PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Keras Additional Functions

GyeongYeong Kim

ML/DL Lab 3rd Day

PIAI Research Department

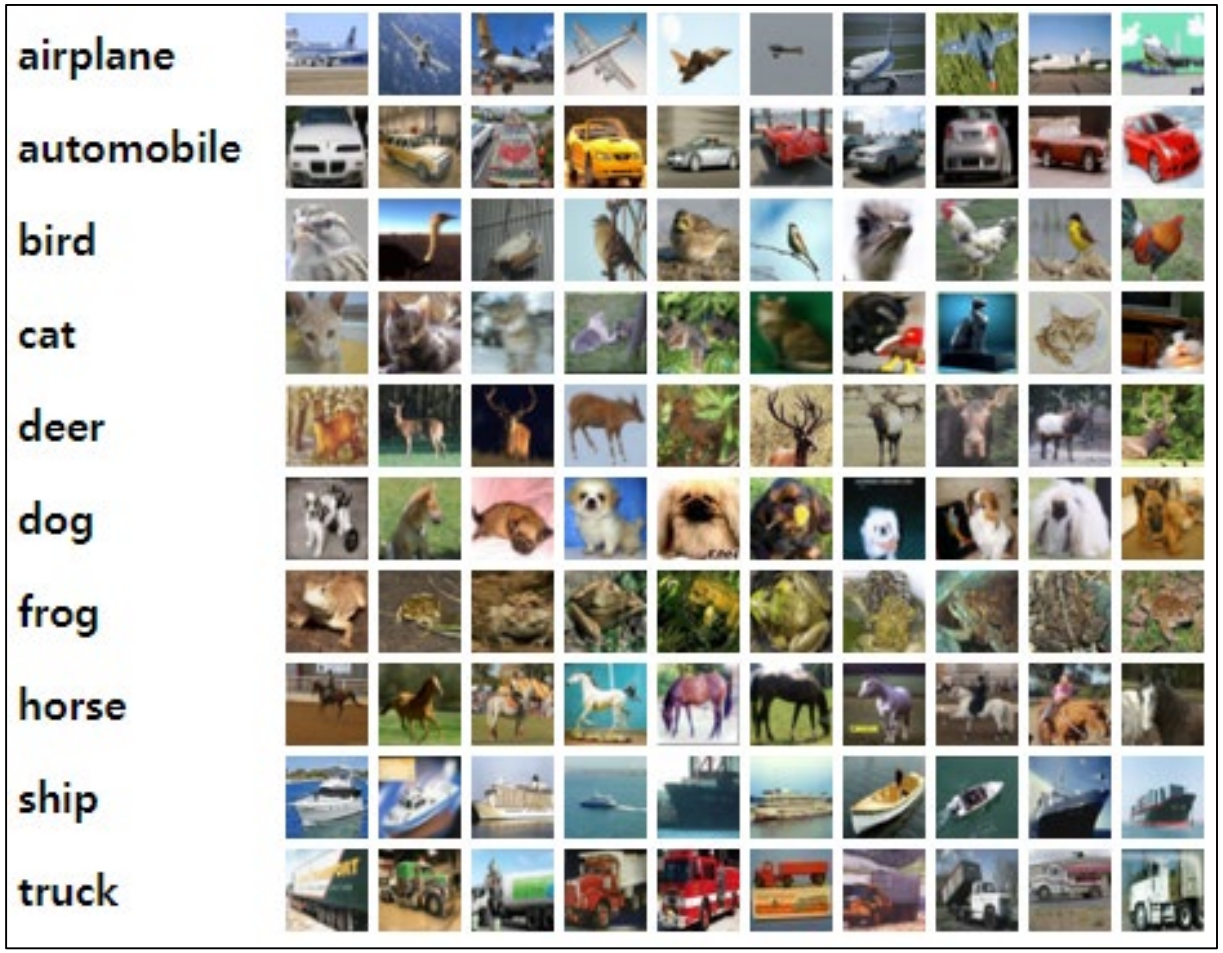
가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Keras Additional Functions
  - CIFAR10 Introduction
  - Model Summation
  - Data Augmentation
  - Transfer Learning
  - LSTM
  - GAN
- Pytorch Additional Functions
  - CIFAR10 Introduction
  - Data Augmentation
  - Transfer Learning
  - Training Customized Dataset
  - GAN

CNN – CIFAR 10

PIAI Research Department

- **데이터셋 소개 – CIFAR 10**



- CIFAR-10은 열 가지의 class로 이루어진 공인 Dataset
- 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭으로 구성
- 32 × 32 픽셀의 3채널 RGB 이미지
- Class당 6,000개의 이미지로 이루어져 있으며 총 60,000장
- 5만장이 Train / 1만장이 Test
- 머신러닝 / 딥러닝 챌린징 분야에서 널리 사용되는 데이터셋

데이터셋 다운로드 링크: https://www.cs.toronto.edu/~kriz/cifar.html

Model Summation

PIAI Research Department


```python
def conv_maxpool_layers(n_in):
    # Coding Time
    model = Sequential()
    model.add(Conv2D(16, kernel_size=(3, 3), padding='same', activation='relu', input_shape=(n_in)))
    model.add(Conv2D(32, kernel_size=(3, 3), padding='same', strides=(2, 2), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

def fc_layers(n_out):
    # Coding Time
    model = Sequential()
    model.add(Dense(units =128, input_shape=(2048,), activation='relu'))
    model.add(Dense(units =n_out, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

```python
def CNN_sum(n_in, n_out):
    feature_extractor=conv_maxpool_layers(n_in)
    feature_extractor.trainable=True
    ann_classifier = fc_layers(n_out)
    ann_classifier.trainable=True

    x = Input(shape=n_in)
    feature = feature_extractor(x)
    y = ann_classifier(feature)
    model = Model(inputs = x, outputs = y)
```

- trainable 특정 layer 중심으로 학습 여부를 설정할 수 있음 -> Fine Tuning 같은 역할을 할 수 있음
- Feature_extractor, ann_classifier이 customized layer같은 성격을 가질 수 있게 됨
- VGG / ResNet / InceptionNet에 있는 layer block 역할처럼도 쓰임!

Data Augmentation & Transfer Learning

PIAI Research Department

Collecting data is the most difficult thing

Data Augmentation & Transfer Learning

PIAI Research Department

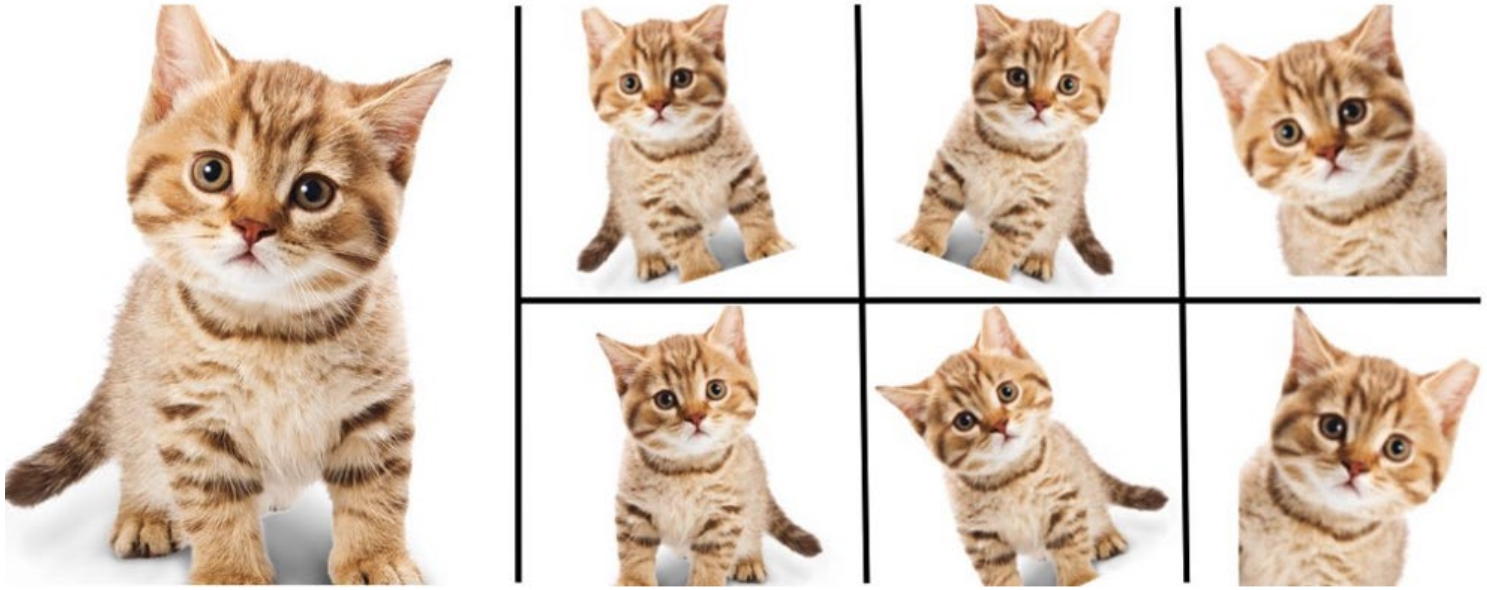
Collecting data is
the most difficult thing

How to solve?

- Data Augmentation
- Transfer Learning
- Self-supervised Learning
- Semi-supervised Learning
- …

Data Augmentation

PIAI Research Department



Enlarge your Dataset

Keras ImageDataGenerator

