

# 데이터 없이 학습하는 GAN

---

## Contents

1. GAN
2. Knowledge Distillation
3. 데이터 없이 학습하는 GAN
4. Back-up

CDO AI 빅데이터담당

오퍼레이션DX기술팀

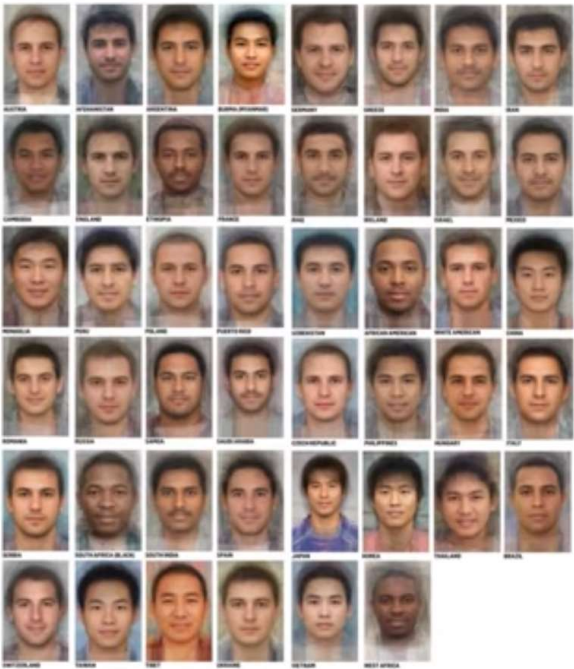
최진구 책임연구원

# Background & History (GAN)

## 이미지 데이터에 대한 확률분포

- 이미지 데이터는 다차원 특징 공간의 한 점으로 표현됩니다.
  - 이미지의 분포를 근사하는 모델을 학습할 수 있습니다.
- 사람의 얼굴에는 통계적인 평균치가 존재할 수 있습니다.
  - 모델은 이를 수치적으로 표현할 수 있게 됩니다.

코의 길이, 눈썹 길이, 입술, ...  
확률 분포를 학습



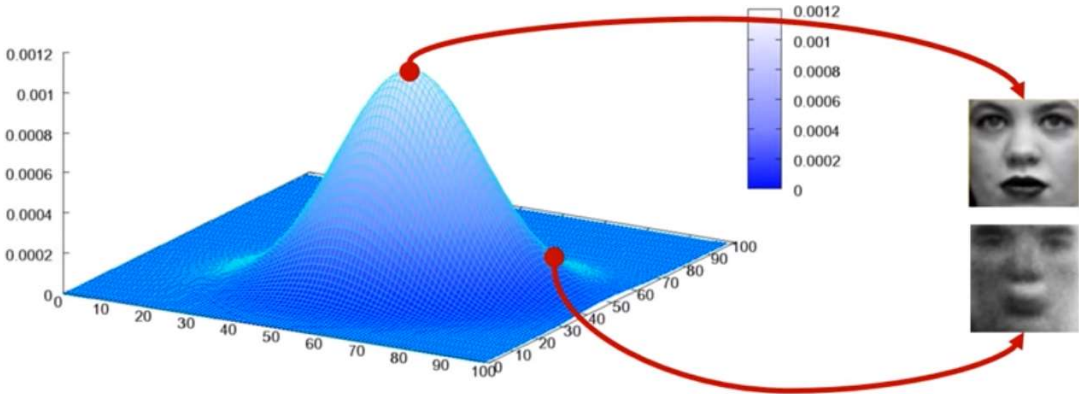
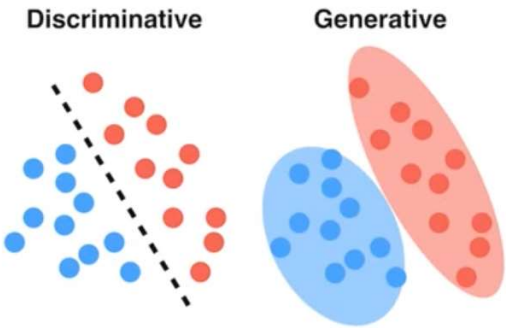
# Background & History (GAN)

## 생성 모델 (Generative Models)

- 생성 모델은 실존하지 않지만 있을 법한 이미지를 생성할 수 있는 모델을 의미합니다.

