```
# 필요한 라이브러리 import
import os
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.utils import data
import torchvision.datasets as datasets
import torchvision.transforms as transforms
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # matplotlib
import pandas as pd # pandas
import natsort
import cv2
import os
from torch.utils.tensorboard import SummaryWriter
```

```
from google.colab import drive

drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
!ls -al /content/gdrive/MyDrive/project
```

```
total 5270717
drwx------ 8 root root       4096 Dec 12 08:11 class
-rw------- 1 root root     194650 Dec  7 13:29 ILSVRC2010_ground_truth.txt
-rw------- 1 root root 5397002240 Dec  7 13:46 ILSVRC2010_images_.tar
drwx------ 2 root root       4096 Dec 12 18:49 input
drwx------ 8 root root       4096 Dec 12 20:04 output
drwx------ 2 root root       4096 Dec 10 03:56 val
```

```
os.chdir('/content/gdrive/MyDrive/project')
```

```
#압축해제
#!tar -xvf /content/gdrive/MyDrive/project/ILSVRC2010_images_.tar
```

```
!ls -al /content/gdrive/MyDrive/project
```

```
total 5270717
drwx------ 8 root root       4096 Dec 12 08:11 class
-rw------- 1 root root     194650 Dec  7 13:29 ILSVRC2010_ground_truth.txt
-rw------- 1 root root 5397002240 Dec  7 13:46 ILSVRC2010_images_.tar
drwx------ 2 root root       4096 Dec 12 18:49 input
drwx------ 8 root root       4096 Dec 12 20:04 output
drwx------ 2 root root       4096 Dec 10 03:56 val
```

```
# 결과변수 불러오기
y = pd.read_csv('/content/gdrive/MyDrive/project/ILSVRC2010_ground_truth.txt', header=None, names=['answer'])
y
```

	answer
0	78
1	854
2	435
3	541
4	973
...	...
49995	467
49996	646
49997	68
49998	93
49999	561

50000 rows × 1 columns

```
# 경로불러오기
image_path = '/content/gdrive/MyDrive/project/val/'

img_list = natsort.natsorted(os.listdir(image_path))
img_list_jpeg = [img for img in img_list if img.endswith(".JPEG")]
```

```
print("img_list_jpeg: {}".format(img_list_jpeg))
```

```
img_list_jpeg: ['ILSVRC2010_val_00000001.JPEG', 'ILSVRC2010_val_00000002.JPEG', 'ILSVRC2010_val_00000003.JPEG', 'ILSVRC2010_val_00000004.JPEG',
```

```
import os
from glob import glob
import shutil
from sklearn.model_selection import train_test_split

def batch_move_files(file_list, source_path, destination_path):
  for file in file_list:
    image=file.split('/')[-1]+'.JPEG'
    shutil.copy(os.path.join(source_path,image),destination_path)
  return

test_dir="/content/gdrive/MyDrive/project/output/test"
train_dir="/content/gdrive/MyDrive/project/output/train"
valid_dir="/content/gdrive/MyDrive/project/output/valid"


#images 리스트로 변환
image_files=glob("/content/gdrive/MyDrive/project/class/78/*.JPEG")
images=[name.replace(".JPEG","") for name in image_files]

train_names, test_names =train_test_split(images, test_size=0.2,
                                          random_state=42, shuffle=True)
valid_names, test_names=train_test_split(test_names, test_size=0.5,
                                       random_state=42, shuffle=True)

source_dir="/content/gdrive/MyDrive/project/class/78"

batch_move_files(train_names, source_dir, train_dir)
batch_move_files(test_names, source_dir, test_dir)
batch_move_files(valid_names, source_dir, valid_dir)


#images 리스트로 변환
image_files=glob("/content/gdrive/MyDrive/project/class/250/*.JPEG")
images=[name.replace(".JPEG","") for name in image_files]

train_names, test_names =train_test_split(images, test_size=0.2,
                                          random_state=42, shuffle=True)
valid_names, test_names=train_test_split(test_names, test_size=0.5,
                                       random_state=42, shuffle=True)

source_dir="/content/gdrive/MyDrive/project/class/250"

batch_move_files(train_names, source_dir, train_dir)
batch_move_files(test_names, source_dir, test_dir)
batch_move_files(valid_names, source_dir, valid_dir)


#images 리스트로 변환
image_files=glob("/content/gdrive/MyDrive/project/class/438/*.JPEG")