```
datagen = ImageDataGenerator(
    featurewise_center = False,
    samplewise_center = False,
    featurewise_std_normalization = False,
    samplewise_std_normalization = False,
    zca_whitening = False,
    rotation_range = 2, # 회전
    zoom_range = 0.1, # 확대 축소
    width_shift_range = 0.1, # 수평 이동
    height_shift_range = 0.1, # 수직 이동
    horizontal_flip = True, # 수평 반전
    vertical_flip = False # 수직 반전
)

datagen.fit(X_train)
```

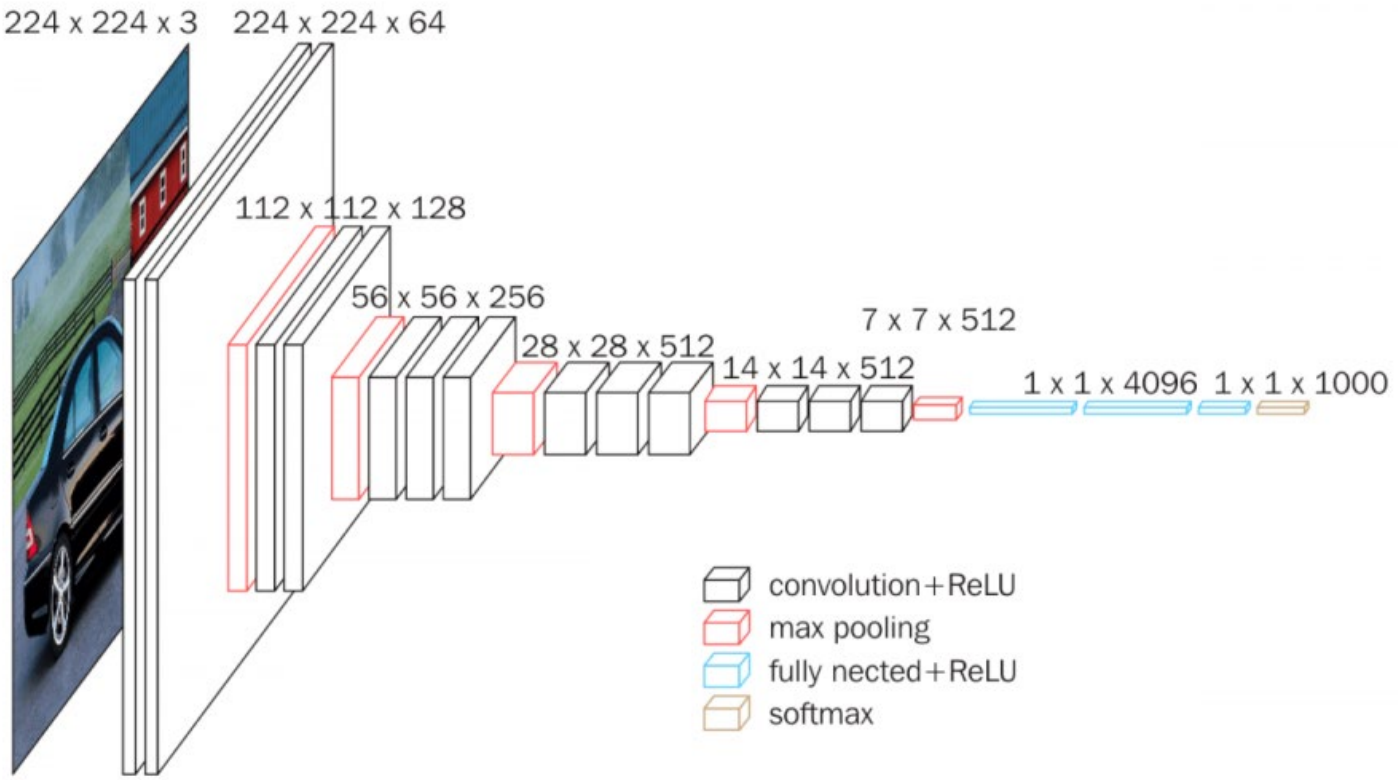
```
model.fit(X_train, Y_train,
```

