데이터 증강 라벨링 이미지 처리 데이터셋 구현 03

전진희

실습01. 데이터 셋 구현 기초

```
import torch
import os
import glob
from PIL import Image
from torch.utils.data import Dataset,DataLoader
```

```
class CustomImageDataset(Dataset):
    def __init__(self, path, mode='train'):
        # mode가 'train'일 경우에만 file_list를 가져오도록 설정
        if mode == 'train':
            file_list = os.listdir(path)  # 경로 내의 파일 목록
        self.a = [1, 2, 3, 4, 5]  # 임의의 리스트를 생성
        pass

    def __getitem__(self, index):

        return self.a[index]  # 인덱스에 해당하는 아이템을 반환

    def __len__(self):
        return len(self.a)  # 아이템의 개수



dataset_inst = CustomImageDataset('./images/', 'valid')  # 경로와 valid모드를 인자로 사용

for item in dataset_inst:
    print(item)  # 데이터셋 내의 각 아이템 프린트
```

Dunder (Double underscore) 메서드:

언더바 두 개로 시작하고 끝나는 파이선 내부 메서드
예를 들어 __init__, __getitem__, __len__ 등이 해당
이러한 메서드는 클래스의 특수한 동작(클래스)을 정의하거나, 특별한 기능을 제공하는 역할

실습 02. Pytorch 사용자 정의 데이터셋 구현 실습

```
class CustomImageDataset(Dataset):                    # Dataset 클래스 상속빋이 3가지 함수 생성
    def __init__(self, image_paths, transform=None):                # 1. 데이터 셋 초기화 및 설정
        # 이미지 파일 경로들을 저장하는 리스트를 초기화
        self.image_paths = glob.glob(os.path.join(image_paths, "*","*", "*.png")) # 패스, 파일이름
(뭐든 ok), 하위폴더까지 ok, 파일 포맷

        # 이미지 변환 함수를 저장하는 변수.
        self.transform = transform

        # 클래스 레이블과 숫자를 매핑한 딕셔너리
        self.label_dict = {
            "blue_cheese": 0,
             "camembert_cheese": 1,
             "cheddar_cheese": 2,
             "emmental_cheese": 3         }

    def __getitem__(self, index):                                    # 2. 데이터 샘플 반환
        # 주어진 인덱스에 해당하는 이미지 파일 경로를 가져옴
        image_path = self.image_paths[index]

        # 이미지 파일을 열어서 Image 객체로 변환
        image = Image.open(image_path)

        # 이미지 파일 경로에서 폴더 이름을 추출
        # 예시: "./sample_data_01/lightning/2100.jpg" -> "lightning"
        folder_name = image_path.split("₩₩")   # 경로는 역슬래쉬 두번
        folder_name = folder_name[2]

        # 추출한 폴더 이름을 이용하여 레이블을 가져옴
        label = self.label_dict[folder_name]
```

```
        # 이미지 변환 함수가 주어졌을 경우 변환을 적용
        if self.transform:                                  #  if self.transform is not None와 같은 말
            image = self.transform(image)

        return image, label

    def __len__(self):                                      # 3. 데이터 셋 크기 반환
        # 데이터셋의 총 데이터 개수를 반환
        return len(self.image_paths)
```

```
# 데이터 경로 ./sample_data_01/이미지 폴더/image.jpg

image_paths = "./cheese/"              # 데이터 폴더 경로 지정
dataset = CustomImageDataset(image_paths, transform=None)


# 디버깅

if __name__ == "__main__":
    image_paths = "./cheese/"
    dataset = CustomImageDataset(image_paths)                    # transform = None

    for item in dataset:
        print(f"Data and label: {item}")
```

```
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=287x176 at 0x2A3F3859370>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=272x186 at 0x2A3F3859970>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=274x184 at 0x2A3F3859130>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=299x168 at 0x2A3F3859370>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=275x183 at 0x2A3F38596D0>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=259x194 at 0x2A3F3859130>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=274x184 at 0x2A3F3859A30>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=300x168 at 0x2A3F38596D0>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=318x159 at 0x2A3F3859220>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=275x183 at 0x2A3F3859A30>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=259x194 at 0x2A3F3859460>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=275x183 at 0x2A3F3859220>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=300x168 at 0x2A3F3859970>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=225x225 at 0x2A3F3859460>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=239x210 at 0x2A3F3859370>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=259x194 at 0x2A3F3859970>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=183x275 at 0x2A3F3859130>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=225x225 at 0x2A3F3859370>, 0)
Data and label: (<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=275x183 at 0x2A3F38596D0>, 0)
```