images=[name.replace(".JPEG","") for name in image_files]

train_names, test_names =train_test_split(images, test_size=0.2,
                                          random_state=42, shuffle=True)
valid_names, test_names=train_test_split(test_names, test_size=0.5,
                                       random_state=42, shuffle=True)

source_dir="/content/gdrive/MyDrive/project/class/438"

batch_move_files(train_names, source_dir, train_dir)
batch_move_files(test_names, source_dir, test_dir)
batch_move_files(valid_names, source_dir, valid_dir)


#images 리스트로 변환
image_files=glob("/content/gdrive/MyDrive/project/class/733/*.JPEG")
images=[name.replace(".JPEG","") for name in image_files]

train_names, test_names =train_test_split(images, test_size=0.2,
                                          random_state=42, shuffle=True)
valid_names, test_names=train_test_split(test_names, test_size=0.5,
                                       random_state=42, shuffle=True)

source_dir="/content/gdrive/MyDrive/project/class/733"

batch_move_files(train_names, source_dir, train_dir)
batch_move_files(test_names, source_dir, test_dir)
batch_move_files(valid_names, source_dir, valid_dir)


#images 리스트로 변환
image_files=glob("/content/gdrive/MyDrive/project/class/831/*.JPEG")
images=[name.replace(".JPEG","") for name in image_files]

train_names, test_names =train_test_split(images, test_size=0.2,
                                          random_state=42, shuffle=True)
```

```
valid_names, test_names=train_test_split(test_names, test_size=0.5,
                                         random_state=42, shuffle=True)

source_dir="/content/gdrive/MyDrive/project/class/831"

batch_move_files(train_names, source_dir, train_dir)
batch_move_files(test_names, source_dir, test_dir)
batch_move_files(valid_names, source_dir, valid_dir)
```

```
train_names[0]
```

```
'/content/gdrive/MyDrive/project/class/831/ILSVRC2010_val_00023009'
```

```
!cd '/content/gdrive/MyDrive/project'
```

```
!ls -al
```

```
total 5270717
drwx------ 8 root root       4096 Dec 12 08:11 class
-rw------- 1 root root     194650 Dec  7 13:29 ILSVRC2010_ground_truth.txt
-rw------- 1 root root 5397002240 Dec  7 13:46 ILSVRC2010_images_.tar
drwx------ 2 root root       4096 Dec 12 18:49 input
drwx------ 8 root root       4096 Dec 12 20:04 output
drwx------ 2 root root       4096 Dec 10 03:56 val
```

```
!mkdir input
```

```
mkdir: cannot create directory 'input': File exists
```

```
!ls -al
```

```
total 5270717
drwx------ 8 root root       4096 Dec 12 08:11 class
-rw------- 1 root root     194650 Dec  7 13:29 ILSVRC2010_ground_truth.txt
-rw------- 1 root root 5397002240 Dec  7 13:46 ILSVRC2010_images_.tar
drwx------ 2 root root       4096 Dec 12 18:49 input
drwx------ 8 root root       4096 Dec 12 20:04 output
drwx------ 2 root root       4096 Dec 10 03:56 val
```

```
#78, 250, 438, 733, 831
!cp -r /content/gdrive/MyDrive/project/class/78/* /content/gdrive/MyDrive/project/input
!cp -r /content/gdrive/MyDrive/project/class/250/* /content/gdrive/MyDrive/project/input
!cp -r /content/gdrive/MyDrive/project/class/438/* /content/gdrive/MyDrive/project/input
!cp -r /content/gdrive/MyDrive/project/class/733/* /content/gdrive/MyDrive/project/input
!cp -r /content/gdrive/MyDrive/project/class/831/* /content/gdrive/MyDrive/project/input
```

```
image2_files=glob("/content/gdrive/MyDrive/project/input/*.JPEG")
image2_files[0]
```

```
'/content/gdrive/MyDrive/project/input/ILSVRC2010_val_00000001.JPEG'
```

```
img_list = os.listdir('/content/gdrive/MyDrive/project/input')
```

```
# 이미지 그리기
plt.figure(figsize=(40, 20))

for i in img_list:
  path = '/content/gdrive/MyDrive/project/input/' + i

  img = cv2.imread(path)

  img_rgb=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

  plt.subplot(25, 10, img_list.index(i)+1)
  plt.imshow(img_rgb)
```

```
img.shape
```

```
