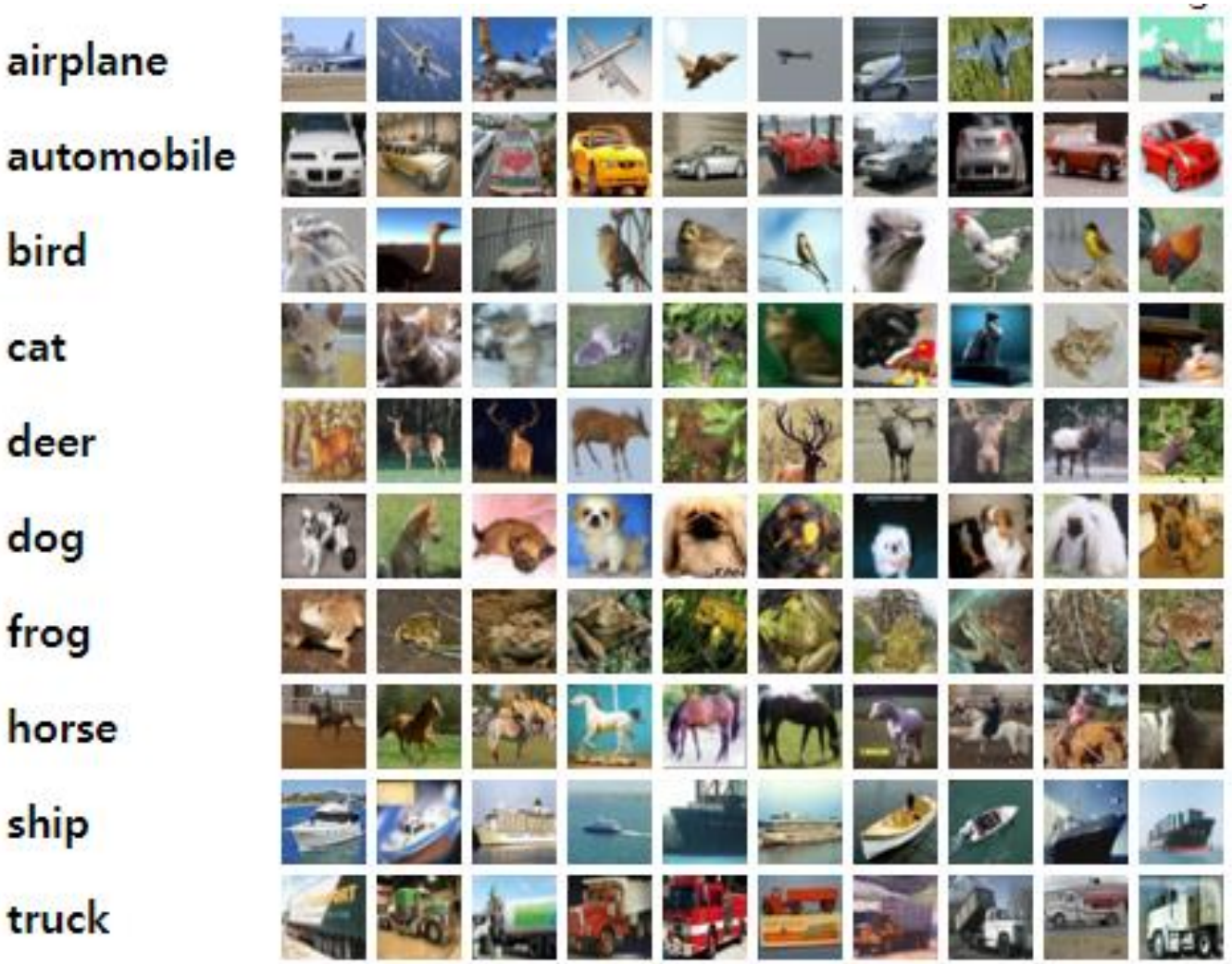
이미지 분류모델 구현 – CNN

박경규

CIFAR-10 dataset

- 32x32픽셀의 60,000개 컬러이미지가 포함되어있으며, 각 이미지는 10개의 클래스로 라벨링이 되어있습니다



CIFAR-10 분류모델 - MLP

https://github.com/Justin-A/DeepLearning101/blob/master/4-1_CIFAR_MLP.ipynb

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(32 * 32 * 3, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 10)

    def forward(self, x):
        x = x.view(-1, 32 * 32 * 3)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        x = F.relu(x)
        x = self.fc3(x)
        x = F.log_softmax(x, dim = 1)
        return x
```

CIFAR-10 분류모델 – MLP