실습03. Pytorch DataLoader

```
import torch
import os
import glob
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
```

```
def is_grayscale(img):
    return img.mode == 'L'



class CustomImageDataset(Dataset):
    def __init__(self, image_paths, transform = None):
        self.image_paths = glob.glob(os.path.join(image_paths, "*", "*", "*.png"))
        self.transform = transform
        self.label_dict = {
            "blue_cheese": 0,
            "camembert_cheese": 1,
            "cheddar_cheese": 2,
            "emmental_cheese": 30}



    def __getitem__(self, index):
        image_path: str = self.image_paths[index]
        image = Image.open(image_path).convert("RGB")



        if not is_grayscale(image):
            folder_name = image_path.split("₩₩")
            folder_name = folder_name[2]
```

```
            label = self.label_dict[folder_name]



            if self.transform:
                image = self.transform(image)



            return image, label



        else:
            print(f"{image_path} 파일은 흑백 이미지입니다.")



    def __len__(self):
        return len(self.image_paths)


# 데이터 경로

image_paths = "./cheese/" # 데이터 폴더 경로 지정
dataset = CustomImageDataset(image_paths, transform=None)
```

```
# 데이터 로더 정의


if __name__ == "__main__":
    transform = transforms.Compose([
```

```
        transforms.Resize((224, 224)),
        transforms.ToTensor()
    ])

    image_paths = './cheese/'
    dataset = CustomImageDataset(image_paths, transform=transform)

    data_loader = DataLoader(dataset, 32, shuffle=True)

    for images, labels in data_loader:
        print(f"Data and label : {images}, {labels}")
        print(f"Data and label : {images}, {labels}")
```

```
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000]],

   [[1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    ...,
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000]],

   [[1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    ...,
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000],
    [1.0000, 1.0000, 1.0000,  ..., 1.0000, 1.0000, 1.0000]]]), tensor([30,  0,  1,  2,  0, 30,  0,  2, 30, 30,  1])
```