```
model.fit(datagen.flow(X_train[:-10000], Y_train[:-10000], batch_size = 1000),
```

Transfer Learning

PIAI Research Department

VGGNet

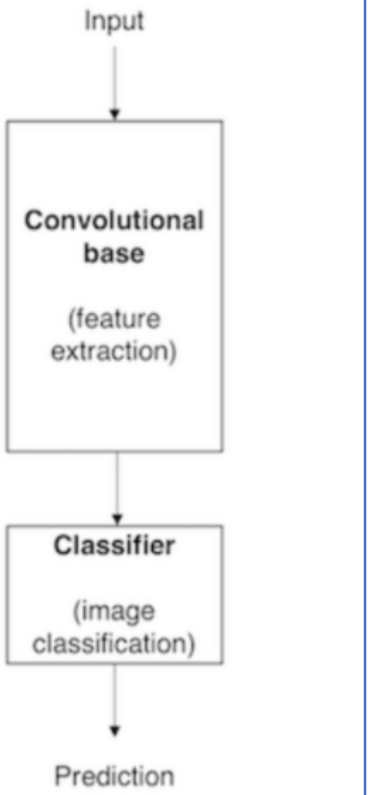


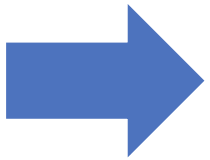
처음부터 모델을 구성하는 것이 아니라 기존에 존재하는 성능 좋은 모델을 가져와 학습하는 기법

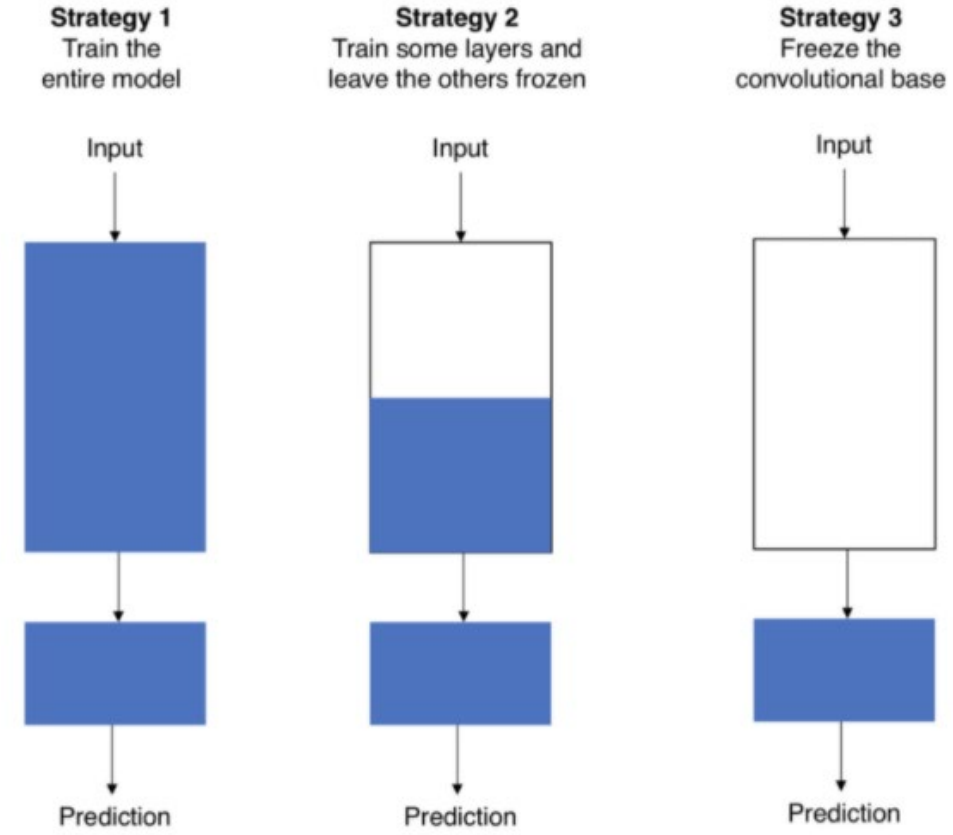
Transfer Learning

PIAI Research Department

Pretrained model



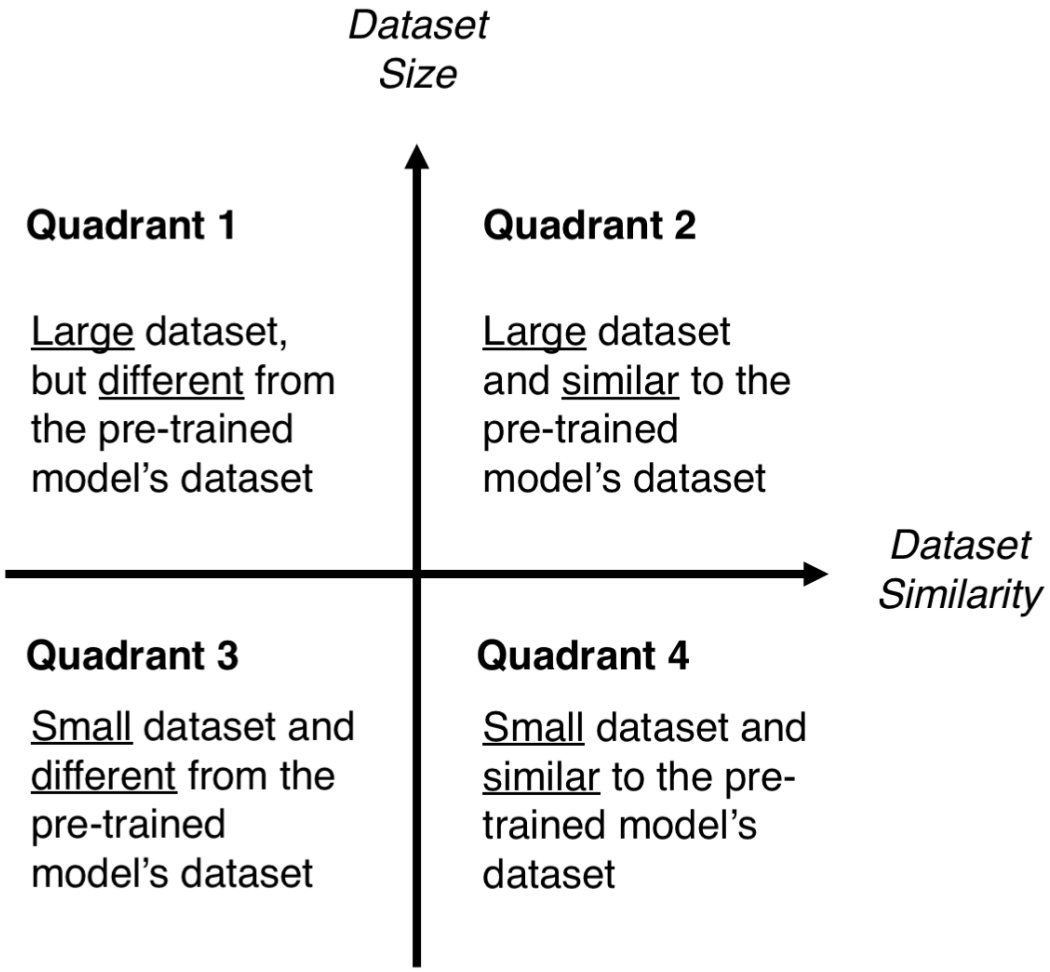


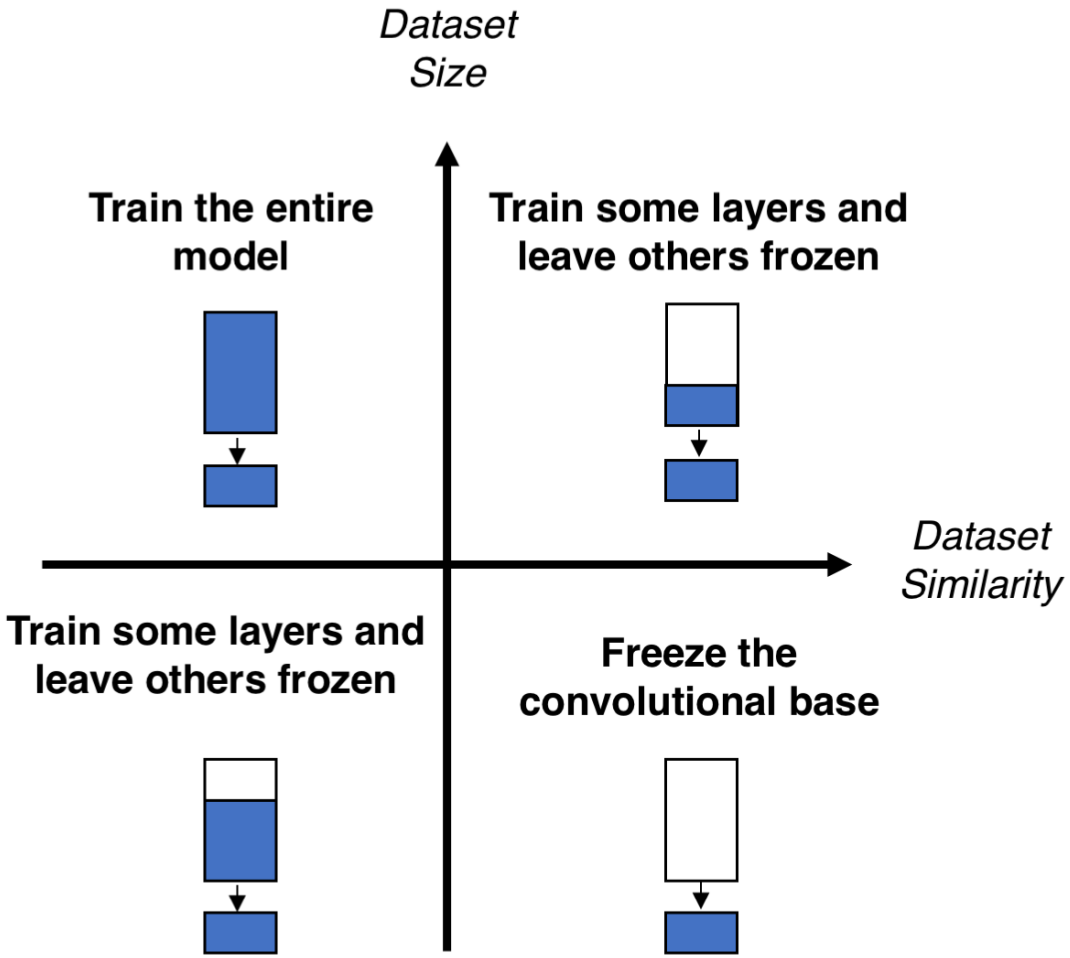


Fine Tuning

Transfer Learning

PIAI Research Department





LSTM

PIAI Research Department

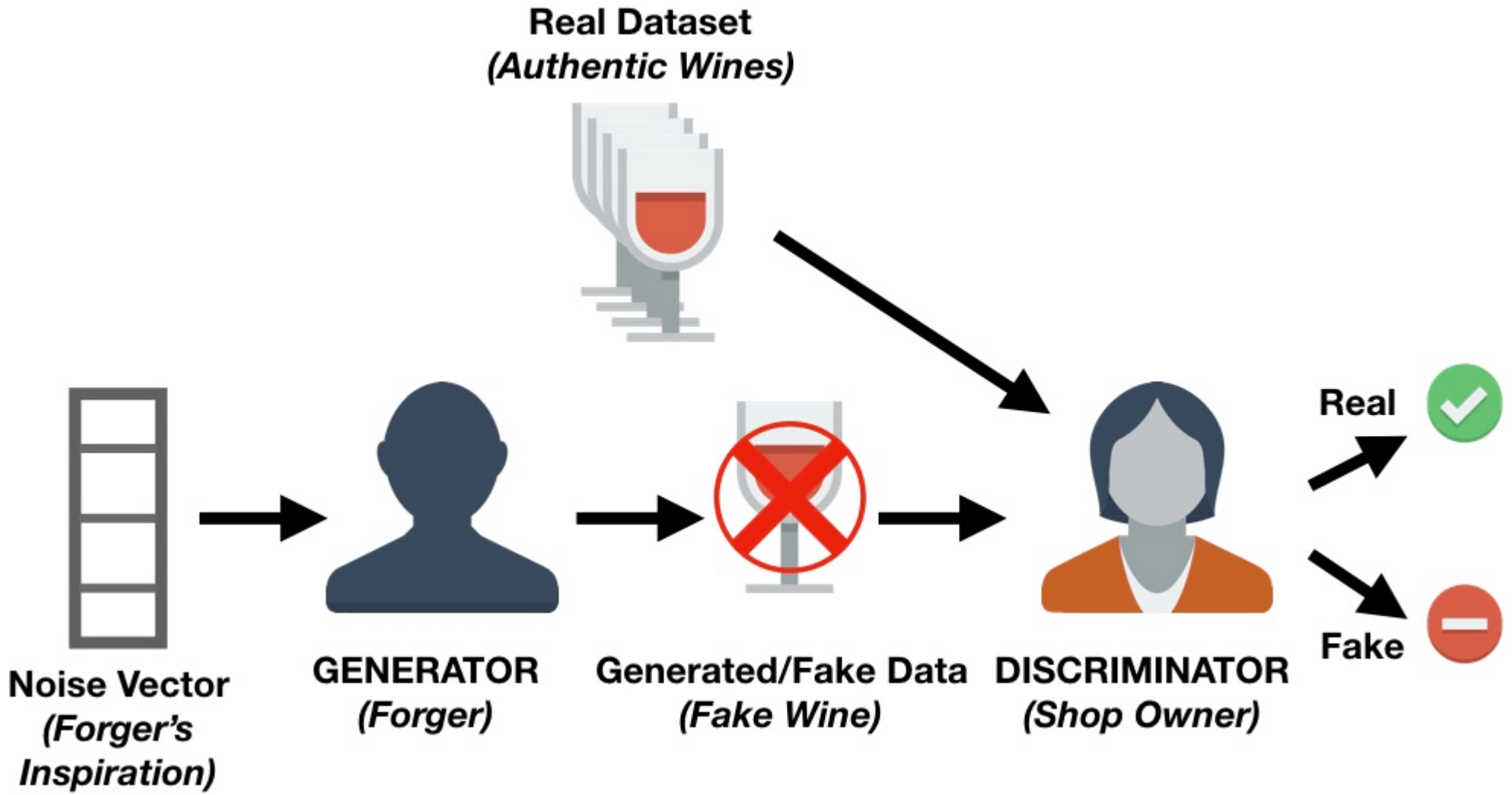

