🔥 Custom Dataset 실습 🔥

---

1) **CSV** 파일 기준
2) **JSON** 파일 기준
3) 이미지 폴더 기준

---

1) 이미지 폴더 기준

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

class CustomDataset(Dataset):
    def __init__(self, image_paths,
transform=None):
        self.image_paths =
glob.glob(os.path.join(image_paths, "*", "*",
"*.jpg"))
        # image_paths 내의 모든 하위 디렉토리에서
"*.jpg"로 끝나는 이미지 파일들의 경로를 찾습니다.
        print(self.image_paths)

        self.transform = transform

        self.label_dict = {
```

```
            "dew": 0, "fogsmog": 1, "frost":
2,
            "glaze": 3, "hail": 4,
"lightning": 5, "rain": 6, "rainbow": 7,
            "rime": 8, "sandstorm": 9, "snow":
10
        }

    def __getitem__(self, index):
        image_path = self.image_paths[index]

        # 이미지 열기
        image =
Image.open(image_path).convert("RGB")
        print(image_path)

        # 이미지 경로에서 폴더 이름 추출
        folder_name = image_path.split('/')
        folder_name = folder_name[6]  # "dew",
"fogsmog"와 같은 폴더 이름을 사용합니다.
        print("폴더 이름:", folder_name)

        # 레이블 딕셔너리에서 해당 폴더 이름에 대한
레이블 가져오기
        label = self.label_dict[folder_name]

        if self.transform:
            # 이미지 변환 함수가 있으면 이미지에
적용합니다.
            image = self.transform(image)
```

```
            return image, label

        def __len__(self):
            # 데이터셋의 총 데이터 개수 반환
            return len(self.image_paths)


if __name__ == '__main__':
    image_paths =
'./MS/CV/0619/sample_data_01/'
    dataset = CustomDataset(image_paths,
transform=None)

    dataloader = DataLoader(dataset,
batch_size=2, shuffle=True)

    for item in dataset:
        print(f"데이터와 레이블: {item}")
        break
```

---

2) CSV 파일 기준

```
from typing import Any
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, "*", "*",
"*.jpg"))
        self.transform = transform
        self.label_dict = {
            "dew": 0, "fogsmog": 1, "frost":
2, "glaze": 3, "hail": 4,
            "lightning": 5, "rain": 6,
"rainbow": 7, "rime": 8, "sandstorm": 9,
            "snow": 10
        }

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

        # 흑백 이미지가 아닌 경우에만 실행
        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)

if __name__ == "__main__":
    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor()
    ])

    image_paths = './sample_data_01/'
    dataset = CustomImageDataset(image_paths,
transform=transform)

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

```
    for images, labels in data_loader:
        print(f"데이터와 레이블: {images},
{labels}")
```

---

3) JSON 파일 기준

```
import json
import os
from typing import Any
from PIL import Image
import torch
from torch.utils.data import Dataset,
DataLoader

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)

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

```
            # image = Image.open(img_path)

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

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

            return image, {'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}")
```