실습04. DataLoader csv 데이터 이용하여 만들어 보기

```
import torch
import os
import glob
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
```

```
class HeightWeightDataset(Dataset):

    # 몸무게를 파운드 단위에서 kg로 바꾸는 함수

    def convert_to_kg(self, weight_lb):
```

```
        return weight_lb * 0.453592

    # 키를 인치에서 cm로 바꾸는 함수
    def inch_to_cm(self,inch):
        return inch * 2.54


    # 몸무게를 파운드 단위에서 kg로 바꾸는 함수

    def convert_to_kg(self, weight_lb):
        return weight_lb * 0.453592

    # 키를 인치에서 cm로 바꾸는 함수
    def inch_to_cm(self,inch):
        return inch * 2.54

    def __init__(self, csv_path):
        self.data = []
        with open(csv_path, 'r', encoding='utf-8') as f:
            next(f)
            for line in f:
                _, height, weight = line.strip().split(",")     # 인덱스 넘버 무시, 키와 몸무개를 strip은 공백을 삭제하고 콤마단위로 쪼개서 불러오기
                height = float(height)  # 문자열로 들어온 걸 숫자로 변경
                weight = float(weight)

                convert_to_kg_data = round(self.convert_to_kg(weight), 2) # 인치와 파운드로 단위가 설정된 것을 cm, kg으로 변환
                convert_to_cm_data = round(self.inch_to_cm(height), 1)

                self.data.append([convert_to_cm_data, convert_to_kg_data])

        pass


    def __getitem__(self, index):
        data = torch.tensor(self.data[index], dtype=torch.float)
        return data
```

```
    def __len__(self):
        return len(self.data)




if __name__ == "__main__":
    dataset = HeightWeightDataset("./hw_200.csv")
    dataloader = DataLoader(dataset, batch_size = 1, shuffle = True)

    for batch in dataloader:

        x = batch[:, 0].unsqueeze(1)
        y = batch[:, 1].unsqueeze(1) # unsqueeze는 괄호를 까준다

        print(x,y)
```

```
tensor([[177.6000]]) tensor([[66.6900]])
tensor([[169.]]) tensor([[49.1400]])
tensor([[163.3000]]) tensor([[46.6500]])
tensor([[180.3000]]) tensor([[72.1000]])
tensor([[164.8000]]) tensor([[46.3100]])
tensor([[173.6000]]) tensor([[62.8700]])
tensor([[177.9000]]) tensor([[70.4800]])
tensor([[180.6000]]) tensor([[58.1200]])
tensor([[168.6000]]) tensor([[54.1500]])
tensor([[178.]]) tensor([[59.6900]])
tensor([[177.3000]]) tensor([[64.1800]])
tensor([[169.1000]]) tensor([[54.8500]])
tensor([[167.9000]]) tensor([[52.5200]])
tensor([[178.5000]]) tensor([[56.9200]])
tensor([[175.8000]]) tensor([[58.5600]])
tensor([[176.2000]]) tensor([[63.0700]])
tensor([[175.5000]]) tensor([[52.4200]])
tensor([[178.3000]]) tensor([[65.7700]])
tensor([[178.3000]]) tensor([[54.9400]])
```

실습05. DataLoader json 데이터 이용하여 만들어 보기

```
import torch
import os
import glob
from PIL import Image
from torch.utils.data import Dataset, DataLoader
```

```
import json
```

```
class JsonCustomDataset(Dataset):
    def __init__(self, json_path, transform=None):
        self.transform = transform
        with open(json_path, 'r', encoding='utf-8') as f:
            self.data = json.load(f)  # json 내부에 있는 것 모두 불러옴

    def __getitem__(self, index):
        img_path = self.data[index]['filename']
        img_path = os.path.join("이미지 폴더", img_path)

        # image = Image.open(img_path) 이렇게 부르게 된다

        bboxes = self.data[index]['ann']['bboxes']
        labels = self.data[index]['ann']['labels']

        #if self.transform:
            # image = self.transform(image)

        return img_path, {'boxes': bboxes, 'labels':labels}

    def __len__(self):
        return len(self.data)


if __name__ == "__main__":
    dataset = JsonCustomDataset("./test.json", transform=None)

    for item in dataset:
        print(f"Data of dataset: {item}")
```

```
Data of dataset: ('이미지 폴더₩₩image_001.jpg', {'boxes': [[10, 10, 50, 50], [100, 100, 200, 200]], 'labels': [0, 1]})
Data of dataset: ('이미지 폴더₩₩image_002.jpg', {'boxes': [[20, 20, 60, 60], [300, 300, 400, 400]], 'labels': [1, 2]})
Data of dataset: ('이미지 폴더₩₩image_003.jpg', {'boxes': [[30, 30, 60, 60], [300, 300, 400, 400]], 'labels': [1, 2]})
Data of dataset: ('이미지 폴더₩₩image_004.jpg', {'boxes': [[10, 10, 60, 60], [300, 300, 400, 400]], 'labels': [1, 2]})
```