```
def lstm(n_in, n_out):
    # Coding Time
    model = Sequential()
    model.add(LSTM(30, input_shape=(28,28)))
    model.add(Dense(n_out, activation='softmax'))
    return model

model = lstm(n_in, n_out)
```

종류

- **RNN (MinimalRNNCell)**
- **SimpleRNN (SimpleRNNCell)**
- **GRU (GRUCell) (CuDNNGRU)**
- **LSTM (LSTMCell) (CuDNNLSTM)**
- **ConvLSTM2D**
- **Bidirectional**

(https://keras.io/ko/layers/recurrent/)

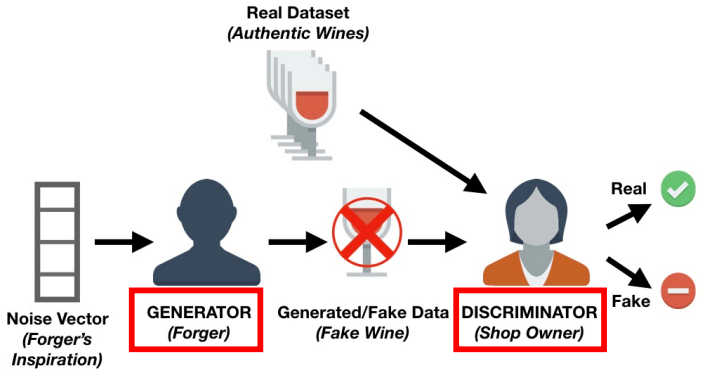
GAN


Real Dataset
(Authentic Wines)
Noise Vector
(Forger's
Inspiration)
GENERATOR
(Forger)
Generated/Fake Data
(Fake Wine)
DISCRIMINATOR
(Shop Owner)
Real
Fake

GAN

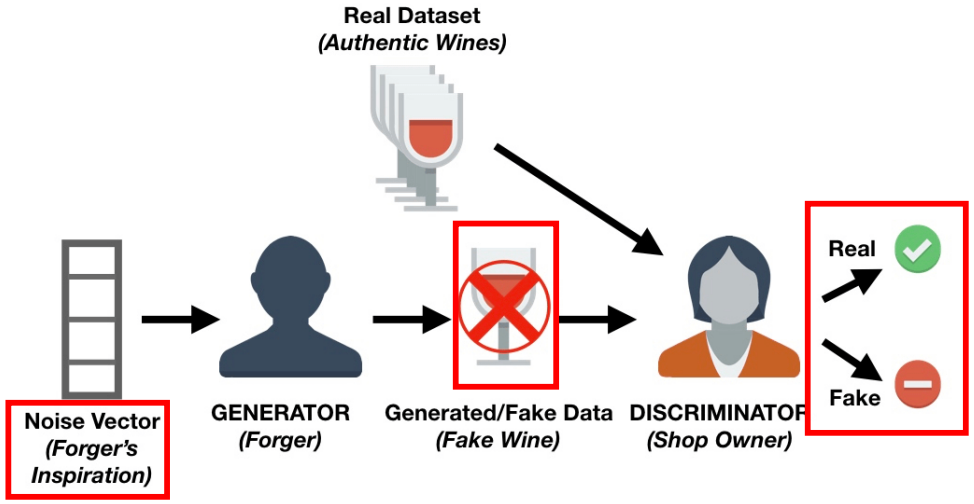
PIAI Research Department

```
def get_generator(optimizer):
    generator = Sequential()
    generator.add(Dense(256, input_dim=random_dim, kernel_initializer=initializers.RandomNormal(stddev=0.02)))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(512))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(1024))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(784, activation='tanh'))
    generator.compile(loss='binary_crossentropy', optimizer=optimizer)
    return generator


def get_discriminator(optimizer):
    discriminator = Sequential()
    discriminator.add(Dense(1024, input_dim=784, kernel_initializer=initializers.RandomNormal(stddev=0.02)))
    discriminator.add(LeakyReLU(0.2))
    discriminator.add(Dropout(0.3))

    discriminator.add(Dense(512))
    discriminator.add(LeakyReLU(0.2))
    discriminator.add(Dropout(0.3))

    discriminator.add(Dense(256))
    discriminator.add(LeakyReLU(0.2))
    discriminator.add(Dropout(0.3))

    discriminator.add(Dense(1, activation='sigmoid'))
    discriminator.compile(loss='binary_crossentropy', optimizer=optimizer)
    return discriminator
```



GAN

PIAI Research Department




```
def get_gan_network(discriminator, random_dim, generator, optimizer):

    discriminator.trainable = False # Generator와 Discriminator를 동시에 학습시 trainable을 False로 설정

    gan_input = Input(shape=(random_dim,)) # gan_input : 노이즈(100 차원)

    x = generator(gan_input) # X:이미지

    gan_output = discriminator(x) # gan_output : 이미지가 진짜인지 아닌지에 대한 확률

    gan = Model(inputs=gan_input, outputs=gan_output)
    gan.compile(loss='binary_crossentropy', optimizer=optimizer)
    return gan
```

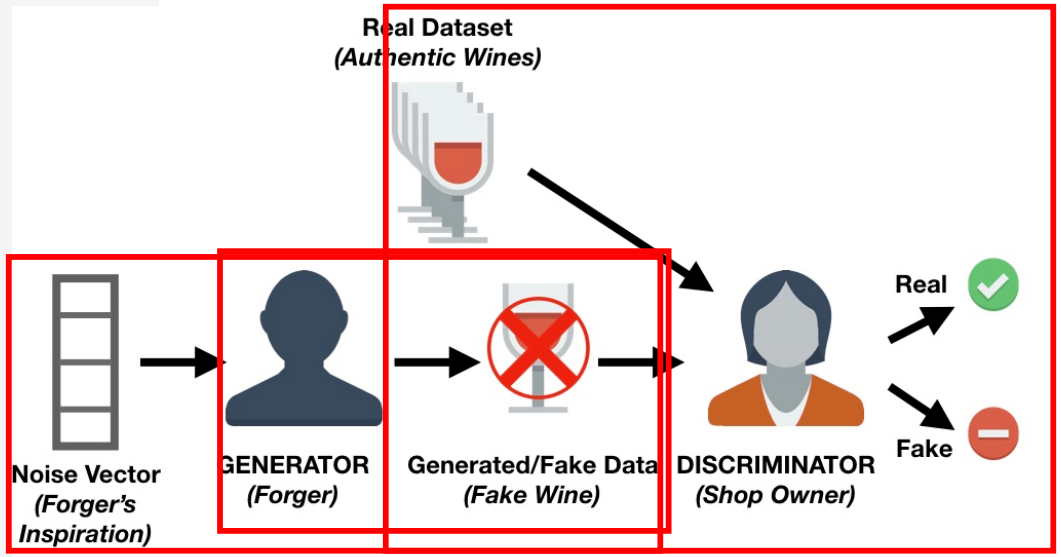
GAN

PIAI Research Department

```python
# Generator를 통해 MNIST 이미지를 생성
generated_images = generator.predict(noise, verbose =0)
X = np.concatenate([image_batch, generated_images])
```

```python
# Discriminator 학습
y_dis = np.zeros(2*batch_size)
y_dis[:batch_size] = 0.9
discriminator.trainable = True
discriminator.train_on_batch(X, y_dis)
```

