KU KONKUK UNIVERSITY

Generative Adversarial Network

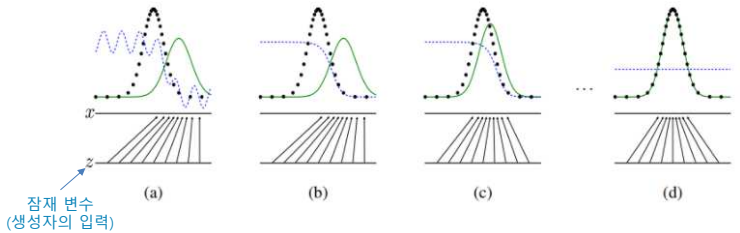
자연어처리연구실

Generative Adversarial Network (GAN)

- 적대적 생성 신경망
  - 고품질의 이미지를 생성하기 위해서 생성자(generator)와 판별자(discriminator)를 경쟁적 학습시키는 인공 신경망 구조

그림 출처: https://dreamgonfly.github.io/blog/gan-explained/

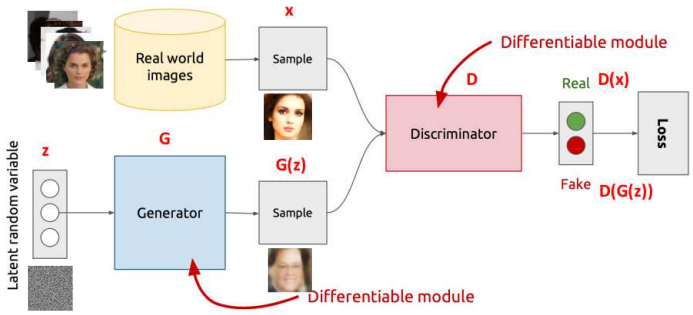
Adversarial Learning Process



- 검은 점선: 학습 데이터의 분포 → $P_{data}(X)$
- 녹색 실선: 생성자에 의해 생성된 데이터의 분포 → $P_{Gen}(X)$
- 파란 점선: 판별자의 출력 → $0 < P_{Dis}(X) < 1$

