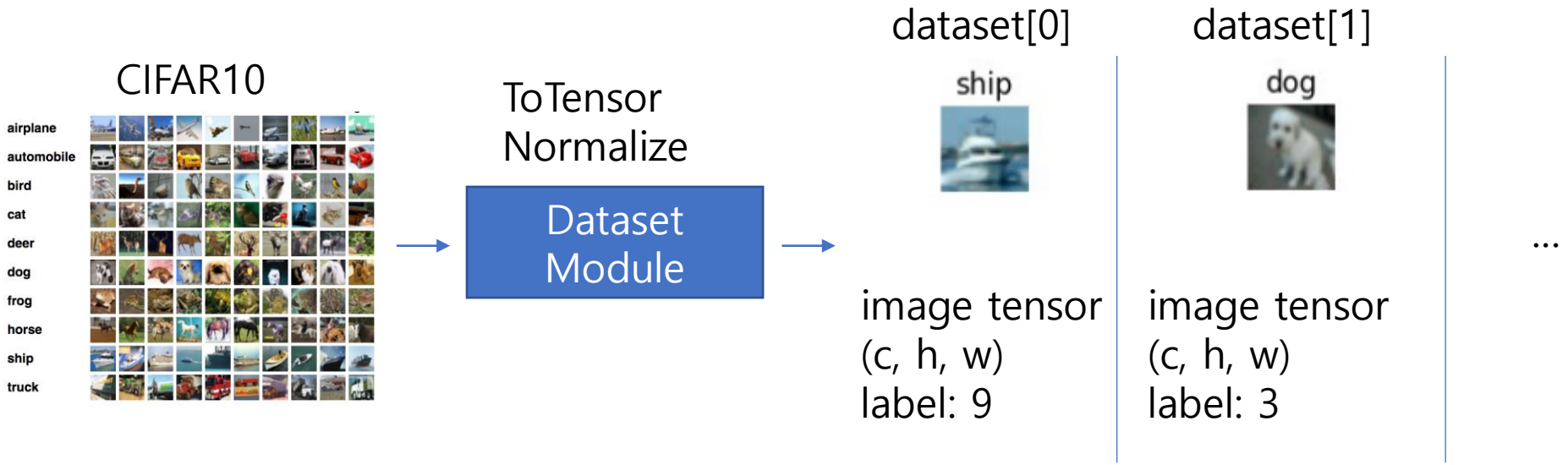
Code Review
CNN

김균엽

Day 3(CIFAR10 with LeNet)

- dataset load
  - CIFAR dataset load
  - transform ToTensor and normalize
  - dataset module을 통해 하나의 이미지씩 반환

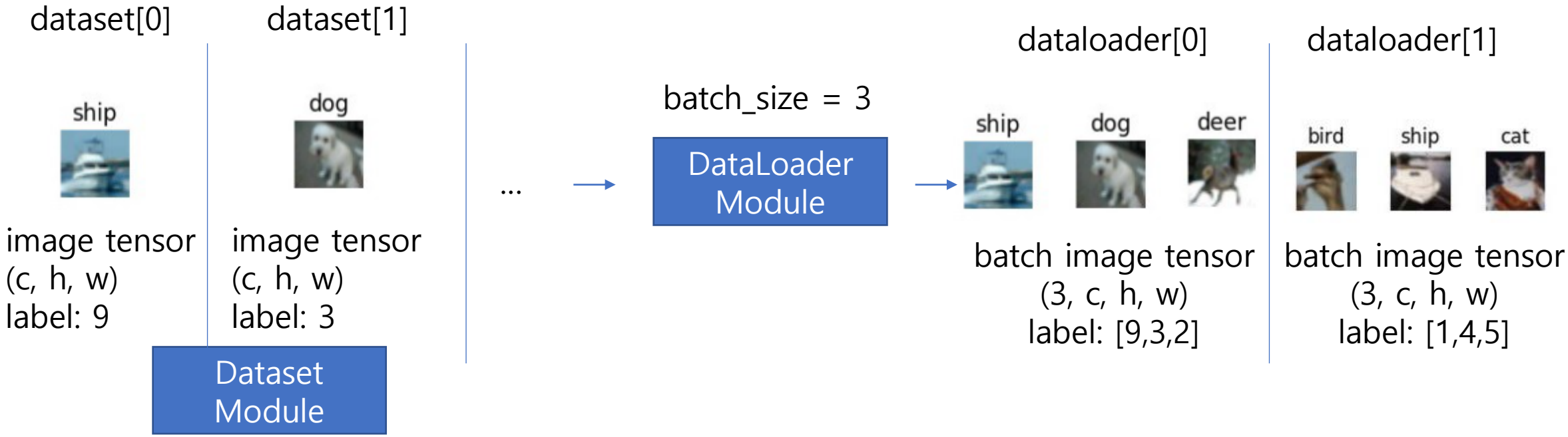
```
#load할 dataset에 사전작업될 이미지 전처리
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize(0.5,0.5)])

#torchvision을 통한 dataset load
trainDataset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                            download=True, transform=transform
testDataset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                           download=True, transform=transform)
```



Day 3(CIFAR10 with LeNet)

- dataloader build
  for batch processing
  - CIFAR dataset load
  - transform ToTensor and normalize
  - dataset module을 통해 하나의 이미지씩 반환

```
#batch processing을 위한 dataloader 구성
batch_size = 20
trainDataloader = torch.utils.data.DataLoader(trainDataset,
batch_size=batch_size, shuffle=True)
testDataloader = torch.utils.data.DataLoader(testDataset,
batch_size=batch_size,shuffle=False)
```



Day 3(CIFAR10 with LeNet)

- CNN model structure
  - CNN모듈을 통하여 이미지의 공간정보 학습
  - 결과로 120차원의 output이 나오도록 구현

```
class myModel(nn.Module):
    def __init__(self):
        super(myModel, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.AvgPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.conv3 = nn.Conv2d(16, 120, 5)

def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = self.relu(self.conv3(x))
```

dataloader[1]

batch image tensor
(3, c, h, w)#3,32,32
label: [1,4,5]

6,28,28

16,10,10

120,1,1

Day 3(CIFAR10 with LeNet)

- Linear model structure
  - classificatio을 위해 결과가 data의 class수와 같아야한다.
  - CIFAR은 10개의 class를 가지고 있기 때문에 10차원의 output이 나와야 한다.
  - airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck

```
class myModel(nn.Module):
    def __init__(self):
        super(myModel, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.AvgPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(100, 10)

def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = self.relu(self.conv3(x))
        x = x.reshape(-1, 120)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
```

airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck

[p1, p2, ,p3, p4, p5, p6, p7, p8, p9, p10]

[p1, p2, ,p3, p4, p5, p6, p7, p8, p9, p10]

[p1, p2, ,p3, p4, p5, p6, p7, p8, p9, p10]

Day 3(CIFAR10 with LeNet)

- train process
  - model output 연산 `output = model(data)`
  - loss 연산 `loss = lf(output, target)`
  - loss.backward를 통해 gradient 계산 `loss.backward()`
  - optimizer.step을 통해 parameter update `optimizer.step()`

  - train dataset을 통해 학습하여 parameter를 조정함
  - 조정된 parameter가 좋은 결과와 성능을 얻어내는지를 확인하기 위해 test dataset을 통하여 검증함