**Generative Model** (produce) → An image that does not exist but is likely to exist

- A statistical model of the joint probability distribution
- An architecture to generate new data instances



높은 확률 값에 해당하는  
이미지 생성



확률 분포를 학습 → 확률 값이 높은 = 있을 법한 이미지를 만들어 냄

# Background & History (GAN)

## Generative Adversarial Networks (GAN)

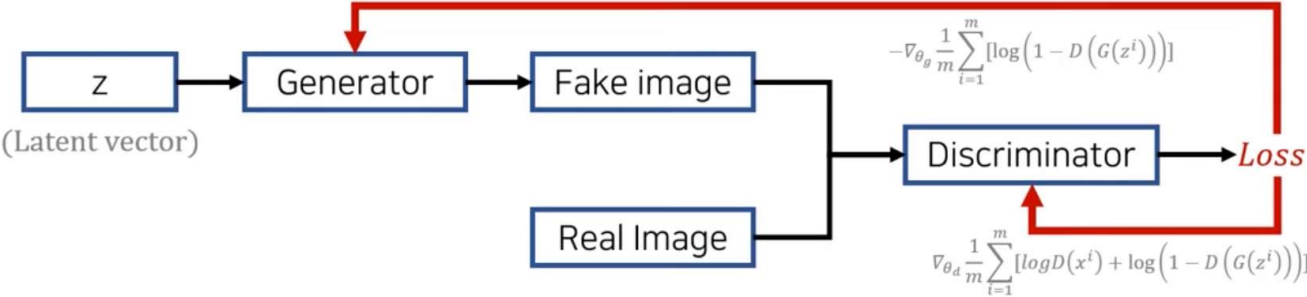
- 생성자(generator)와 판별자(discriminator) 두 개의 네트워크를 활용한 생성 모델입니다.
- 다음의 목적 함수(objective function)를 통해 생성자는 이미지 분포를 학습할 수 있습니다. 노이즈 분포에서 샘플  $z$ 를 뽑아 개댓값 계산

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Generator  $G(z)$ : new data instance

Discriminator  $D(x)$  = Probability: a sample came from the real distribution (Real: 1 ~ Fake: 0)

원본 데이터 분포에서 샘플  $x$ 를 뽑아  $\log D(x)$  기대값 계산



## Background & History (Knowledge Distillation)

가중치가

Knowledge Distillation을 제안합니다. Knowledge Distillation은 teacher model이 갖고 있는 지식을 더 작은 모델인 student model에 transfer 하는 것

실제 모델을 배포할 때, 더 작은 모델을 사용하여 예측 속도도 높이고, 정확도도 높일 수 있습니다. 이 외에도 knowledge distillation은 model을 generalization 하는 효과도 있어, test error를 낮출 수 있습니다.

MNIST dataset에서 숫자 3 데이터를 제거하여 student model을 knowledge distillation 방법으로 학습시킵니다. 숫자 3에 대한 정보를 학습하지 않았지만, soft label이 갖고 있는 정보로만 학습하여 test 3 이미지에 대해 98.6%의 정확도를 달성

교사 모델이 정답이 아님  
교사 모델이 항상 옳은 것이 아님

사전에 학습된 teacher model로부터 soft label을 출력합니다. soft label은 정답일 확률이 [0.1, 0.2, 0.3, 0.05] 처럼 극단적인 값을 갖지 않습니다. 정답 이외의 확률이 존재하여 해당 이미지에서 더 많은 정보를 추출합니다. 더 많은 정보를 갖고 있는 soft label을 사용하여 student model을 학습합니다.