실습06. Pytorch DataLoader, Albumentations – transform구현

```
import cv2
import numpy as np
from torch.utils.data import Dataset
from torchvision import transforms



import albumentations as A
from albumentations.pytorch import ToTensorV2
```

```
class AlbumentationsDataset(Dataset):
    def __init__(self, file_paths, labels, transform=None):
        self.file_paths = file_paths
        self.labels = labels
        self.transform = transform

    def __getitem__(self, index):
        label = self.labels[index]
        file_path = self.file_paths[index]

        image = cv2.imread(file_path)

        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        if self.transform:
            augmented = self.transform(image=image)
            image = augmented['image']

        return image, label

    def __len__(self):
        return len(self.file_paths)

if __name__ == "__main__":
    albumentation_transform = A.Compose([
        A.Resize(256, 256),
        A.RandomCrop(224, 224),
        A.HorizontalFlip(),
        A.Normalize(
```

```
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    ),
    ToTensorV2()
])

file_paths = [
    "sample_data_01/train/dew/2208.jpg",
    "sample_data_01/train/fogsmog/4075.jpg",
    "sample_data_01/train/frost/3600.jpg"
]

labels = [0, 1, 2]

dataset = AlbumentationsDataset(file_paths, labels, transform=albumentation_transform)
```

```
for image, label in dataset:
    print(f"Data of dataset: {image},{label}")
```

```
          ...,
          [-0.4601, -0.4601, -0.4951,  ..., -1.4230, -1.4055, -1.3880],
          [-0.4776, -0.5126, -0.5126,  ..., -1.4405, -1.4405, -1.4230],
          [-0.4951, -0.5301, -0.5301,  ..., -1.4580, -1.4405, -1.4580]],

         [[-1.7870, -1.7870, -1.7870,  ..., -1.8044, -1.7522, -1.7696],
          [-1.7870, -1.7696, -1.7870,  ..., -1.8044, -1.7870, -1.7870],
          [-1.7522, -1.7870, -1.7696,  ..., -1.8044, -1.7870, -1.7870],
          ...,
          [-1.8044, -1.8044, -1.8044,  ..., -1.7870, -1.8044, -1.8044],
          [-1.8044, -1.7870, -1.8044,  ..., -1.7870, -1.8044, -1.7870],
          [-1.7870, -1.7870, -1.7870,  ..., -1.7870, -1.7696, -1.8044]]]),0
Data of dataset: tensor([[[1.8208, 1.8208, 1.8208,  ..., 1.4954, 1.4783, 1.4612],
          [1.8208, 1.8208, 1.8037,  ..., 1.5125, 1.4954, 1.4612],
          [1.8379, 1.8208, 1.8208,  ..., 1.5125, 1.4954, 1.4783],
          ...,
          [0.6049, 0.4166, 0.1083,  ..., 0.2967, 0.3994, 0.4679],
          [0.5878, 0.3138, 0.0741,  ..., 0.2796, 0.3481, 0.4508],
          [0.5536, 0.2282, 0.0398,  ..., 0.2624, 0.2967, 0.4337]],

         [[1.9909, 1.9909, 1.9909,  ..., 1.6583, 1.6408, 1.6232],
          [1.9909, 1.9909, 1.9734,  ..., 1.6758, 1.6583, 1.6232],
          [2.0084, 1.9909, 1.9909,  ..., 1.6758, 1.6583, 1.6408],
          ...,
          [0.7479, 0.5553, 0.2402,  ..., 0.4328, 0.5378, 0.6078],
          [0.7304, 0.4503, 0.2052,  ..., 0.4153, 0.4853, 0.5903],
          [0.6954, 0.3627, 0.1702,  ..., 0.3978, 0.4328, 0.5728]],

         [[2.2043, 2.2043, 2.2043,  ..., 1.8731, 1.8557, 1.8383],
          [2.2043, 2.2043, 2.1868,  ..., 1.8905, 1.8731, 1.8383],
          [2.2217, 2.2043, 2.2043,  ..., 1.8905, 1.8731, 1.8557],
          ...,
          [0.9668, 0.7751, 0.4614,  ..., 0.6531, 0.7576, 0.8274],
          [0.9494, 0.6705, 0.4265,  ..., 0.6356, 0.7054, 0.8099],
          [0.9145, 0.5834, 0.3916,  ..., 0.6182, 0.6531, 0.7925]]]),1
Data of dataset: tensor([[[-0.9020, -0.6965, -0.4568,  ..., -1.3473, -1.3815, -1.4158],
          [-0.9877, -0.7308, -0.5253,  ..., -1.3987, -1.4158, -1.4500],
          [-1.0390, -0.8164, -0.6109,  ..., -1.4329, -1.4500, -1.4843],
          ...,
          [-0.8849, -0.7308, -0.6794,  ...,  1.3070,  0.8618,  0.1083],
          [-0.8507, -0.7308, -0.7650,  ...,  1.7180,  1.3927,  1.0331],
          [-0.8507, -0.8164, -0.8849,  ...,  1.8550,  1.6667,  1.5125]],

         [[-0.6877, -0.4951, -0.3200,  ..., -1.2304, -1.2479, -1.3004],
```