(395, 500, 3)
```

```
# pytorch device 정의하기
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# model parameters 정의하기
NUM_EPOCHS = 90
BATCH_SIZE = 128
MOMENTUM = 0.9
LR_DECAY = 0.0005
LR_INIT = 0.01
IMAGE_DIM = 227 # pixels
NUM_CLASSES = 5
DEVICE_IDS = [0, 1, 2, 3]

# data directory 지정하기
INPUT_ROOT_DIR = '/content/gdrive/MyDrive/project/input'
TRAIN_IMG_DIR = train_dir
OUTPUT_DIR = '/content/gdrive/MyDrive/project/output'
LOG_DIR = OUTPUT_DIR + '/tblogs'  # tensorboard logs
CHECKPOINT_DIR = OUTPUT_DIR + '/models'  # model checkpoints

# checkpoint 경로 directory 만들기
os.makedirs(CHECKPOINT_DIR, exist_ok=True)


class AlexNet(nn.Module):
  def __init__(self, num_classes=5):
    super().__init__()
    ##### CNN layers
    self.net = nn.Sequential(
        # conv1
        nn.Conv2d(in_channels=3, out_channels=96, kernel_size=11, stride=4),
        nn.ReLU(inplace=True),  # non-saturating function
        nn.LocalResponseNorm(size=5, alpha=0.0001, beta=0.75, k=2),  # 논문의 LRN 파라미터 그대로 지정
        nn.MaxPool2d(kernel_size=3, stride=2),
        # conv2
        nn.Conv2d(96, 256, kernel_size=5, padding=2),
        nn.ReLU(inplace=True),
        nn.LocalResponseNorm(size=5, alpha=0.0001, beta=0.75, k=2),
        nn.MaxPool2d(kernel_size=3, stride=2),
        # conv3
        nn.Conv2d(256, 384, 3, padding=1),
        nn.ReLU(inplace=True),
        # conv4
        nn.Conv2d(384, 384, 3, padding=1),
        nn.ReLU(inplace=True),
        # conv5
        nn.Conv2d(384, 256, 3, padding=1),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),

    )

    ##### FC layers
    self.classifier = nn.Sequential(
        # fc1
        nn.Dropout(p=0.5, inplace=True),
        nn.Linear(in_features=(256 * 6 * 6), out_features=4096),
        nn.ReLU(inplace=True),
        # fc2
        nn.Dropout(p=0.5, inplace=True),
        nn.Linear(4096, 4096),
        nn.ReLU(inplace=True),
        nn.Linear(4096, num_classes),
    )
    # bias, weight 초기화
    def init_bias_weights(self):
      for layer in self.net:
        if isinstance(layer, nn.Conv2d):
          nn.init.normal_(layer.weight, mean=0, std=0.01)   # weight 초기화
          nn.init.constant_(layer.bias, 0)   # bias 초기화
      # conv 2, 4, 5는 bias 1로 초기화
      nn.init.constant_(self.net[4].bias, 1)
      nn.init.constant_(self.net[10].bias, 1)
      nn.init.constant_(self.net[12].bias, 1)
    # modeling
    def forward(self, x):
      x = self.net(x)   # conv
      x = x.view(-1, 256*6*6)   # keras의 reshape (텐서 크기 2d 변경)
      return self.classifier(x)   # fc


if __name__ == '__main__':
    # seed value 출력하기
    seed = torch.initial_seed()
    print('Used seed : {}'.format(seed))

    tbwriter = SummaryWriter(log_dir=LOG_DIR)
    print('TensorboardX summary writer created')

    # model 생성하기
    alexnet = AlexNet(num_classes=NUM_CLASSES).to(device)
    # 다수의 GPU에서 train
    alexnet = torch.nn.parallel.DataParallel(alexnet, device_ids=DEVICE_IDS)
    print(alexnet)
    print('AlexNet created')

    # dataset과 data loader 생성하기
    dataset = datasets.ImageFolder(TRAIN_IMG_DIR, transforms.Compose([
        # transforms.RandomResizedCrop(IMAGE_DIM, scale=(0.9, 1.0), ratio=(0.9, 1.1)),
        transforms.CenterCrop(IMAGE_DIM),
        # transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ]))
    print('Dataset created')
    dataloader = data.DataLoader(
        dataset,
        shuffle=True,
        pin_memory=True,
        num_workers=8,
        drop_last=True,
        batch_size=BATCH_SIZE)
    print('Dataloader created')

    # optimizer 생성하기
    optimizer = optim.SGD(
        params=alexnet.parameters(),
        lr=LR_INIT,
        momentum=MOMENTUM,
        weight_decay=LR_DECAY)
    print('Optimizer created')

    # lr_scheduler로 LR 감소시키기 : 30epochs 마다 1/10
    lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=30, gamma=0.1)
    print('LR Scheduler created')

    # train 시작
    print('Starting training...')