반면에, hard label은 [0,0,1,0] 처럼 정답일 확률이 극단적인 값을 갖는 label 입

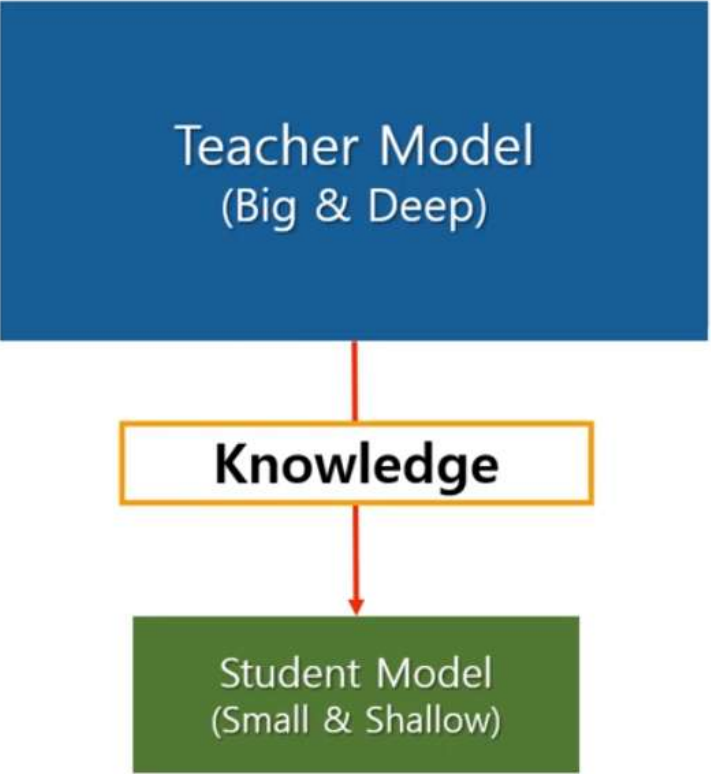
# Background & History (Knowledge Distillation)

Model Pruning 방법은 모델을 가볍게 만들 수 있지만  
**Model family** 변경이 어려움



Teacher, Student Model

# Knowledge Distillation



**Teacher 모델:** 높은 예측 정확도를 가진 복잡한 모델

e.g. 정확도 : 95 %  
추론 시간 : 2시간

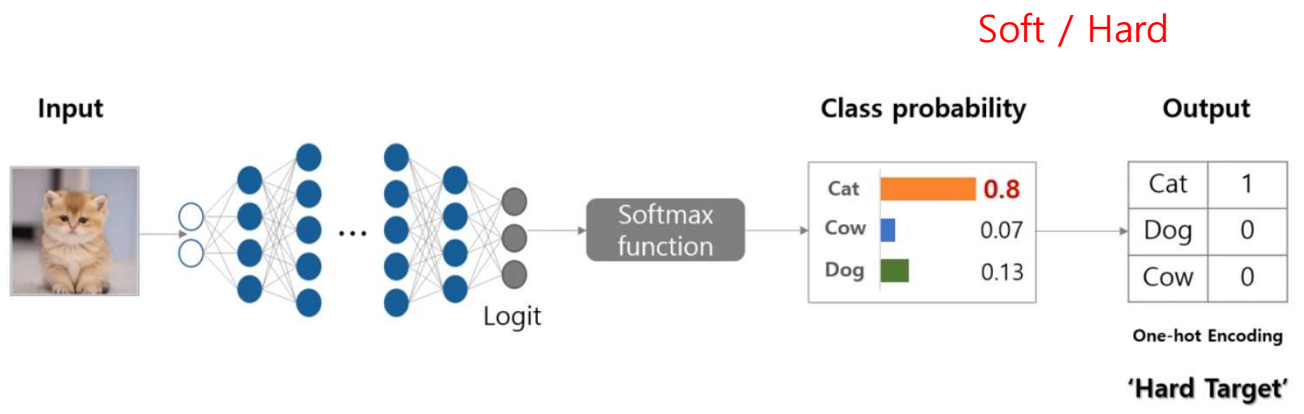
☒ 확률 분포를 학습 → 확률 값이 높은 = 있을 법한 이미지를 만들어 냄