```
Train Epoch: 10 [0/50000 (0%)]  Train Loss: 1.443836
Train Epoch: 10 [6400/50000 (13%)]      Train Loss: 1.206149
Train Epoch: 10 [12800/50000 (26%)]     Train Loss: 1.214895
Train Epoch: 10 [19200/50000 (38%)]     Train Loss: 1.214073
Train Epoch: 10 [25600/50000 (51%)]     Train Loss: 1.284093
Train Epoch: 10 [32000/50000 (64%)]     Train Loss: 1.527012
Train Epoch: 10 [38400/50000 (77%)]     Train Loss: 1.170188
Train Epoch: 10 [44800/50000 (90%)]     Train Loss: 1.124379

[EPOCH: 10],    Test Loss: 1.4839,      Test Accuracy: 48.03 %
```

CIFAR-10 분류모델 - CNN

https://github.com/Justin-A/DeepLearning101/blob/master/4-2_CIFAR_CNN.ipynb

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        self.conv1 = nn.Conv2d(
                        in_channels = 3,
                        out_channels = 8,
                        kernel_size = 3,
                        padding = 1)
        self.conv2 = nn.Conv2d(
                        in_channels = 8,
                        out_channels = 16,
                        kernel_size = 3,
                        padding = 1)
        self.pool = nn.MaxPool2d(kernel_size = 2, stride = 2)

        self.fc1 = nn.Linear(8 * 8 * 16, 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, 10)
```

CIFAR-10 분류모델 - CNN

https://github.com/Justin-A/DeepLearning101/blob/master/4-2_CIFAR_CNN.ipynb

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

    x = x.view(-1, 8 * 8 * 16)
    x = self.fc1(x)
    x = F.relu(x)
    x = self.fc2(x)
    x = F.relu(x)
    x = self.fc3(x)
    x = F.log_softmax(x)
    return x
```

CIFAR-10 분류모델 – CNN

```
Train Epoch: 10 [0/50000 (0%)]  Train Loss: 1.082029
Train Epoch: 10 [6400/50000 (13%)]      Train Loss: 0.821738
Train Epoch: 10 [12800/50000 (26%)]     Train Loss: 0.997346
Train Epoch: 10 [19200/50000 (38%)]     Train Loss: 0.981015
Train Epoch: 10 [25600/50000 (51%)]     Train Loss: 0.686181
Train Epoch: 10 [32000/50000 (64%)]     Train Loss: 1.017531
Train Epoch: 10 [38400/50000 (77%)]     Train Loss: 0.632845
Train Epoch: 10 [44800/50000 (90%)]     Train Loss: 0.945488

