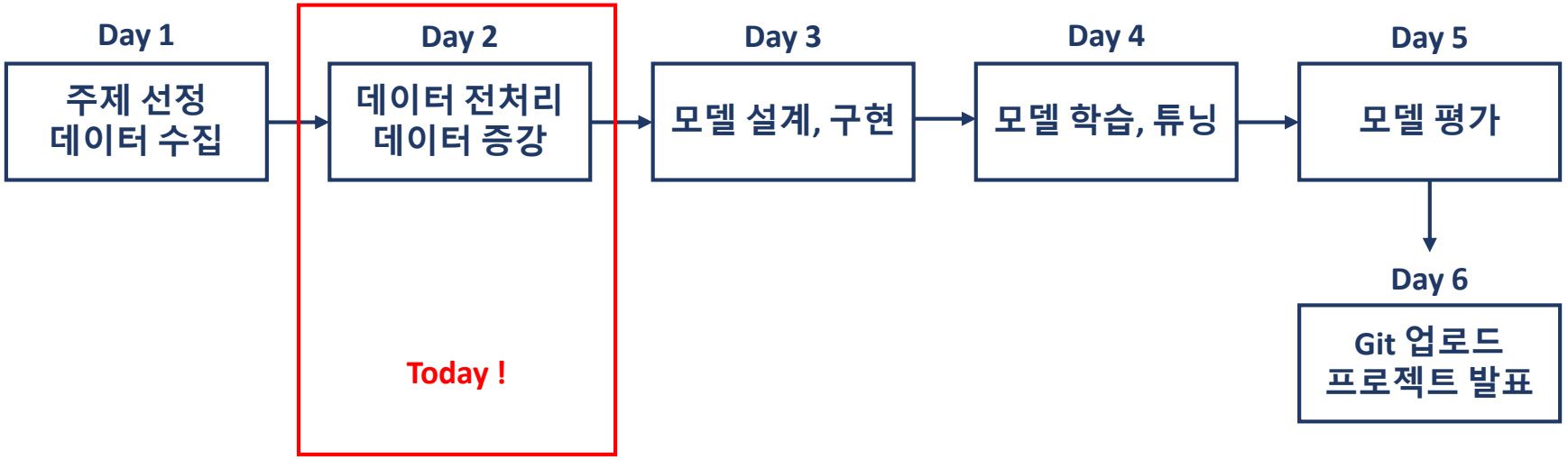
AI 기반 영상 데이터 분석 실습
- Day 2 -

index


Day 1
주제 선정
데이터 수집
Day 2
데이터 전처리
데이터 증강
Today !
Day 3
모델 설계, 구현
Day 4
모델 학습, 튜닝
Day 5
모델 평가
Day 6
Git 업로드
프로젝트 발표