    total_steps = 1
    for epoch in range(NUM_EPOCHS):
        lr_scheduler.step()
        for imgs, classes in dataloader:
            imgs, classes = imgs.to(device), classes.to(device)

            # loss 계산
            output = alexnet(imgs)
            loss = F.cross_entropy(output, classes)

            # parameter 갱신
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            # log the information and add to tensorboard
            # 정보를 기록하고 tensorboard에 추가하기
            if total_steps % 10 == 0:
                with torch.no_grad():
                    _, preds = torch.max(output, 1)
                    accuracy = torch.sum(preds == classes)

                    print('Epoch: {} ₩tStep: {} ₩tLoss: {:.4f} ₩tAcc: {}'
                        .format(epoch + 1, total_steps, loss.item(), accuracy.item()))
                    tbwriter.add_scalar('loss', loss.item(), total_steps)
                    tbwriter.add_scalar('accuracy', accuracy.item(), total_steps)
            # gradient values와 parameter average values 추력하기
            if total_steps % 100 == 0:
                with torch.no_grad():
                    # parameters의 grad 출력하고 저장하기
                    # parameters values 출력하고 저장하기
                    print('*' * 10)
                    for name, parameter in alexnet.named_parameters():
                        if parameter.grad is not None:
                            avg_grad = torch.mean(parameter.grad)
                            print('\t{} - grad_avg: {}'.format(name, avg_grad))
                            tbwriter.add_scalar('grad_avg/{}'.format(name), avg_grad.item(), total_steps)
                            tbwriter.add_histogram('grad/{}'.format(name),
                                    parameter.grad.cpu().numpy(), total_steps)
                        if parameter.data is not None:
                            avg_weight = torch.mean(parameter.data)
                            print('\t{} - param_avg: {}'.format(name, avg_weight))
                            tbwriter.add_histogram('weight/{}'.format(name),
                                    parameter.data.cpu().numpy(), total_steps)
                            tbwriter.add_scalar('weight_avg/{}'.format(name), avg_weight.item(), total_steps)

            total_steps += 1

        # checkpoints 저장하기
        checkpoint_path = os.path.join(CHECKPOINT_DIR, 'alexnet_states_e{}.pkl'.format(epoch + 1))
        state = {
            'epoch': epoch,
            'total_steps': total_steps,
            'optimizer': optimizer.state_dict(),
            'model': alexnet.state_dict(),
            'seed': seed,
        }
        torch.save(state, checkpoint_path)
```

```
Used seed : 70516186023684835
TensorboardX summary writer created
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-34-9f9125c0c480> in <module>
      8 
      9     # model 생성하기
---> 10     alexnet = AlexNet(num_classes=NUM_CLASSES).to(device)
     11     # 다수의 GPU에서 train
     12     alexnet = torch.nn.parallel.DataParallel(alexnet, device_ids=DEVICE_IDS)

                                    1 frames
/usr/local/lib/python3.8/dist-packages/torch/nn/modules/module.py in __getattr__(self, name)
   1263             if name in modules:
   1264                 return modules[name]
-> 1265         raise AttributeError("'{}' object has no attribute '{}'".format(
   1266             type(self).__name__, name))
   1267 

AttributeError: 'ReLU' object has no attribute 'nn'
```

SEARCH STACK OVERFLOW

Colab 유료 제품 - 여기에서 계약 취소