잘 학습된 Teacher 모델의 **지식**을 전달하여  
단순한 Student 모델로 비슷한 좋은 성능을 내도록 함

**Student 모델:** Teacher 모델의 지식을 받는 단순한 모델

e.g. 정확도 : 90 %  
추론 시간 : 5분

# Knowledge Distillation

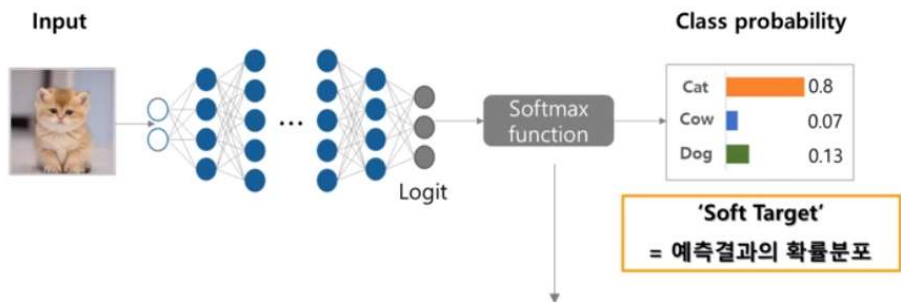




# Knowledge Distillation

Knowledge : Soft Target 사용

여전히 Soft함이 덜함, 정답인 Cat Class는 값이 넘 크고  
나머지 Cow, Dog 너무 작게 극단적인 분포



더 Soft 하게

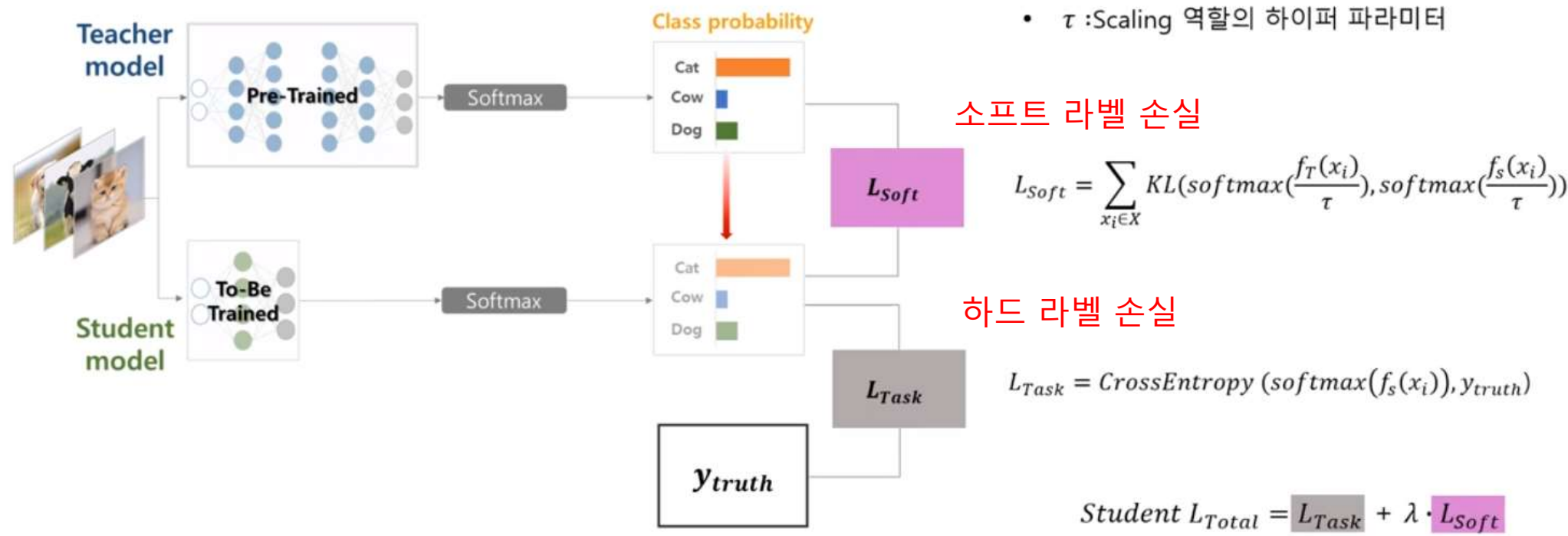
$$Softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \longrightarrow Softmax(z_i) = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}$$

$\tau$  (Temperature): Scaling 역할의 하이퍼 파라미터

- $\tau = 1$  일 때, 기존 softmax function과 동일
- $\tau$  클수록, 더 soft한 확률분포


# Knowledge Distillation

## Distillation 방법 : Offline - distillation



# Knowledge Distillation

## 딥러닝 라이브러리 Keras에서 함수 제공

 Keras

About Keras

Getting started

Developer guides

Keras API reference

Code examples

Why choose Keras?