[EPOCH: 10],    Test Loss: 1.0441,      Test Accuracy: 62.88 %
```

Image Augmentation

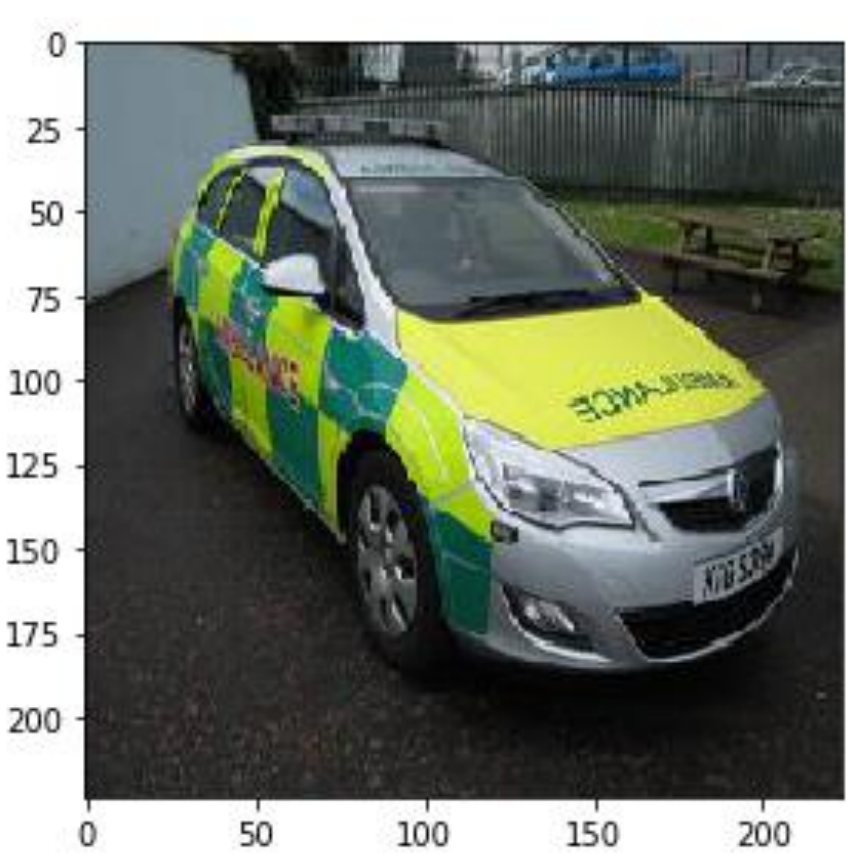
- Image Augmentation은 딥 러닝 모델을 훈련하기 위해 새로운 이미지를 생성하는 프로세스입니다.
- 이러한 새 이미지는 기존 학습 이미지를 사용하여 생성되므로 수동으로 수집 할 필요가 없습니다.

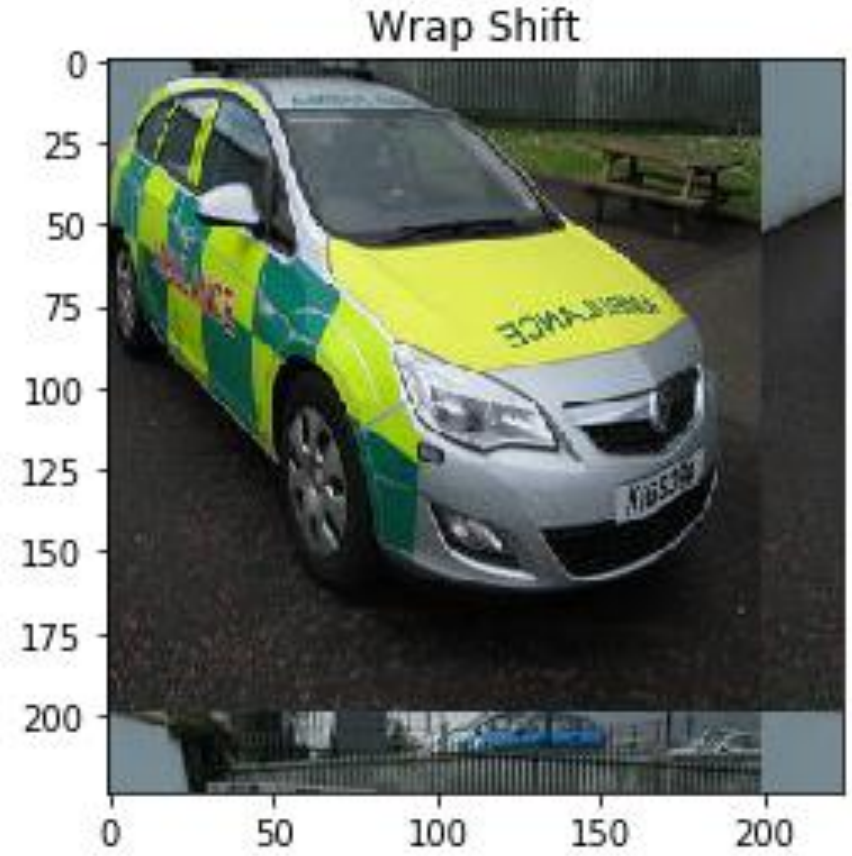
https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Image Augmentation – Rotation

https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

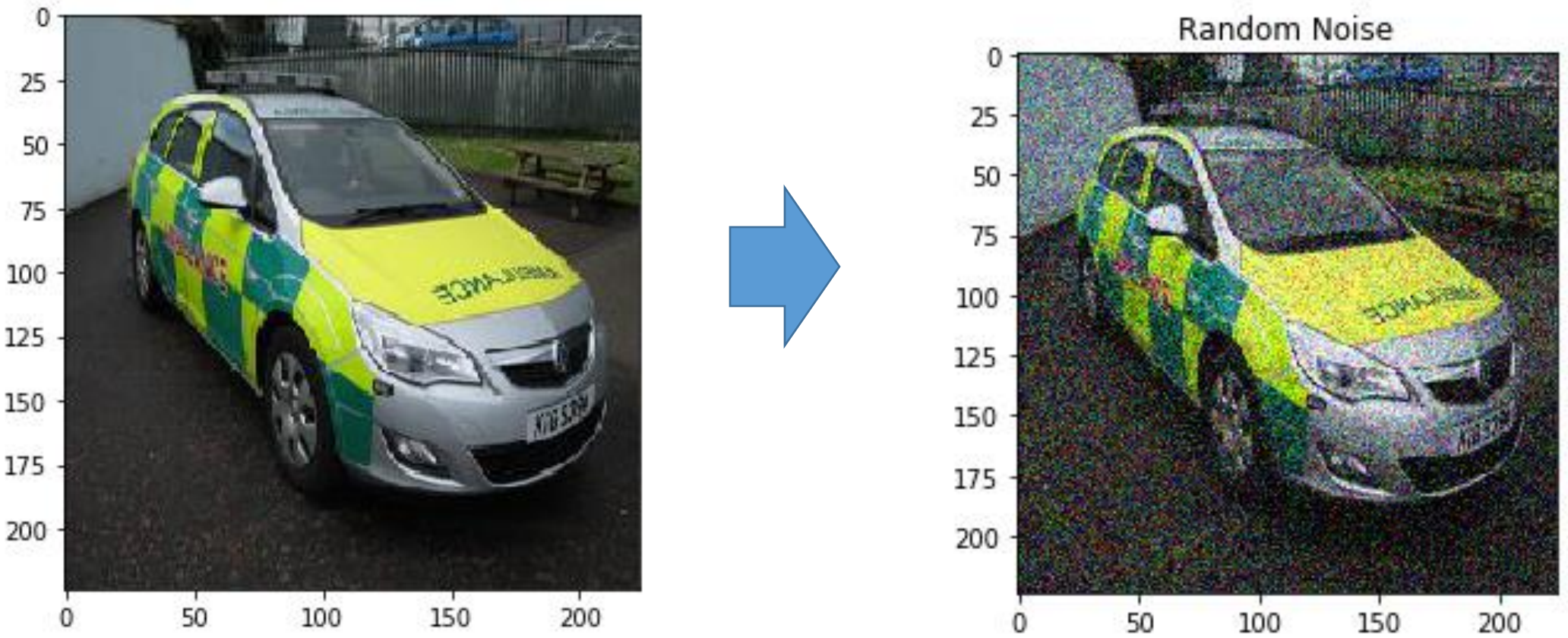
Image Augmentation – Shifting