그림 출처: Ian Goodfellow의 GAN 논문

Architecture of GAN



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

그림 출처: 스탠포드 대학교 Fei-Fei 교수 강의자료

Training Discriminator

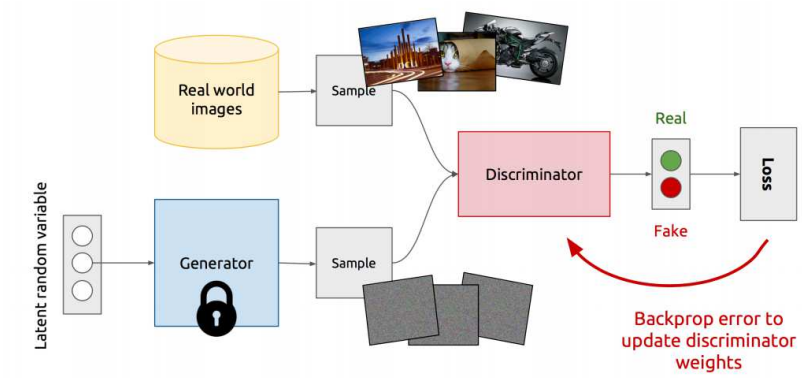


그림 출처: 스탠포드 대학교 Fei-Fei 교수 강의자료

Training Generator

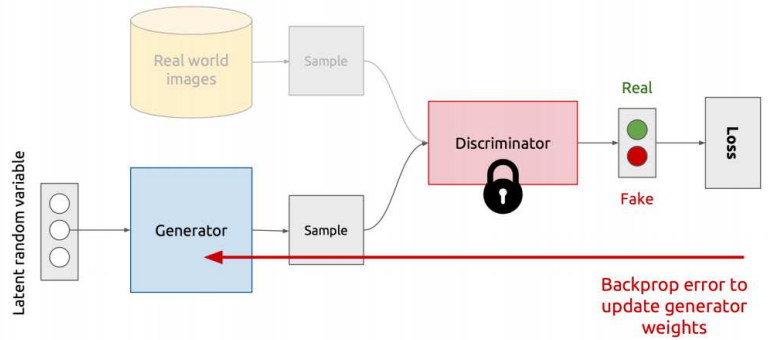


그림 출처: 스탠포드 대학교 Fei-Fei 교수 강의자료

Object of GAN Training

- 목적 함수

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

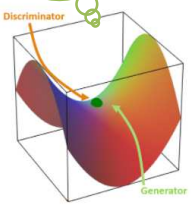
D(x): 본인의 리워드 최대화

$$\begin{cases} \log(D(x)) = \log(1) = 0, if\ x\ is\ a\ real\ image \\ \log(1 - D(G(z))) = \log(1) = 0, if\ G(z)\ is\ a\ fake\ image \end{cases}$$

Nash 평형(equilibrium)

$P_{data}(x) = P_{gen}(x)\ \forall x$

$D(x) = \frac{1}{2}\ \forall x$

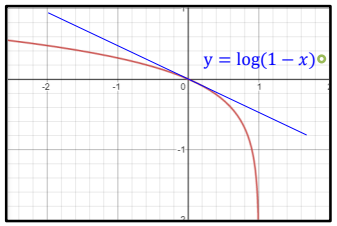


G(z): D(x)의 리워드 최소화

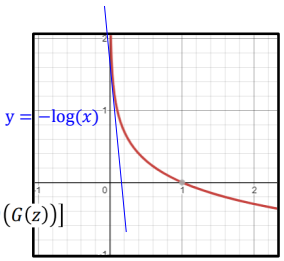
$$\begin{cases} \log(1 - D(G(z))) = \log(1 - 1) = -\infty, if\ G(z)\ is\ a\ real - like\ image \\ \log(1 - D(G(z))) = \log(1 - 0) = 0, if\ G(z)\ is\ a\ fake - like\ image \end{cases}$$

Changing Object Function

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



초기 G(z)는 판별하기 너무 쉬운 이미지 생성 → x=0, Small gradient!



Minimize $-\mathbb{E}_{z\sim q(z)}[\log D(G(z))]$

Limitation of Vanilla GAN

GANs are still evolving!

- Non-Convergence Problem
  - 양쪽이 조금씩 발전해야 내쉬 평형 상태가 되는데 한쪽이 강력해지면 다른 쪽은 발전하지 못하는 문제
- Mode-Collapse Problem
  - 제대로 학습되지 못한 경우에 다양한 이미지를 만드는 것이 아니라 비슷한 이미지만 계속 생성하는 문제

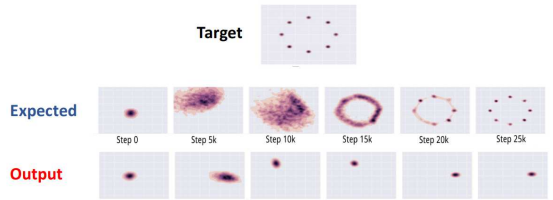
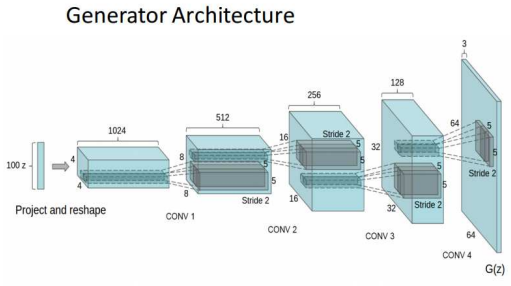


그림 출처: Metz et al. "Unrolled Generative Adversarial Networks", arXiv, 2016

Deep Convolutional GAN (DCGAN)



Key ideas:

- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

그림 출처: 스탠포드 대학교 Fei-Fei 교수 강의자료

Semi-Supervised GAN (SGAN)

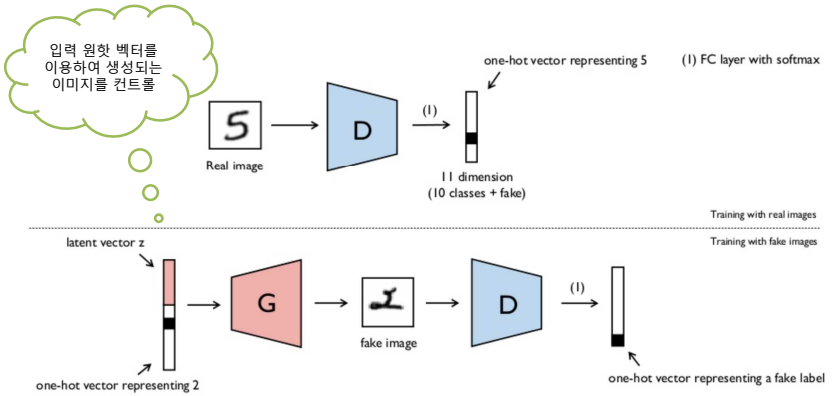


그림 출처: Augustus Odena et al. "Semi-Supervised Learning with Generative Adversarial Networks", 2016

Other GANs

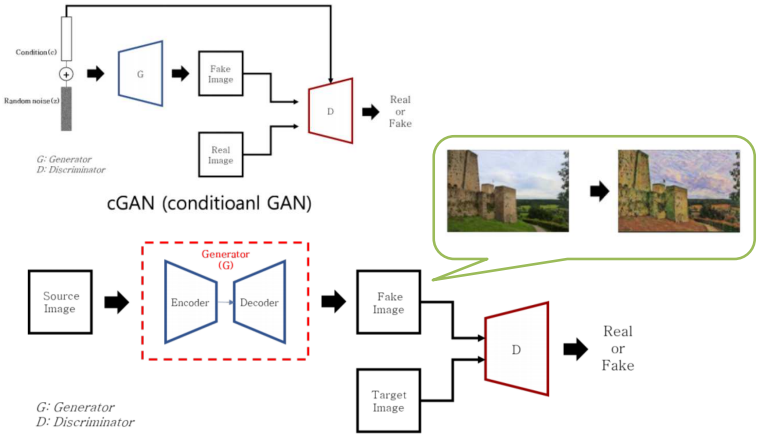


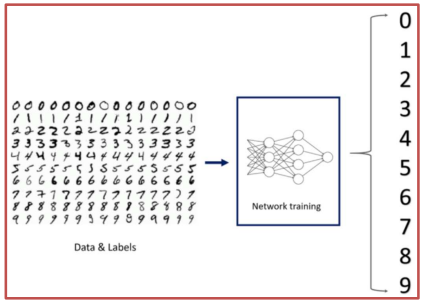
그림 출처: 장성수, 2021년도 건국대 석사학위논문

실습

실습 코드 다운로드:
https://github.com
/KUNLP/Lecture

- MNIST 데이터를 활용하여 손글씨를 생성하는 프로그램을 작성하시오.

  - 입력 데이터셋
    - MNIST dataset
    - 0~9 손 글씨 이미지에 대한 픽셀 값 데이터
  - 문제
    - 이미지의 픽셀 값을 입력으로 하여 해당 이미지가 0~9 중에 어떤 숫자인지 분류



