
plt.show()
```

실행내역

https://github.com/Yunkoo-GIT/Programming/blob/main/20210526.ipynb

# 4. 결과#2

```
Epoch 24/24
----------
train Loss: 0.3532 Acc: 0.8523
val Loss: 0.5054 Acc: 0.7667

Training complete in 0m 41s
Best val Acc: 0.816667
```

predicted: server

predicted: server

predicted: switch

predicted: server

predicted: switch

predicted: switch