Community & governance

Search Keras documentation...

» Code examples / Computer Vision / Knowledge Distillation

### Knowledge Distillation

**Author:** Kenneth Borup  
**Date created:** 2020/09/01  
**Last modified:** 2020/09/01  
**Description:** Implementation of classical Knowledge Distillation.

 [View in Colab](#) •  [GitHub source](#)

[https://keras.io/examples/vision/knowledge\\_distillation/](https://keras.io/examples/vision/knowledge_distillation/)

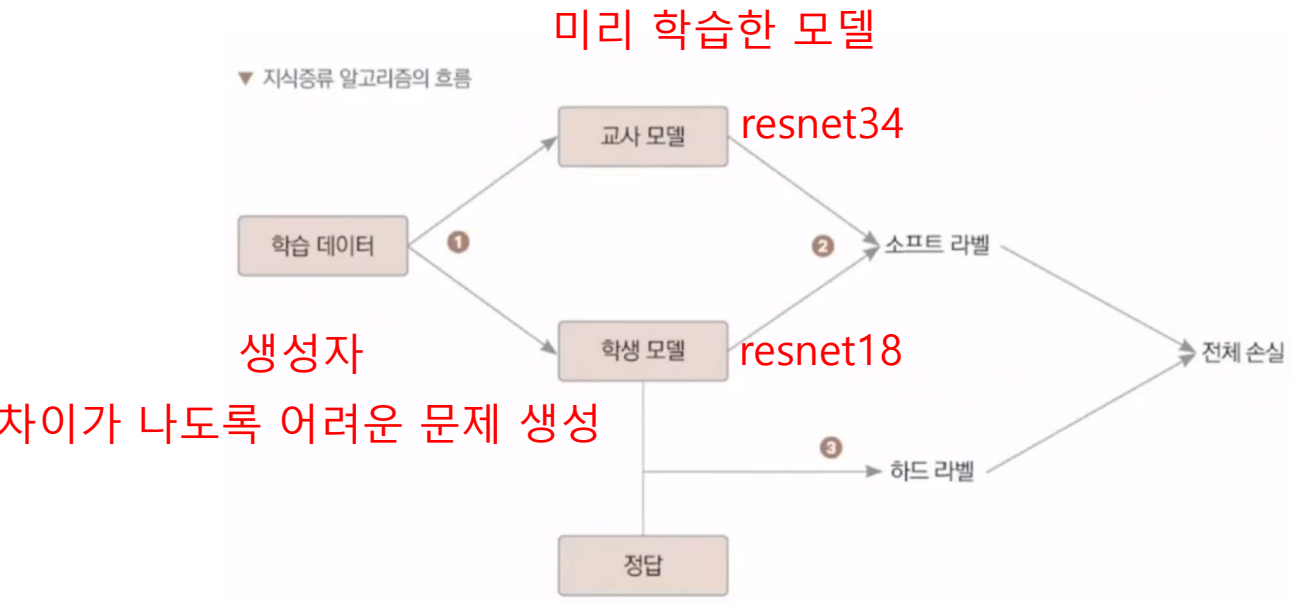
# **Knowledge Distillation: A good teacher is patient and consistent**

Beyer et al. (2021.6 preprint) 

# **Does Knowledge Distillation Really Work?**

Stanton et al. (2021.6 preprint)

# 데이터 없이 학습하는 GAN



▼ 데이터 전처리 정의

```
import tqdm
import torch
import torch.nn as nn

from torchvision.datasets.cifar import CIFAR10
from torchvision.transforms import Compose, ToTensor
from torchvision.transforms import RandomHorizontalFlip, RandomCrop
from torchvision.transforms import Normalize
from torch.utils.data.dataloader import DataLoader
from torchvision.models.resnet import resnet34, resnet18

from torch.optim.adam import Adam

# 학습할 때 이용할 전처리 정의
transforms = Compose([
    RandomCrop((32, 32), padding=4),
    RandomHorizontalFlip(p=0.5),
    ToTensor(),
    Normalize(mean=(0.4914, 0.4822, 0.4465),
              std=(0.247, 0.243, 0.261))
])
```