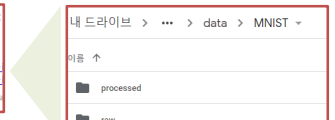
실습

```
import os
import numpy as np
import torch
import torch.nn as nn
from torch.autograd import Variable
from torch.utils.data import (DataLoader, RandomSampler, TensorDataset)
import torchvision.utils as utils
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from matplotlib import pyplot as plt
```

```
# 데이터 읽기 함수
def load_dataset():
  standardizator = transforms.Compose([transforms.ToTensor(), transforms.Normalize(mean=(0.5), std=(0.5))])
  # MNIST dataset
  train_X = datasets.MNIST(root='/gdrive/My Drive/colab/gan/mnist/data/', train=True, transform=standardizator, download=True)
  return train_X
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 503: Service Unavailable

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-
9913344/? [00:09<00:00, 1052434
```




```
def imshow_grid(img):
    img = utils.make_grid(img.cpu().detach())
    img = (img+1)/2
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1,2,0)))
    plt.show()
```

실습

```
##############################
#      Discriminator 설계     #
##############################

class MNIST_Discriminator(nn.Module):

  def __init__(self, config):
    super(MNIST_Discriminator, self).__init__()

    # 입력층 노드 수
    self.inode = config["d_input_node"]
    # 은닉층 노드 수
    self.hnode = config["d_hidden_node"]
    # 출력층 노드 수: 분류해야 하는 레이블 수
    self.onode = config["d_output_node"]

    # 신경망 설계
    self.net = nn.Sequential(nn.Linear(self.inode, self.hnode, bias=True),
                             nn.LeakyReLU(),
                             nn.Dropout(0.1),
                             nn.Linear(self.hnode, self.hnode, bias=True),
                             nn.LeakyReLU(),
                             nn.Dropout(0.1),
                             nn.Linear(self.hnode, self.hnode, bias=True),
                             nn.LeakyReLU(),
                             nn.Dropout(0.1),
                             nn.Linear(self.hnode, self.onode, bias=True),
                             nn.Sigmoid())

  def forward(self, input_features):
    hypothesis = self.net(input_features)
    return hypothesis
```

```
##############################
#       Generator 설계       #
##############################

class MNIST_Generator(nn.Module):

  def __init__(self, config):
    super(MNIST_Generator, self).__init__()

    # 입력층 노드 수
    self.inode = config["g_input_node"]
    # 은닉층 노드 수
    self.hnode = config["g_hidden_node"]
    # 출력층 노드 수: 생성해야 하는 노드 수
    self.onode = config["g_output_node"]

    # 신경망 설계
    self.net = nn.Sequential(nn.Linear(self.inode, self.hnode, bias=True),
                             nn.LeakyReLU(),
                             nn.Dropout(0.1),
                             nn.Linear(self.hnode, self.hnode, bias=True),
                             nn.LeakyReLU(),
                             nn.Dropout(0.1),
                             nn.Linear(self.hnode, self.hnode, bias=True),
                             nn.LeakyReLU(),
                             nn.Dropout(0.1),
                             nn.Linear(self.hnode, self.onode, bias=True),
                             nn.Tanh())

  def forward(self, input_features):
    hypothesis = self.net(input_features)
    return hypothesis
```


실습

```
# 모델 학습 함수
def train(config):

  # Discriminator와 Generator 모델 생성
  ?

  # 데이터 읽기
  input_features = load_dataset()

  # DataLoader를 통해 배치(batch) 단위로 데이터를 나누고 셔플(shuffle)
  train_dataloader = DataLoader(input_features, shuffle=True, batch_size=config["batch_size"])

  # 바이너리 크로스엔트로피 비용 함수
  loss_func = nn.BCELoss()

  # Discriminator와 Generator 옵티마이저 함수 지정
  ?

  for epoch in range(config["epoch"]+1):

    # Discriminator와 Generator 학습 모드 셋팅
    ?

    # epoch 마다 평균 비용을 저장하기 위한 리스트
    D_costs, G_costs = [], []
```

```
    for (step, batch) in enumerate(train_dataloader):

      # batch = (input_features[step], labels[step])*batch_size
      # .cuda()를 통해 메모리에 업로드
      batch = tuple(t.cuda() for t in batch)

      # 배치 크기 만큼 입력 데이터 읽기
      ?
```