```python
# Generator 학습
noise = np.random.normal(0, 1, size=[batch_size, random_dim])
y_gen = np.ones(batch_size)
discriminator.trainable = False
gan.train_on_batch(noise, y_gen)
```



GAN

PIAI Research Department

1 Epoch

20 Epoch

GAN

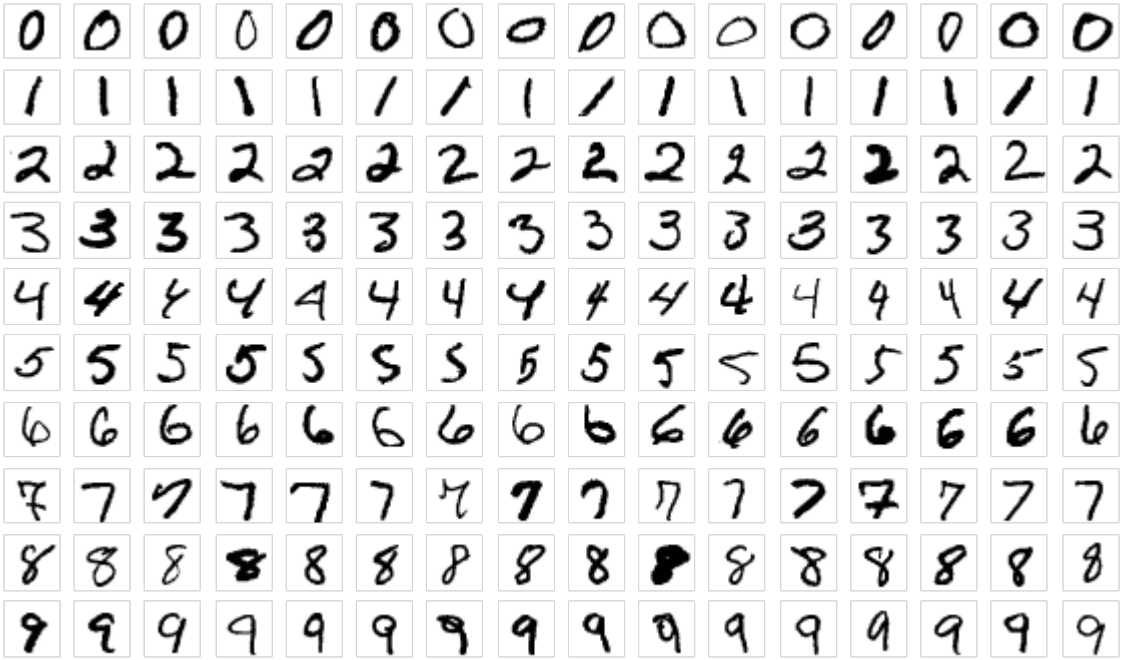
PIAI Research Department

100 Epoch

400 Epoch

GAN – Another Data Augmentation?!

PIAI Research Department



Ground Truth

400 Epoch

Code Running

3. Keras CNN – Additional Functions

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

Machine Learning Analysis

Machine Learning Analysis

PIAI Research Department

- 머신 러닝 분석 이슈

데이터

학습

성능

서비스

데이터

포맷 : 학습하기에 적정한 포맷인가?

규모 : 모델을 학습할 수 있을만큼 충분한 양인가? (Quantity / Level)

품질 : Outlier가 학습에 악영향을 끼칠만큼 있는가?

- 데이터 Outlier : 저품질이거나 대표성이 모호한 Outlier, 중복되는 데이터 (Noise Ratio)
- 데이터-라벨 Outlier : 데이터와 라벨에 대한 매칭이 잘못된 Outlier
- 학습/검증 Outlier : 학습과 검증하는 데이터의 편차 여부(10-fold validation)

구성 : 학습에 적정한 구성을 가지는가? (Imbalance / Bias / Breadth / Depth / 클래스 별 similarity)

라벨링 특성 : 데이터와 라벨의 구성이 합당한가? (성별 / 나이 / 표정)

학습

학습 방법 : 어떤 머신러닝 방식을 쓰는 것이 합당한가? (지도학습 / 비지도 학습 / …)

데이터 연계 : 학습/ 평가하기에 적합한 전처리가 잘 진행되었는가?

네트워크 구조 :

- Layer나 Activation이 잘 설정 되었는가?(층 수, 종류, Input/Output 크기 등)
- 학습할 Parameter 수가 적정한가?

하이퍼 파라미터 : activation, loss, optimizer, learning_rate, metric들을 잘 조정했는가?

학습 방법 : learning rate schedule, batch_size, epoch, validation 구성이 적정한가?

학습 효율 : loss나 정확도가 충분히 수렴되었는가?

성능

Metric : 모델의 성능을 확인하기에 적절한 metric인가? (정량 성능과 실제 성능의 괴리)

예상치 : 모델의 성능이 예상되는 값의 범주에 속하는가? (모델의 성능 한계 예측)

Bias : 모델이 편향되도록 분류하거나 하진 않는가?

에러 패턴 : 결과 에러 중 흔하게 발생하는 경우가 있는가?

개선 방향성 : 네트워크? 학습 방법? 데이터 개선?

서비스

도메인 적합성 : 학습되는 데이터와 실제 사용될 데이터와의 차이가 없는가?

호환성 : 개발 모델이 실제 환경에 호환이 가능한 환경인가?(GPU, 이미지 크기 등)

속도 : 해당 모델이 필요한 서비스에 적용가능한 속도인가? (Speed / Accuracy TradeOff)

백엔드 여부 : 모델 내 나타는 문제를 보완할 백엔드 기술이 필요한가?

적용성 : 쉬운 문제를 어렵게 푼건 아닐까..? (End-To-End / Process System)

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

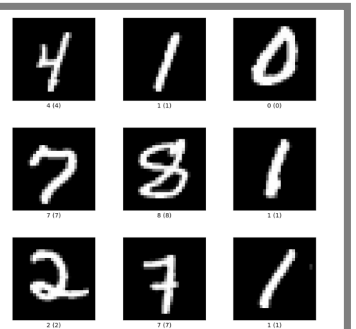
PyTorch Additional Functions

I-Eon, Na

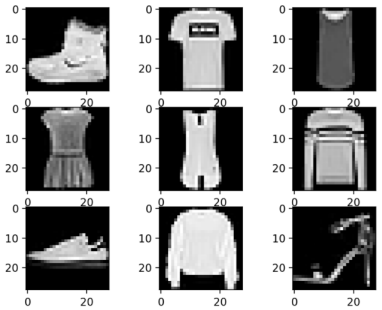
CNN – 공인 데이터셋

PIAI Research Department

이전 수업에서 사용



MNIST



Fashion MNIST

EMNIST

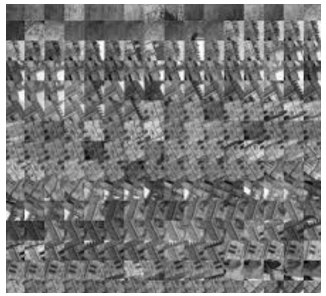
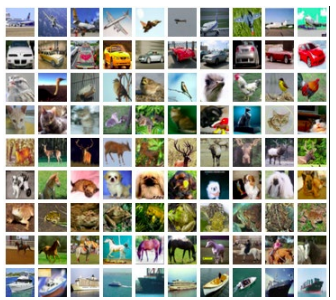


Photo tour

Flickr



CIFAR-10

- 머신러닝/딥러닝 모델을 평가하기 위한 다양한 공인 데이터셋
- MNIST와 다르게 RGB 3채널인 CIFAR-10
- 기타 PyTorch 데이터셋:

  https://teddylee777.github.io/pytorch/pytorch-mnist-dataloader-loading%ED%95%98%EA%B8%B0

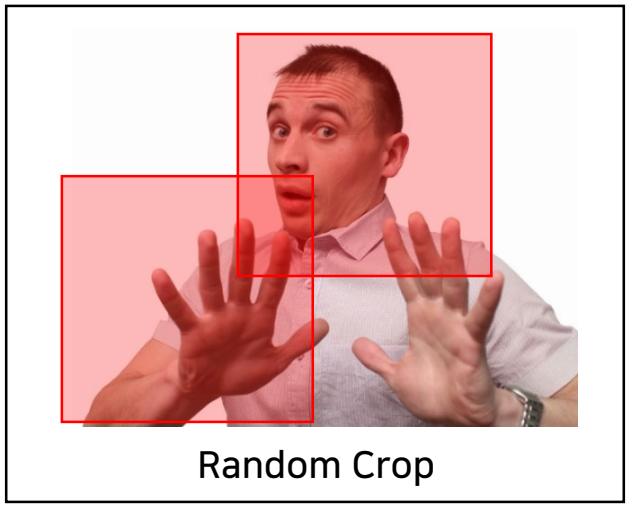
CIFAR-10 다운로드 링크: https://www.cs.toronto.edu/~kriz/cifar.html

CNN – Data Augmentation

PIAI Research Department

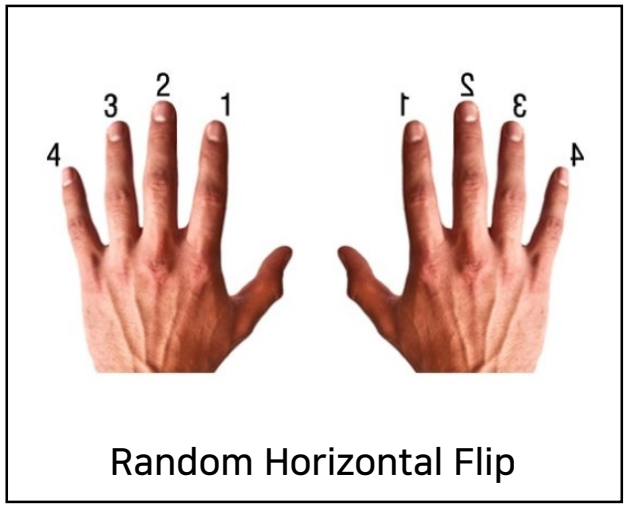
- **데이터 증강 기법**