# 데이터 없이 학습하는 GAN

▼ 교사 모델 학습에 필요한 요소 정의

# 학습용 데이터 준비

training\_data = CIFAR10(root=".",

CIFAR10 데이터

train=True,  
download=True,  
transform=transforms)

test\_data = CIFAR10(root=".",  
train=False,  
download=True,  
transform=transforms)

# 검증용 데이터 준비

train\_loader = DataLoader(  
training\_data,  
batch\_size=32,  
shuffle=True)

test\_loader = DataLoader(  
test\_data,  
batch\_size=32,  
shuffle=False)

device = "cuda" if torch.cuda.is\_available() else "cpu"

# 교사 모델 정의

teacher = resnet34(pretrained=False, num\_classes=10)  
teacher.to(device)

교사 모델 정의

lr = 1e-5  
optim = Adam(teacher.parameters(), lr=lr)

# 데이터 없이 학습하는 GAN

## 꽤 오래 걸림, 교사 모델 학습

▼ 교사 모델 학습 루프 정의

```
# 학습 루프
for epoch in range(30):
    iterator = tqdm.tqdm(train_loader)
    for data, label in iterator:
        optim.zero_grad()

        preds = teacher(data.to(device))

        loss = nn.CrossEntropyLoss()(preds, label.to(device))
        loss.backward()
        optim.step()

        iterator.set_description(f"epoch:{epoch+1} loss:{loss.item()}")

# 교사 모델의 가중치 저장
torch.save(teacher.state_dict(), "teacher.pth")
```

30번 반복

▼ 교사 모델 성능 평가하기

```
# 교사 모델의 가중치 불러오기
teacher.load_state_dict(torch.load("teacher.pth", map_location=device))

num_corr = 0

# 교사 모델의 성능 검증
with torch.no_grad():
    for data, label in test_loader:

        output = teacher(data.to(device))
        preds = output.data.max(1)[1]
        corr = preds.eq(label.to(device).data).sum().item()
        num_corr += corr

print(f"Accuracy:{num_corr/len(test_data)}")
```

Accuracy:0.8098

0.6 정도 나오네

# 데이터 없이 학습하는 GAN

여전히 Soft함의 덜함, 정답 Class 쪽은 넘 크고 나머지는 다 작게 극단적인 분포

```
import torch.nn.functional as F

class Generator(nn.Module):
    def __init__(self, dims=256, channels=3):
        super(Generator, self).__init__()

        # 256차원 벡터를 입력받아 128채널 8x8 이미지 생성
        self.l1 = nn.Sequential(nn.Linear(dims, 128 * 8 * 8))

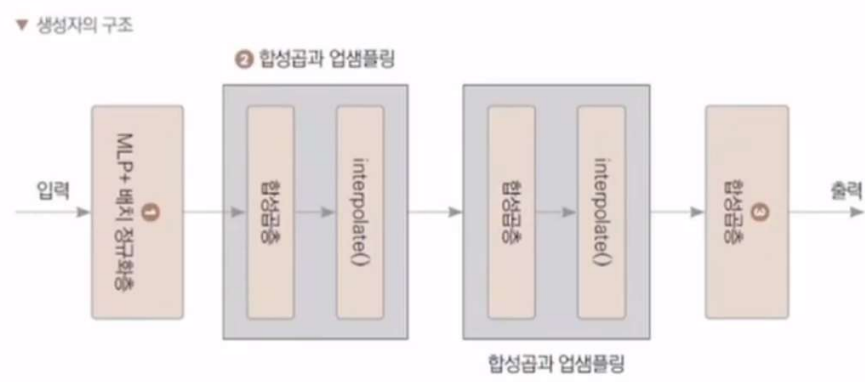
        self.conv_blocks0 = nn.Sequential(
            nn.BatchNorm2d(128),
        )
        self.conv_blocks1 = nn.Sequential(
            nn.Conv2d(128, 128, 3, stride=1, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2), # 1 활성화 함수
        )
        self.conv_blocks2 = nn.Sequential(
            nn.Conv2d(128, 64, 3, stride=1, padding=1),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, channels, 3, stride=1, padding=1),
            nn.Tanh(),
            nn.BatchNorm2d(channels, affine=False) # 2 배치 정규화
        )

    def forward(self, z):
        # 256차원 벡터를 128채널 8x8 이미지로 변환
        out = self.l1(z.view(z.shape[0], -1))
        out = out.view(out.shape[0], -1, 8, 8)

        out = self.conv_blocks0(out)
        # 3 이미지를 두 배로 늘려줌
        out = nn.functional.interpolate(out, scale_factor=2)
        out = self.conv_blocks1(out)
        out = nn.functional.interpolate(out, scale_factor=2)
        out = self.conv_blocks2(out)
        return out
```