실습

```
##############################
#      Discriminator 학습     #
##############################

# 역전파 변화도 초기화
D_optimizer.zero_grad()

# 진짜에 대한 비용 계산
real_hypothesis = D(input_features)
real_labels = Variable(torch.ones(config["batch_size"],1)).cuda()
real_cost = loss_func(real_hypothesis, real_labels)

# 가짜에 대한 비용 계산
fake_input_features = get_noise(config["batch_size"],config["g_input_node"]).cuda()
fake_hypothesis = D(G(fake_input_features))
fake_labels = Variable(torch.zeros(config["batch_size"],1)).cuda()
fake_cost = loss_func(fake_hypothesis, fake_labels)

# 전체 비용 역전파 수행
total_cost = real_cost + fake_cost
total_cost.backward()
D_optimizer.step()

# 현재 batch의 스텝 별 Discriminator 비용 저장
D_costs.append(total_cost.data.item())
```

```
def get_noise(batch_size=16, g_input_node=100):
    return torch.randn(batch_size, g_input_node)
```

```
##############################
#       Generator 학습       #
##############################

# 역전파 변화도 초기화
G_optimizer.zero_grad()

# 가짜에 대한 비용 계산
fake_input_features = get_noise(config["batch_size"],config["g_input_node"]).cuda()
fake_hypothesis = D(G(fake_input_features))
fake_labels = Variable(torch.ones(config["batch_size"],1)).cuda()
fake_cost = loss_func(fake_hypothesis, fake_labels)

# 가짜 생성 비용 역전파 수행
fake_cost.backward()
G_optimizer.step()

# 현재 batch의 스텝 별 Generator 비용 저장
G_costs.append(fake_cost.data.item())
```


Edited by Harksoo Kim

실습

```
# 10 에폭마다 중간 결과 출력 및 저장
if epoch%10 == 0:

  # 평균 비용 출력
  print("Avg Loss D={0:f}, Avg Loss G={1:f}".format(np.mean(D_costs), np.mean(G_costs)))

  # Generator 저장
  torch.save(G.state_dict(), os.path.join(config["output_dir"], "epoch_{0:d}.pt".format(epoch)))

  # 생성된 샘플 출력
  do_test(G,config['g_input_node'])
```

```
def do_test(model, input_node):
  # 평가 모드 셋팅
  model.eval()

  with torch.no_grad():

    X = get_noise(g_input_node=input_node).cuda()
    hypothesis = model(X)
    hypothesis_ = hypothesis.reshape((-1,28,28)).unsqueeze(1)
    imshow_grid(hypothesis_)
```

```
def get_noise(batch_size=16, g_input_node=100):
    return torch.randn(batch_size, g_input_node)
```


Edited by Harksoo Kim

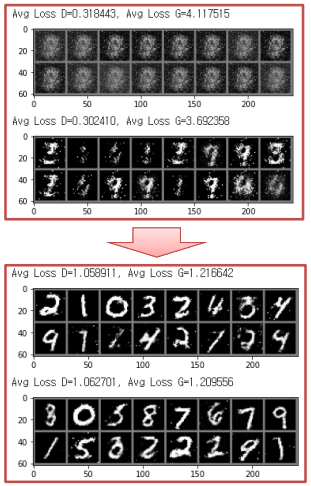
실습

```
if(__name__=="__main__"):

    root_dir = "/gdrive/My Drive/colab/gan/mnist"
    output_dir = os.path.join(root_dir, "output")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    config = {"model_name":"epoch_{0:d}.pt".format(10),
              "root_dir":root_dir,
              "output_dir":output_dir,
              "d_input_node":784,
              "d_hidden_node":256,
              "d_output_node":1,
              "g_input_node":100,
              "g_hidden_node":256,
              "g_output_node":784,
              "learn_rate":0.0002,
              "batch_size":100,
              "epoch":100,
              }

    train(config)
```



Edited by Harksoo Kim

질의응답

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr

Edited by Harksoo Kim