실습 07. Imgaug 라이브러리 실습

Imgaug는 이미지 데이터 증강을 위한 파이썬 라이브러리

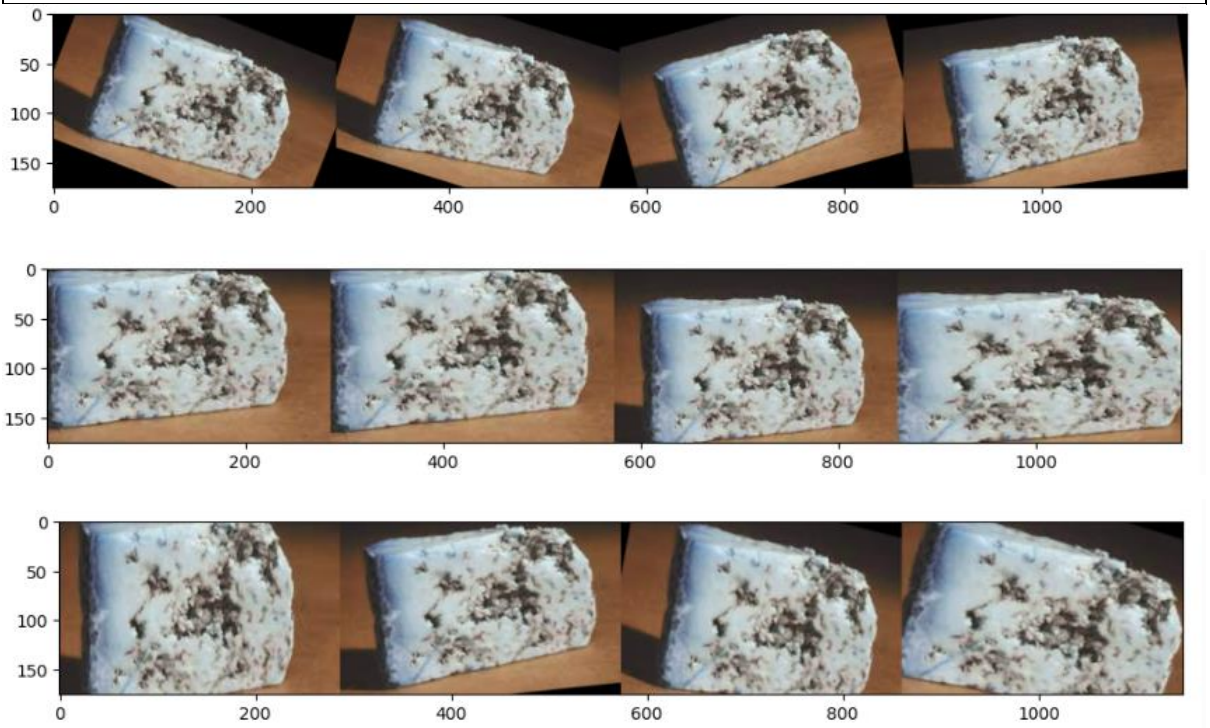
```
!pip install Imgaug  # 라이브러리 설치
```

```
import numpy as np
import imgaug.augmenters as iaa # ImaAug Augmenters를 줄여 iaa라고 부른다
import cv2
import matplotlib.pyplot as plt
```

```
# aumentation 확인
image = cv2.imread("cheese₩₩train₩₩blue_cheese₩₩0_blue_cheese.png")

images = [image, image, image, image]



# 1. rotate
rotate = iaa.Affine(rotate=(-25,25))
images_aug = rotate(images=images)

plt.figure(figsize=(12,12))
plt.imshow(np.hstack(images_aug))
plt.show()




# 2. crop

crop = iaa.Crop(percent=(0, 0.2))
image_aug01 = crop(images = images)

plt.figure(figsize=(12,12))
plt.imshow(np.hstack(image_aug01))
plt.show()
```

```
# 3. rotate crop
rotate_crop = iaa.Sequential([
    iaa.Affine(rotate=(-25, 25)),
    iaa.Crop(percent=(0, 0.2))

], random_order = True)
images_aug02 =rotate_crop(images=images)
plt.figure(figsize=(12,12))
plt.imshow(np.hstack(images_aug02))
plt.show()



# 4. iaa.OneOf()를 사용하면 여러 augmentation 기법들 중 하나를 선택
seq = iaa.OneOf([
    iaa.Grayscale(alpha=(0.0, 1.0)),
    iaa.AddToSaturation((-50, 50))
])
image_aug04 = seq(images=images)

#plt.figure(figsize=(12,12))
#plt.imshow(np.hstack(image_aug04))
#plt.show()                                  # numpy 버전 때문에 안됨


# 5. 이미지에 특정한 증강 작업을 적용하는 빈도나 확률을 제어
seq = iaa.Sequential([
    iaa.Sometimes(
        0.6,
        iaa.AddToSaturation((-50, 50))
    ),
    iaa.Sometimes(
        0.2,
        iaa.Grayscale(alpha=(0.0, 1.0))
    )
])
```

```
images_augmented = seq(images=images)


#plt.figure(figsize=(12, 12))
#plt.imshow(np.hstack(images_augmented))
#plt.show()                          # numpy 버전 때문에 안됨
```



실습08. pytorhch dataset dataload + imgaug 라이브러리 실습 및 시각화

```
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader
import imgaug as ia
import imgaug as ia
from imgaug import augmenters as iaa
import numpy as np
import matplotlib.pyplot as plt
```