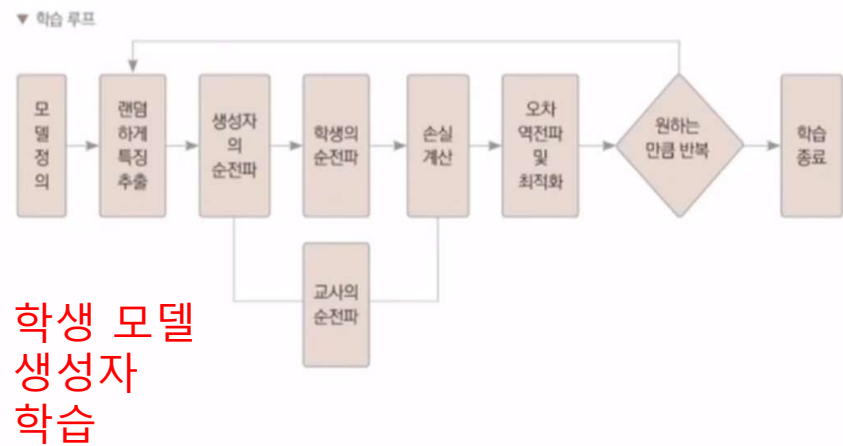
256차원 벡터 → 128X8X8 로 늘림, 8X8 image 개수만큼 늘려줌

순전파 부분





# 데이터 없이 학습하는 GAN



▼ 학생 모델과 생성자 학습

```
from torch.optim.sgd import SGD

# ❶ 교사 모델 불러오기
teacher = resnet34(pretrained=False, num_classes=10)
teacher.load_state_dict(torch.load("./teacher.pth", map_location=device))
teacher.to(device)
teacher.eval()

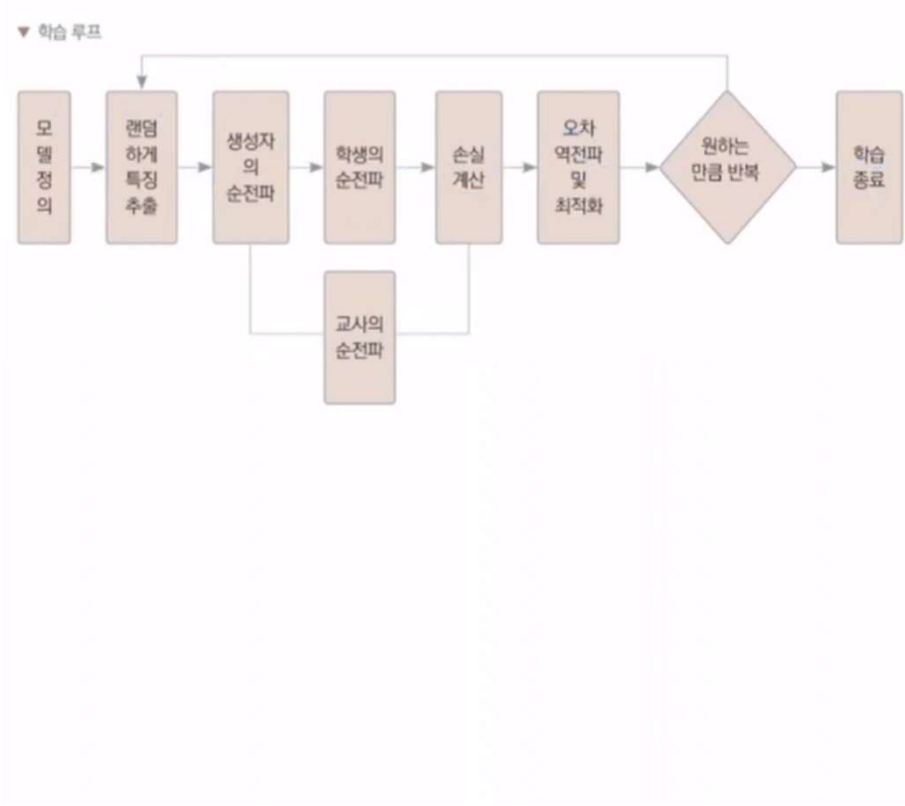
# ❷ 학생 모델 정의
student = resnet18(pretrained=False, num_classes=10)
student.to(device)

# ❸ 생성자 정의
generator = Generator()
generator.to(device)

# ❹ 생성자는 Adam으로, 학생 모델은 SGD를 이용해서 학습
G_optim = Adam(generator.parameters(), lr=1e-3)
S_optim = SGD(student.parameters(), lr=0.1, weight_decay=5e-4, momentum=0.9)
```