```
# Load dataset into python variable
input_transform = transforms.Compose([
  Resize(56),
  RandomCrop(56, padding=6),
  RandomHorizontalFlip(),
  ToTensor(),
])
```



Random Crop

출처: https://bonlivre.tistory.com/833



Random Horizontal Flip

출처: https://part2-what-do-we-want.tistory.com/11

- 한정된 데이터로 더 정확도 높은 예측을 하기 위한 데이터 증강 기법

- Random Crop
- Random Horizontal Flip
- Image Rotation
- Brightness
- Contrast
- Saturation
- Hue
- Gaussian Noise
- Random Erasing

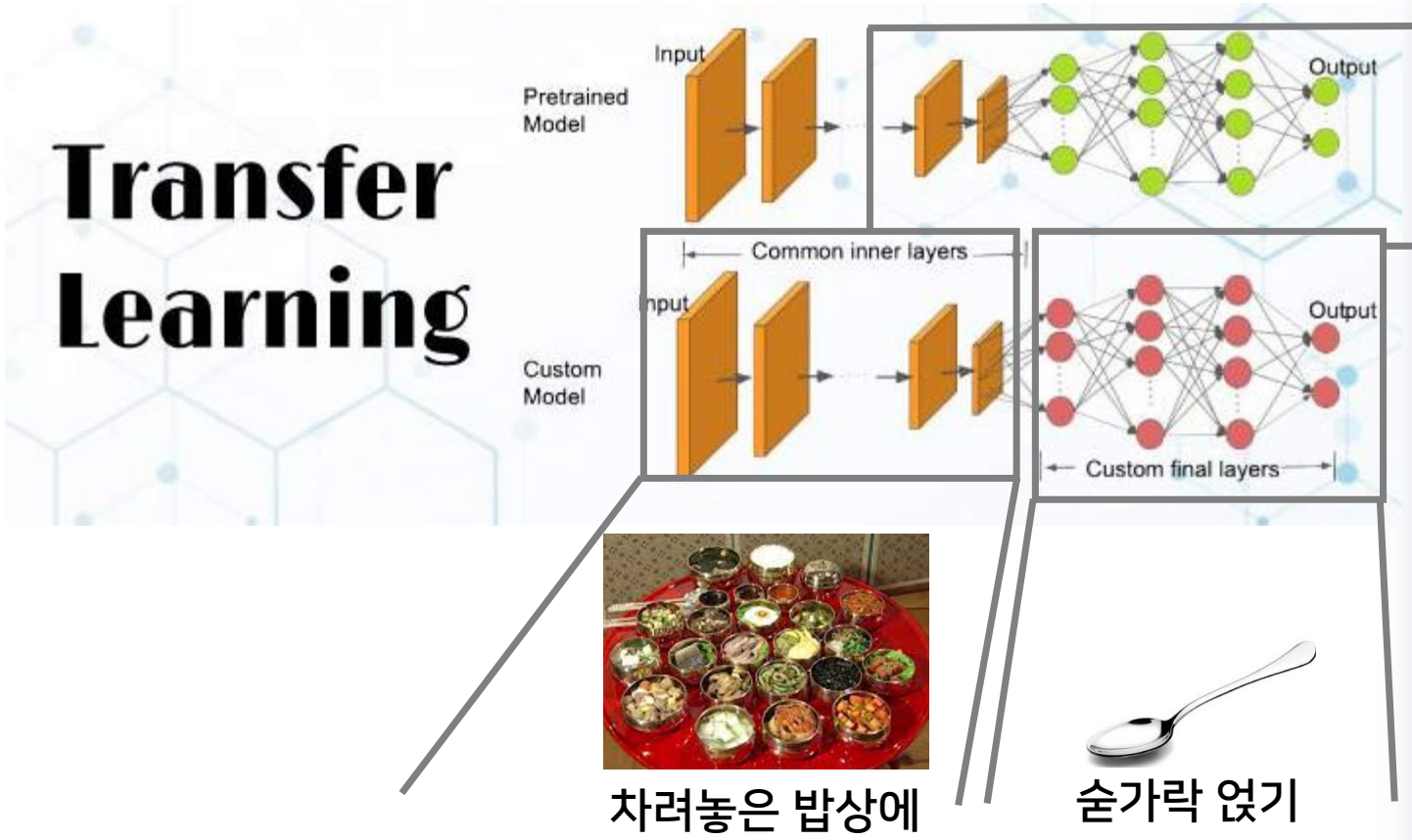
- 코드 사용 안내:
  https://androidkt.com/pytorch-image-augmentation-using-transforms/

PyTorch 공식 홈페이지 명령어 안내: https://pytorch.org/vision/stable/transforms.html

Transfer Learning – Custom Dataset Train

PIAI Research Department

- **Save Time by training only few layers**



- 학습 시간을 효과적으로 줄이기 위하여 이미 학습이 완료된 Pre-Trained 모델을 이용
- Pre-Trained 모델의 끝단 레이어만 학습
- 학습 시 필요한 데이터의 양과 학습 시간을 효율적으로 절약

출처: https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/

Transfer Learning – Custom Dataset Train

PIAI Research Department

- **Save Time by training only few layers**

  - AlexNet
  - ConvNeXt
  - DenseNet
  - EfficientNet
  - EfficientNetV2
  - GoogLeNet
  - Inception V3
  - MNASNet
  - MobileNet V2
  - MobileNet V3
  - RegNet
  - ResNet
  - ResNeXt
  - ShuffleNet V2
  - SqueezeNet
  - SwinTransformer
  - VGG
  - VisionTransformer
  - Wide ResNet

이번에 사용할 모델(ResNet 18)

`resnet18` (*[, weights, progress])	ResNet-18 from Deep Residual Learning for Image Recognition.
`resnet34` (*[, weights, progress])	ResNet-34 from Deep Residual Learning for Image Recognition.
`resnet50` (*[, weights, progress])	ResNet-50 from Deep Residual Learning for Image Recognition.
`resnet101` (*[, weights, progress])	ResNet-101 from Deep Residual Learning for Image Recognition.
`resnet152` (*[, weights, progress])	ResNet-152 from Deep Residual Learning for Image Recognition.

출처: https://pytorch.org/vision/stable/models.html

Transfer Learning – Custom Dataset Train

PIAI Research Department

- **데이터셋 준비 – 원숭이와 사람 구분하기**

원숭이 100장

사람 100장

- 분류하기 원하는 사진을 class 당 개수가 동일하도록 준비
- 데이터셋의 양은 class당 최대 1,000장

How Do You Know You Have Enough Training Data?

2019. 4. 22. — Computer Vision: For image classification using deep learning, a **rule of thumb** is **1,000** images **per class**, where this number can go down ...

Transfer Learning – Custom Dataset Train

PIAI Research Department

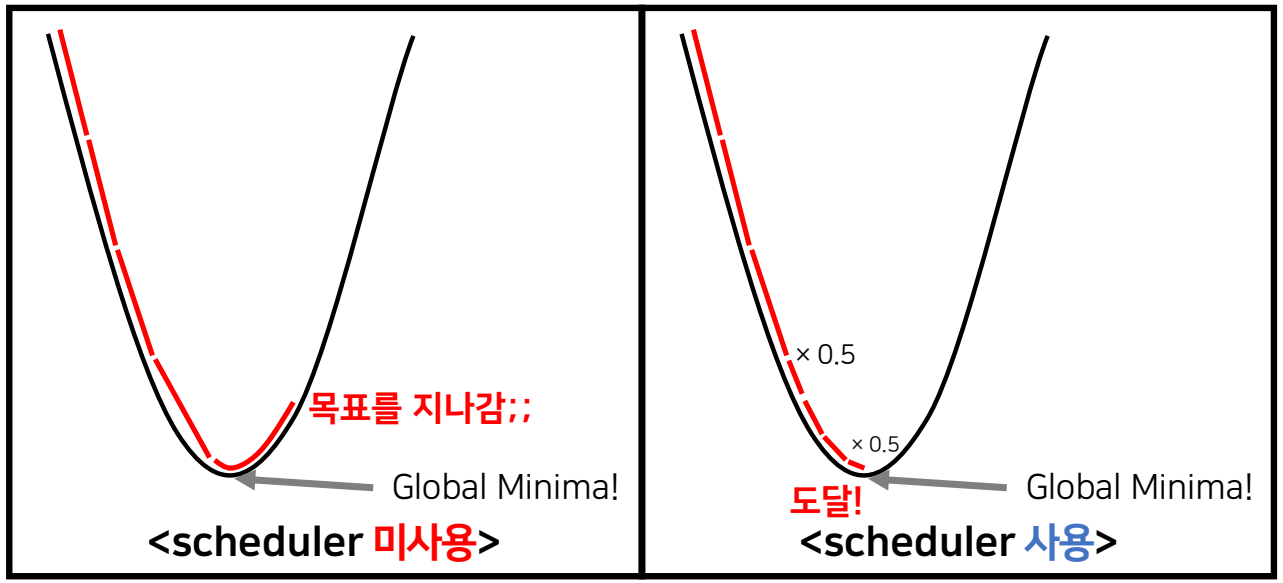
- **Pre-Train Model – ResNet 18 이용**