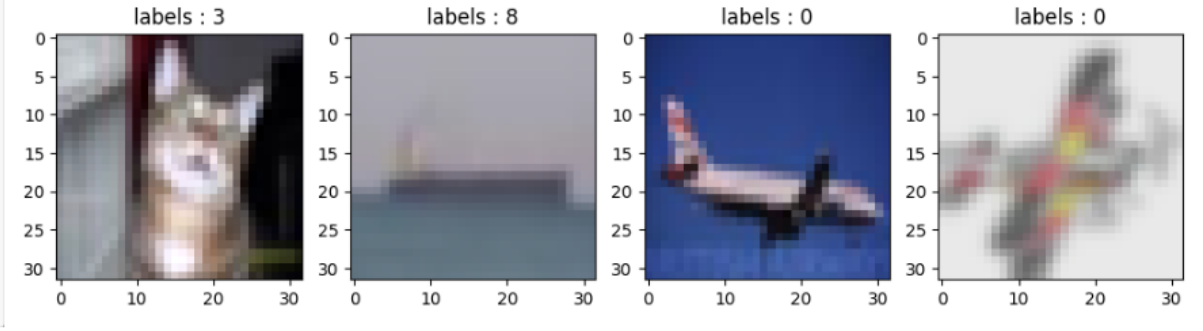
```
# 이미지 증강 함수


def imgaug_transform(image):
    seq = iaa.Sequential([
```

```
        iaa.Fliplr(0.5),
            iaa.GaussianBlur(sigma=(0, 1.0)),
            iaa.Multiply((0.8, 1.2)),
    ])
    image_np = image.permute(1, 2, 0).numpy()
    image_aug = seq(image=image_np)
    image_aug_copy = image_aug.copy()
    image_aug_copy = torch.from_numpy(image_aug_copy).permute(2, 0, 1)
    return image_aug_copy

# 입력 이미지를 파이토치 텐서로 변환 후 데이터 증강 적용하여 그 이미지 반환
def transform_data(image):
    tensor = transforms.ToTensor()(image)
    transformed_tensor = imgaug_transform(tensor)
    return transformed_tensor

# CIFAR10 dataset 로드
train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform_data)

# DataLoader 설정
batch_size = 4
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
```

```
# 증강된 이미지 시각화하여 확인
for images, labels in train_dataloader:
    fig, axes = plt.subplots(1, batch_size, figsize=(12,4))

    for i in range(batch_size):
        image = images[i].permute(1, 2, 0).numpy()
        axes[i].imshow(image)
        axes[i].set_title(f"labels : {labels[i]}")

    plt.show()
```

```
    break # 첫번째 미니배치만 확인
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



실습08. Pytorch 데이터셋 및 데이터 로드 속도 개선 [캐시 사용방법 실습]

캐시: 이전에 계산된 값을 임시로 저장하여 중복 작업을 피한다. 첫번 째 학습은 다 else로 가지만 그 이후에 반복되는 학습들은 캐시를 사용하여 진행

```
import torch
import os
import glob
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
```

```
def is_grayscale(img):
    return img.mode == 'L'

class CustomDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.images = glob.glob(os.path.join(root_dir, "*","*", "*.png"))  # root_dir 안에 있는 폴더와 "*.png" 형식의 파일들을 찾아서 리스트로 저장
        self.transform = transform  # 데이터 변환을 위한 transform 함수
        self.label_dict = {'cat':0, 'dog':1}  # 레이블과 해당하는 클래스 인덱스를 매핑하는 딕셔너리

        self.cache = {}  # 이미지 캐시용 딕셔너리

    def __getitem__(self, index):
        if index in self.cache:  # 이미지가 이미 캐시에 있는 경우
```

```
            image, label  = self.cache[index]  # 캐시에서 이미지와 레이블을 가져옴

        else:  # 이미지가 캐시에 없는 경우
            image_path = self.images[index]  # 해당 인덱스에 해당하는 이미지 경로
            image = Image.open(image_path)  # 이미지를 열기

            if not is_grayscale(image):  # 흑백 이미지가 아닌 경우
                folder_name = image_path.split('₩₩')  # 이미지 경로에서 폴더 이름 추출
                folder_name = folder_name[2]  # 폴더 이름에서 클래스명 추출
                label = self.label_dict[folder_name]  # 폴더 이름에 해당하는 레이블을 가져옴
                self.cache[index] = (image, label)  # 캐시에 이미지와 레이블 저장

            else:  # 흑백 이미지인 경우
                print(f"흑백 이미지 >> {image_path}")
                return None, None  # None 값을 반환하여 무시

        if self.transform:  # 데이터 변환 함수가 존재하는 경우
            image = self.transform(image)  # 이미지에 데이터 변환 함수를 적용

        return image, label  # 변환된 이미지와 레이블 반환

    def __len__(self):
        return len(self.images)  # 데이터셋의 총 이미지 개수 반환
```