# 데이터 없이 학습하는 GAN

## 학생 모델의 학습



▼ 학습 루프 정의

```
for epoch in range(500):  
    # ❶ 학생 모델을 5번, 생성자는 1번 가중치 학습  
    for _ in range(5):  
        # ❶ 이미지 생성을 위한 노이즈 생성  
        noise = torch.randn(256, 256, 1, 1, device=device)  
        S_optim.zero_grad()  
        # ❷ 이미지 생성  
        fake = generator(noise).detach()  
        # ❸ 교사의 예측  
        teacher_output = teacher(fake)  
        # ❹ 학생의 예측  
        student_output = student(fake)  
        # ❺ 학생의 오차 계산  
        S_loss = nn.L1Loss()(student_output, teacher_output.detach())  
  
        print(f"epoch{epoch}: S_loss {S_loss}")  
        # ❻ 오차 역전파  
        S_loss.backward()  
        S_optim.step()
```

256배치크기, 256 특징

가짜 이미지 학생, 교사 모델에 넣기

# 데이터 없이 학습하는 GAN

## ▼ 생성자 학습

```
# ❶ 이미지 생성을 위한 노이즈 정의
noise = torch.randn(256, 256, 1, 1, device=device)
G_optim.zero_grad()

# ❷ 이미지 생성
fake = generator(noise)

# ❸ 교사와 학생 모델의 출력 계산
teacher_output = teacher(fake)
student_output = student(fake)

# ❹ 생성자의 오차 계산
G_loss = -1 * nn.L1Loss()(student_output, teacher_output)

# ❺ 오차 역전파
G_loss.backward()
G_optim.step()

print(f"epoch{epoch}: G_loss {G_loss}")
```

부호만 반대로 교사와 학생이 다른 출력 내도록

## 성능 평가 부분

### ▼ 학생 모델 성능 평가하기

```
num_corr = 0

student.load_state_dict(
    torch.load("student.pth", map_location=device))

# 학습용 데이터에 대한 정확도
with torch.no_grad():
    for data, label in train_loader:

        output = student(data.to(device))
        preds = output.data.max(1)[1]
        corr = preds.eq(label.to(device).data).sum().item()
        num_corr += corr

    print(f"Accuracy:{num_corr/len(training_data)}")

num_corr = 0

# 검증용 데이터에 대한 정확도
with torch.no_grad():
    for data, label in test_loader:

        output = student(data.to(device))
        preds = output.data.max(1)[1]
        corr = preds.eq(label.to(device).data).sum().item()
        num_corr += corr

    print(f"Accuracy:{num_corr/len(test_data)}")

Accuracy:0.50836
Accuracy:0.5003
```

가장 높은 index에 대해서 맞췄는지  
맞춘 개수 더해서

0.1... 정도 나오는데