```
from torchvision import models
model_resnet = models.resnet18(pretrained = True)
```

```
num_ftrs = model_resnet.fc.in_features
model_resnet.fc = nn.Linear(num_ftrs, 2)
net = model_resnet.to(device)
```

```
maximum_epoch = 10
loss_fn = nn.CrossEntropyLoss()
optim = Adam(net.parameters(), lr=0.00001)
from torch.optim import lr_scheduler
lr_scheduler = lr_scheduler.StepLR(optim, step_size = 3, gamma = 0.5)
```

- 모델을 선언하고, pretrained 모드 설정
- 마지막 *FC 레이어의 입, 출력 개수를 변경
- 하이퍼 파라미터 설정
- 학습 추이에 따라 Learning Rate가 줄어들어 학습에 유리한 Schedular 설정



FC레이어: Fully Connected Layer

Transfer Learning – Custom Dataset Train

PIAI Research Department

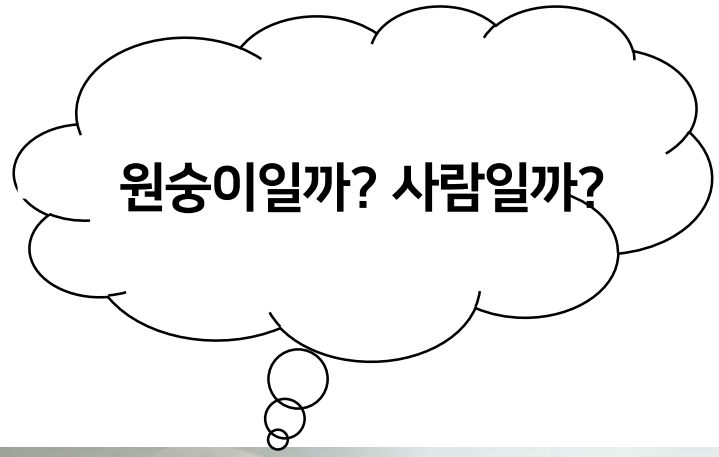
- **Pre-trained 모델 전이 학습 결과**

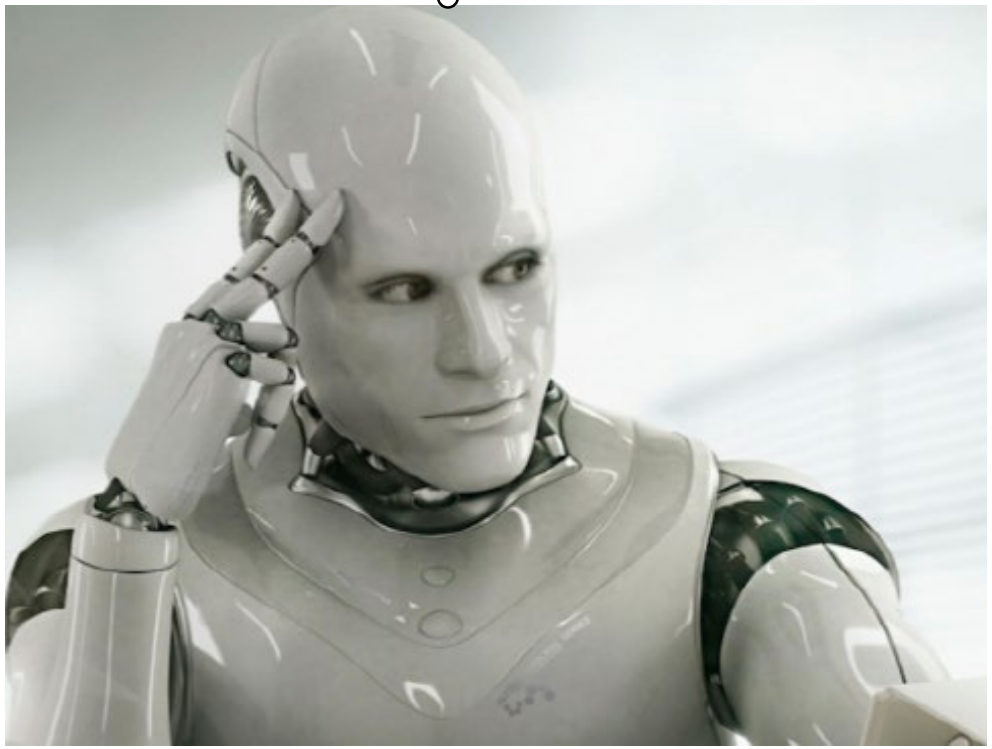

```
Iter:   10 >> T_loss 0.24788    T_acc 0.90714    V_loss 0.06752    V_acc 1.00000    4.119s
Iter:    9 >> T_loss 0.26694    T_acc 0.85714    V_loss 0.05087    V_acc 1.00000    3.900s
Iter:    8 >> T_loss 0.23141    T_acc 0.93571    V_loss 0.04966    V_acc 1.00000    3.961s
Iter:    7 >> T_loss 0.20230    T_acc 0.94286    V_loss 0.05406    V_acc 1.00000    4.053s
Iter:    6 >> T_loss 0.26257    T_acc 0.88571    V_loss 0.06644    V_acc 1.00000    4.012s
Iter:    5 >> T_loss 0.20305    T_acc 0.96429    V_loss 0.07678    V_acc 1.00000    3.990s
Iter:    4 >> T_loss 0.27384    T_acc 0.89286    V_loss 0.09413    V_acc 1.00000    3.985s
Iter:    3 >> T_loss 0.26819    T_acc 0.92857    V_loss 0.12793    V_acc 1.00000    3.947s
Iter:    2 >> T_loss 0.36696    T_acc 0.87857    V_loss 0.19387    V_acc 0.97500    4.015s
Iter:    1 >> T_loss 0.56073    T_acc 0.70000    V_loss 0.41837    V_acc 0.77500    4.255s