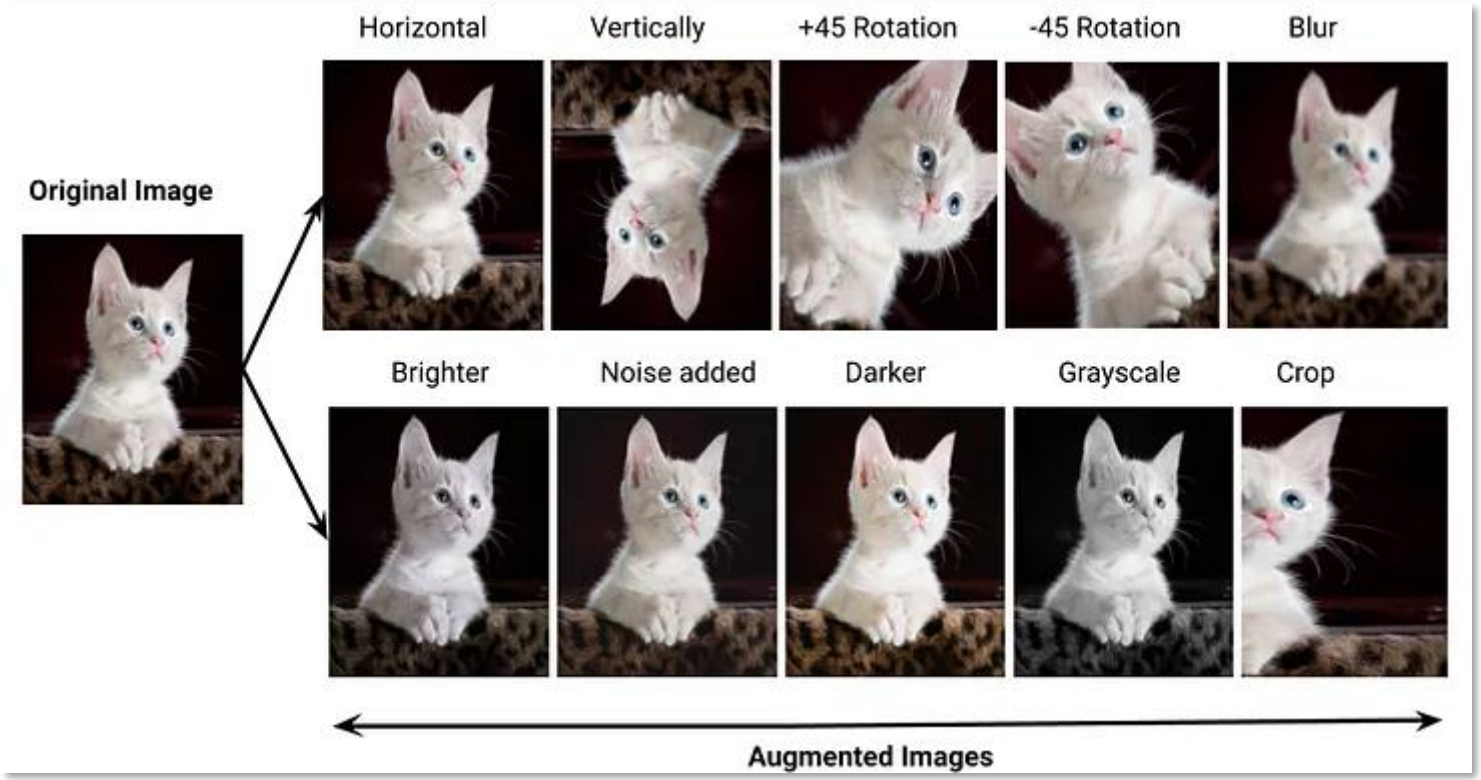
Afternoon Lab session (실습)

Day 2 : Course overview

- 전처리, 증강 구현
- 데이터 입력부(Dataset, DataLoader) 구현
- train, validation, test set split

Afternoon Lab session (실습)

Data Augmentation



Afternoon Lab session (실습)

Data Augmentation

- **Regularization(과적합 방지)**
  - 인공지능 모델이 똑같은 입력을 지속적으로 봄으로써 발생하는 특정 픽셀이나 패턴을 암기하는 현상(overfitting)을 방지하고 데이터의 본질적인 특징에 집중하도록 강제

- **Invariance(불변성)**
  - 위치가 바뀌거나 회전하거나 밝아지거나 어두워져도 분류해야할 target object와 라벨은 변하지 않음을 학습

- **Robustness(강건함)**
  - target object의 경계를 부드럽게 만들어 unseen data에 강건하도록 만듬

Afternoon Lab session (실습)

실습

- **Data Augmentation**
  - torchvision.transforms 사용

- **Custom Dataset Class**
  - 데이터를 어떤 형태로 받아올 지 정의
  - torch.utils.data.Dataset

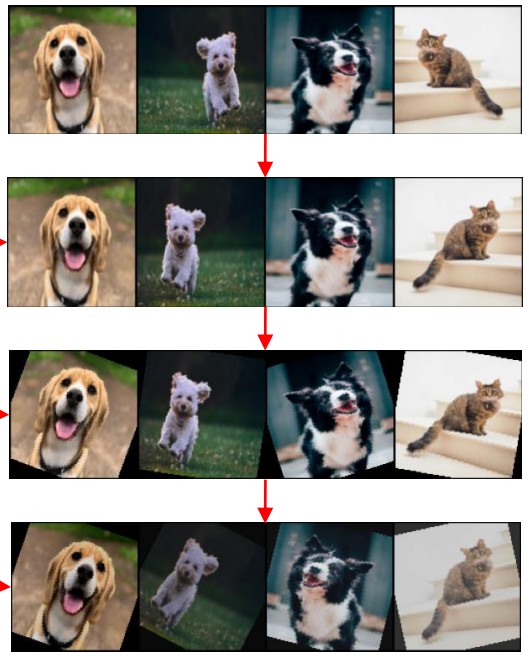
- **DataLoader**
  - 데이터를 받아오는 Loader의 역할
  - Batch 단위로 모아서 iteration의 형태로 구성, 데이터 Shuffle
  - torch.utils.data.DataLoader