https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Image Augmentation – Adding Noise

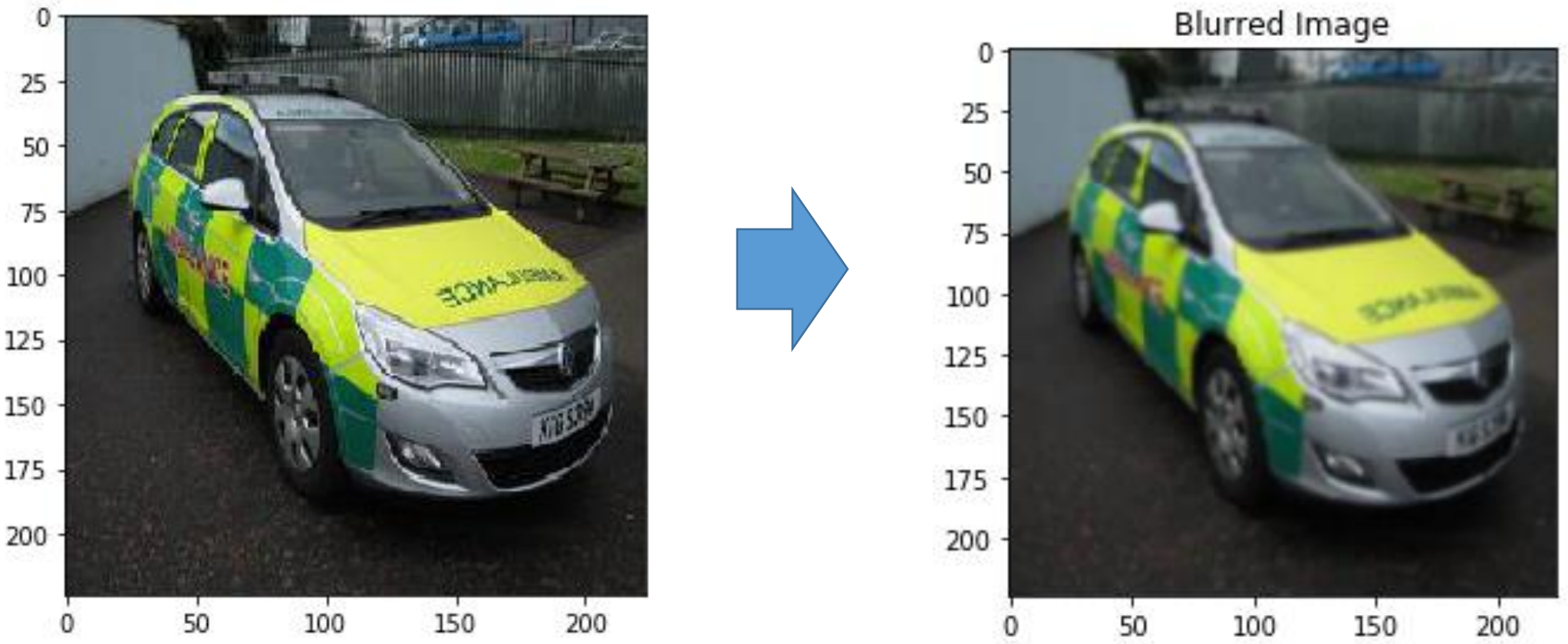


https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Image Augmentation – Shifting

https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Image Augmentation – Blurring



https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

CIFAR-10 분류모델 - Augmentation

https://github.com/Justin-A/DeepLearning101/blob/master/4-3_CIFAR_CNN_Augmentation.ipynb

