```
!pip install split-folders
!pip install split-folders[full]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: split-folders[full] in /usr/local/lib/python3.8/dist-packages (0.5.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from split-folders[full]) (4.64.1)
```

```
# 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
import splitfolders
```

```
from google.colab import drive

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

```
Mounted at /content/gdrive
```

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

```
total 5270737
drwx------ 2 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 14 02:48 input
drwx------ 2 root root       4096 Dec 14 16:00 model
drwx------ 2 root root       4096 Dec 14 10:43 output
drwx------ 2 root root       4096 Dec 13 12:56 output_1000class
drwx------ 2 root root       4096 Dec 14 10:57 output_5
drwx------ 2 root root       4096 Dec 13 12:56 output_5class
drwx------ 2 root root       4096 Dec 14 10:53 output_tvt
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 5270737
drwx------ 2 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 14 02:48 input
drwx------ 2 root root       4096 Dec 14 16:00 model
drwx------ 2 root root       4096 Dec 14 10:43 output
drwx------ 2 root root       4096 Dec 13 12:56 output_1000class
drwx------ 2 root root       4096 Dec 14 10:57 output_5
drwx------ 2 root root       4096 Dec 13 12:56 output_5class
drwx------ 2 root root       4096 Dec 14 10:53 output_tvt
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

```
#splitfolders.ratio("/content/gdrive/MyDrive/project/class", output="/content/gdrive/MyDrive/project/output_1000class", seed=1337, ratio=(.8, .1, .1))
```

```
#5classes : 78,250, 438, 733, 831
!ls -al /content/gdrive/MyDrive/project
```

```
total 5270737
drwx------ 2 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 14 02:48 input
drwx------ 2 root root       4096 Dec 14 16:00 model
drwx------ 2 root root       4096 Dec 14 10:43 output
drwx------ 2 root root       4096 Dec 13 12:56 output_1000class
drwx------ 2 root root       4096 Dec 14 10:57 output_5
drwx------ 2 root root       4096 Dec 13 12:56 output_5class
drwx------ 2 root root       4096 Dec 14 10:53 output_tvt
drwx------ 2 root root       4096 Dec 10 03:56 val
```

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

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

```
total 20
drwx------ 2 root root 4096 Dec 14 10:51 250
drwx------ 2 root root 4096 Dec 14 10:51 438
drwx------ 2 root root 4096 Dec 14 10:52 733
drwx------ 2 root root 4096 Dec 14 10:51 78
drwx------ 2 root root 4096 Dec 14 10:52 831
```

```
#splitfolders.ratio("/content/gdrive/MyDrive/project/output", output="/content/gdrive/MyDrive/project/output_tvt", seed=1234, ratio=(.8, .1, .1))
```

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

```
total 20
drwx------ 2 root root 4096 Dec 14 16:45 models
drwx------ 2 root root 4096 Dec 14 16:50 tblogs
drwx------ 2 root root 4096 Dec 14 10:53 test
drwx------ 2 root root 4096 Dec 14 10:53 train
drwx------ 2 root root 4096 Dec 14 10:53 val
```

```
# 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.data import DataLoader
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


# 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/output'
TRAIN_IMG_DIR = '/content/gdrive/MyDrive/project/output_tvt/train'
OUTPUT_DIR = '/content/gdrive/MyDrive/project/output_tvt'
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)

train_img_dir='/content/gdrive/MyDrive/project/output_tvt/train'
val_img_dir='/content/gdrive/MyDrive/project/output_tvt/valid'
test_img_dir='/content/gdrive/MyDrive/project/output_tvt/test'


dataset = datasets.ImageFolder(train_img_dir, transforms.Compose([
        transforms.RandomResizedCrop(227, scale=(0.9, 1.0), ratio=(0.9, 1.1)),
        transforms.CenterCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ]))


dataset.classes
```

```
['250', '438', '733', '78', '831']
```

```
dataset.class_to_idx
```

```
{'250': 0, '438': 1, '733': 2, '78': 3, '831': 4}
```

```
dataloader = data.DataLoader(
        dataset,
        shuffle=True,
        pin_memory=True,
        num_workers=8,
        drop_last=True,
        batch_size=BATCH_SIZE)
```

```
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarning: This DataLoader will create 8 worker processes in total.
  warnings.warn(_create_warning_msg(
```

```
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


    seed = torch.initial_seed()
    print('Used seed : {}'.format(seed))
```

```
Used seed : 6775789779613122915
```

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

```
TensorboardX summary writer created
```

```
    #alexnet
    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')
```

```
AlexNet(
  (net): Sequential(
    (0): Conv2d(3, 96, kernel_size=(11, 11), stride=(4, 4))
    (1): ReLU(inplace=True)
    (2): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=2)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): ReLU(inplace=True)
    (6): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=2)
    (7): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=True)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=True)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=5, bias=True)
  )
)
AlexNet created
```

```
    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')
```

```
Dataset created
```

```
    dataset
```

```
Dataset ImageFolder
    Number of datapoints: 200
    Root location: /content/gdrive/MyDrive/project/output_tvt/train
    StandardTransform
Transform: Compose(
               CenterCrop(size=(227, 227))
```

```
                    ToTensor()
                    Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
                )

dataloader = data.DataLoader(
dataset,
shuffle=True,
pin_memory=True,
num_workers=8,
drop_last=True,
batch_size=BATCH_SIZE)
print('Dataloader created')
```

```
Dataloader created
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarning: This DataLoader will create 8 worker processes in total.
  warnings.warn(_create_warning_msg(
```

```
dataloader
```

```
<torch.utils.data.dataloader.DataLoader at 0x7f49659a9d60>
```

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

```
Optimizer created
```

```
lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=30, gamma=0.1)
print('LR Scheduler created')
```

```
LR Scheduler created
```

```
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: {} WtStep: {} WtLoss: {:.4f} WtAcc: {}'
                        .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('Wt{} - 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('Wt{} - 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 : 6775789779613122915
TensorboardX summary writer created
AlexNet(
  (net): Sequential(
    (0): Conv2d(3, 96, kernel_size=(11, 11), stride=(4, 4))
    (1): ReLU(inplace=True)
    (2): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=2)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): ReLU(inplace=True)
    (6): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=2)
    (7): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=True)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=True)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
```

```
)
AlexNet created
Dataset created
Dataloader created
Optimizer created
LR Scheduler created
Starting training...
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-25-a1d4bccfc542> in <module>
     53
     54             # loss 계산
---> 55             output = alexnet(imgs)
     56             loss = F.cross_entropy(output, classes)
     57

                                 1 frames
/usr/local/lib/python3.8/dist-packages/torch/nn/modules/module.py in
_forward_unimplemented(self, *input)
    242         registered hooks while the latter silently ignores them.
    243     """
--> 244     raise NotImplementedError(f"Module [{type(self).__name__}] is missing the
required W"forwardW" function")
    245
```

Colab 유료 제품 - 여기에서 계약 취소

✓ 0초 오후 1:16에 완료됨