Afternoon Lab session (실습)

실습

- **Data Augmentation**
  - torchvision.transforms 사용

```
my_transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomRotation(degrees=30),
    transforms.ColorJitter(brightness=0.2, contrast=0.3),
    transforms.ToTensor(),
])
```



Afternoon Lab session (실습)

실습

- **Custom Dataset Class**
  - 데이터를 어떤 형태로 받아올 지 정의
  - torch.utils.data.Dataset

```
from torch.utils.data import Dataset
```

```
data_path = '/content/train' # 이미지가 담겨있는 폴더
labels = ['dog', 'cat']
```

```
class CustomImageDataset(Dataset):
    def __init__(self, img_dir, labels, transform=None):
        self.img_dir = img_dir
        self.labels = labels
        self.transform = transform
        self.img_Paths_Labels = [] # (image 경로, 라벨)로 구성된 배열
        for i, lbl in enumerate(self.labels):
          img_path = os.path.join(img_dir, lbl)
          for img_file in os.listdir(img_path):
            self.img_Paths_Labels.append([os.path.join(img_path, img_file), i])

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

    def __getitem__(self, idx):
        img_path, label = self.img_Paths_Labels[idx]
        image = read_image(img_path)
        if self.transform:
            image = self.transform(image)
        return image, label
```

Afternoon Lab session (실습)

실습

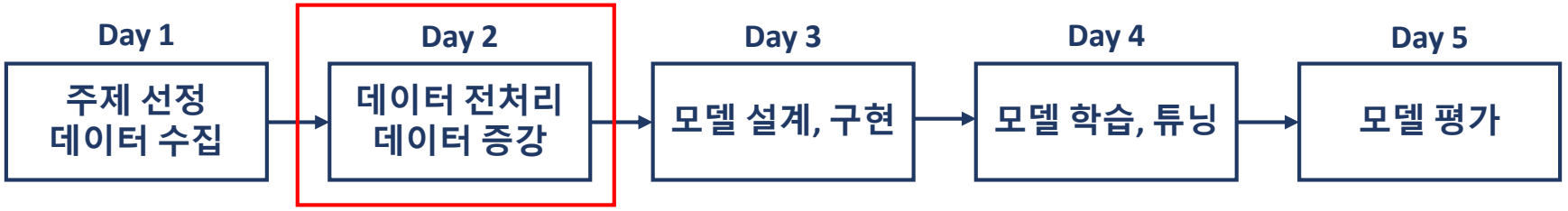
- **DataLoader**
  - 데이터를 받아오는 Loader의 역할
  - Batch 단위로 모아서 iteration의 형태로 구성, 데이터 Shuffle
  - torch.utils.data.DataLoader

```
from torch.utils.data import DataLoader
```

```
batch_size = 4 # 한 번에 받을 데이터 수

# shuffle=True 내부 데이터 섞기
train_loader = DataLoader(my_dataset, batch_size=batch_size, shuffle=False)
```

01/20 To-do



- **visualization(시각화)**
  - 코드 제공

```
def show_images(img_tensor):
    plt.figure(figsize=(8, 8))
    img_grid = torchvision.utils.make_grid(img_tensor) # 여러 장을 한 장으로 합치기
    npimg = img_grid.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.axis('off')
    plt.show()

# 로더에서 받은 첫 번째 묶음(배치) 시각화
dataiter = iter(train_loader)
images, labels = next(dataiter)

# 그림 그리기 함수 호출
show_images(images)
```

```
총 6장의 데이터셋이 선언되었습니다.
DataLoader(batch_size : 4) 선언 완료
```

Thank you