Best val Acc: 1.000000
```

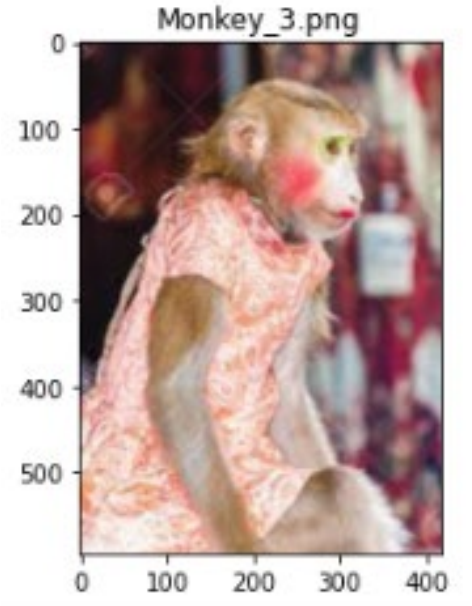
- 3 epoch부터 Validation 분류 정확도 100% 달성
- [Weakly To Do]
  직접 설계한 모델과 Transfer Learning의 성능을 CIFAR-10 데이터셋 기준으로 비교해보자

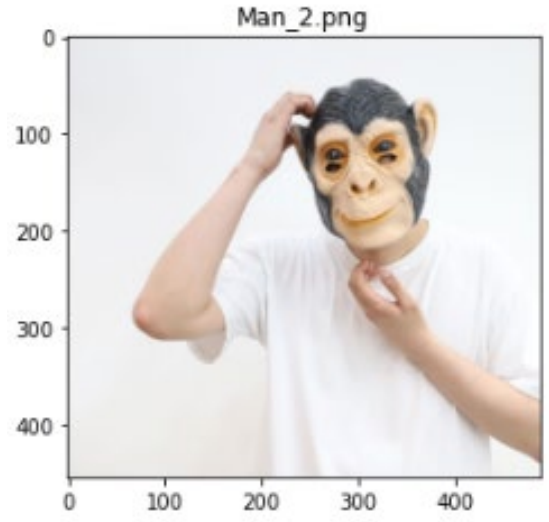
Transfer Learning – Custom Dataset Train

PIAI Research Department





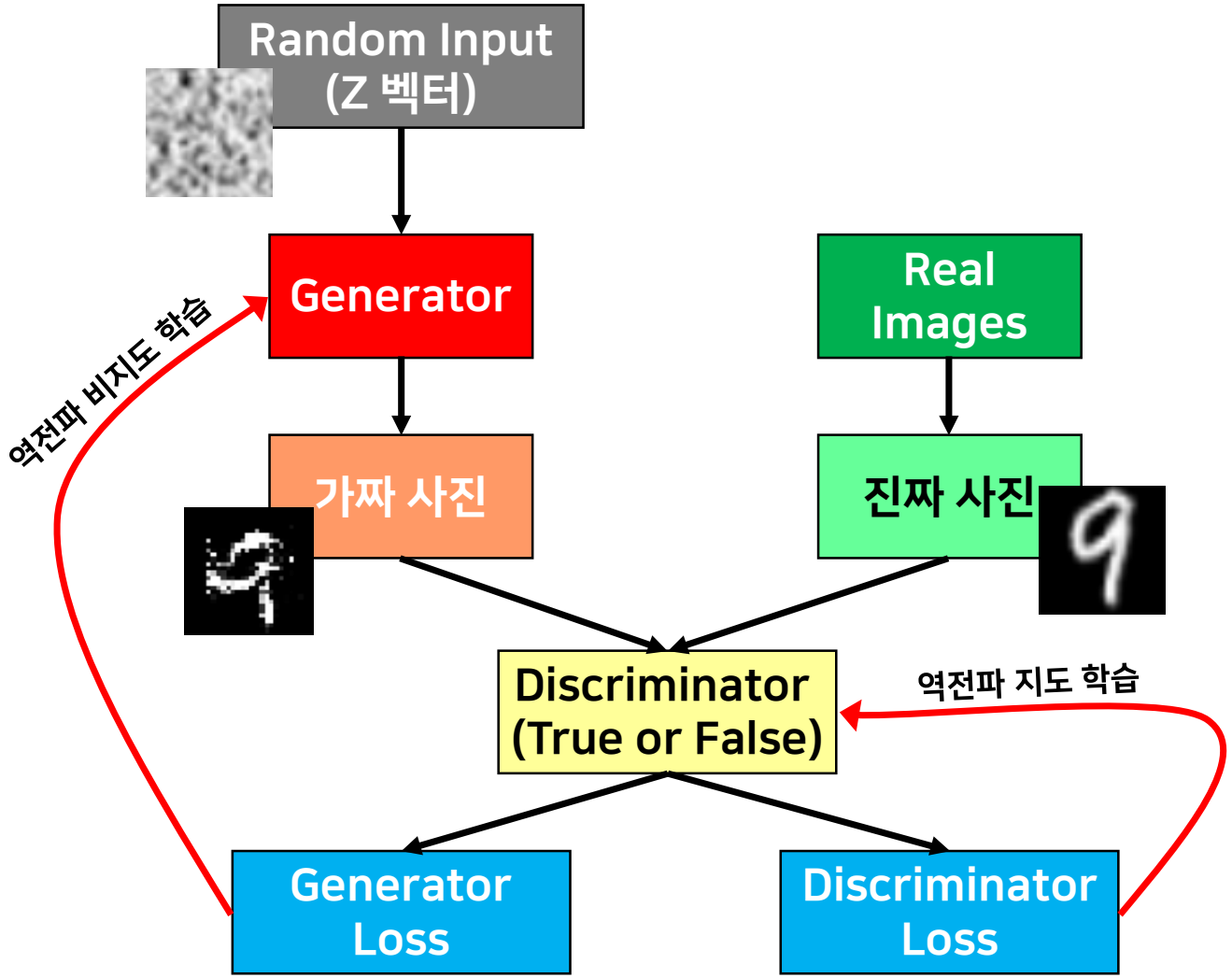




- 원숭이 탈을 쓴 사람과, 사람의 옷을 입은 원숭이를 모델이 제대로 분류할 수 있을까?
- 구분한다면 원리가 무엇일까?

GAN – with MNIST

PIAI Research Department



- GAN(Generative Adversarial Networks)
- 생성적 적대 신경망이라는 의미
- 비지도 학습에 사용되는 머신러닝
- 기존에 없는 새로운 데이터를 생성
- Generator와 Discriminator가 서로 경쟁하며 발전

GAN – with MNIST

- **Generator 아키텍처**

-1 ~ 1 사이로 정규화

```
# MNIST Dataset
transform = transforms.Compose([ToTensor(), Normalize(mean=(0.5,), std=(0.5,))])
```

```
# 진짜같은 가짜 생성기
class Generator(nn.Module):
    def __init__(self, g_input_dim, g_output_dim):
        super(Generator, self).__init__()
        self.fc1 = nn.Linear(g_input_dim, 256)
        self.fc2 = nn.Linear(self.fc1.out_features, self.fc1.out_features*2)
        self.fc3 = nn.Linear(self.fc2.out_features, self.fc2.out_features*2)
        self.fc4 = nn.Linear(self.fc3.out_features, g_output_dim)

    # forward method
    def forward(self, x):
        x = F.leaky_relu(self.fc1(x), 0.2)
        x = F.leaky_relu(self.fc2(x), 0.2)
        x = F.leaky_relu(self.fc3(x), 0.2)
        return torch.tanh(self.fc4(x))
```

- 가짜 데이터를 생성하는 Generator 아키텍처 설계
- 입력으로 난수 입력
- 출력으로 28 * 28 이미지 생성
- -1 ~ 1 사이의 값으로 출력하기 위하여 Tanh Activation Function 사용

GAN – with MNIST

- **Generator 학습 알고리즘**

```
z_dim = 100
mnist_dim = train_dataset.train_data.size(1) * train_dataset.train_data.size(2)

G = Generator(g_input_dim = z_dim, g_output_dim = mnist_dim).to(device)
D = Discriminator(mnist_dim).to(device)
```

```
def G_train(x):

    G.zero_grad()

    z = torch.randn(batch_size, z_dim).to(device)
    y = torch.ones(batch_size, 1).to(device)

    G_output = G(z)
    D_output = D(G_output)
    G_loss = criterion(D_output, y)

    G_loss.backward()
    G_optimizer.step()

    return G_loss.data.item()
```

Binary Cross Entropy Loss

- G를 Generator 모델로 선언
- 난수로 가짜 이미지 생성
- 가짜 이미지를 Discriminator 투입
- Discriminator의 예측 값과 1 사이의 loss 산정
- 계산된 loss로 Generator 학습
- 결국 진짜에 가까운 이미지를 만들도록 학습됨

GAN – with MNIST

- **Discriminator 아키텍처**

```
class Discriminator(nn.Module):
    def __init__(self, d_input_dim):
        super(Discriminator, self).__init__()
        self.fc1 = nn.Linear(d_input_dim, 1024)
        self.fc2 = nn.Linear(self.fc1.out_features, self.fc1.out_features//2)
        self.fc3 = nn.Linear(self.fc2.out_features, self.fc2.out_features//2)
        self.fc4 = nn.Linear(self.fc3.out_features, 1)

    # forward method
    def forward(self, x):
        x = F.leaky_relu(self.fc1(x), 0.2)
        x = F.dropout(x, 0.3)
        x = F.leaky_relu(self.fc2(x), 0.2)
        x = F.dropout(x, 0.3)
        x = F.leaky_relu(self.fc3(x), 0.2)
        x = F.dropout(x, 0.3)
        return torch.sigmoid(self.fc4(x))
```

진짜일 확률값을 출력

- 진짜와 가짜를 판별하는 Discriminator 아키텍처 설계
- 입력 28×28 이미지
- 출력으로 1 or 0 (진짜 or 가짜)

GAN – with MNIST

PIAI Research Department

- **Discriminator 학습 알고리즘**

```
def D_train(x):

    D.zero_grad()

    x_real = x.view(-1, mnist_dim)
    y_real = torch.ones(batch_size, 1)

    x_real, y_real = x_real.to(device), y_real.to(device)

    D_output = D(x_real)
    D_real_loss = criterion(D_output, y_real)
    D_real_score = D_output

    z = torch.randn(batch_size, z_dim).to(device)
    x_fake, y_fake = G(z), torch.zeros(batch_size, 1).to(device)

    D_output = D(x_fake)
    D_fake_loss = criterion(D_output, y_fake)

    D_loss = D_real_loss + D_fake_loss
    D_loss.backward()
    D_optimizer.step()

    return  D_loss.data.item()
```

- 실제 데이터셋 로드
- 진짜를 구별하도록 Loss 산정
- G로 가짜 데이터 생성
- 가짜를 구별하도록 Loss 산정
- 진짜와 가짜를 모두 구별하도록 학습

GAN – with MNIST

- **GAN (G + D) 학습 알고리즘**

```
for epoch in range(1, n_epoch+1):
    D_losses, G_losses = [], []
    for batch_idx, (x, _) in enumerate(train_loader):
        D_losses.append(D_train(x))
        G_losses.append(G_train(x))
```

- G와 D를 번갈아 가며 학습

GAN – with MNIST

PIAI Research Department

1 epochs　　100 epochs　　200 epochs

- MNIST 데이터를 학습하여 세상에 없던 손 글씨를 생성
- MNIST 데이터 증강이 가능할지도..?

n epochs

출처: https://debuggercafe.com/generating-mnist-digit-images-using-vanilla-gan-with-pytorch/

Code Running

3. Pytorch CNN – Additional Functions