```
from torchvision import transforms

train_dataset = datasets.CIFAR10(root = "../data/CIFAR_10",
                                 train = True,
                                 download = True,
                                 transform = transforms.Compose([
                                   transforms.RandomHorizontalFlip(),
                                   transforms.ToTensor(),
                                   transforms.Normalize(
                                  (0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]))
```

CIFAR-10 분류모델 - Augmentation

https://pytorch.org/vision/stable/transforms.html

- torchvision.transforms.CenterCrop(*size*)
- torchvision.transforms.ColorJitter(*brightness=0, contrast=0, saturation=0, hue=0*)
- torchvision.transforms.RandomCrop(*size, padding=None, pad_if_needed=False, fill=0, padding_mode='constant'*)
- torchvision.transforms.RandomGrayscale(*p=0.1*)
- torchvision.transforms.RandomHorizontalFlip(*p=0.5*)
- torchvision.transforms.RandomPerspective(*distortion_scale=0.5, p=0.5, interpolation=<InterpolationMode.BILINEAR: 'bilinear'>, fill=0*)
- torchvision.transforms.RandomRotation(*degrees, interpolation=<InterpolationMode.NEAREST: 'nearest'>, expand=False, center=None, fill=0, resample=None*)
- torchvision.transforms.RandomVerticalFlip(*p=0.5*)
- torchvision.transforms.GaussianBlur(*kernel_size, sigma=(0.1, 2.0)*)

CIFAR-10 분류모델 - Augmentation

```
Train Epoch: 10 [0/50000 (0%)]  Train Loss: 0.907567
Train Epoch: 10 [6400/50000 (13%)]      Train Loss: 0.965306
Train Epoch: 10 [12800/50000 (26%)]     Train Loss: 1.070564
Train Epoch: 10 [19200/50000 (38%)]     Train Loss: 1.250733
Train Epoch: 10 [25600/50000 (51%)]     Train Loss: 0.730824
Train Epoch: 10 [32000/50000 (64%)]     Train Loss: 0.759369
Train Epoch: 10 [38400/50000 (77%)]     Train Loss: 1.060894
Train Epoch: 10 [44800/50000 (90%)]     Train Loss: 0.703006

[EPOCH: 10],    Test Loss: 0.0306,      Test Accuracy: 65.85 %
```

Thank you

kgpark88@gmail.com