

GANOMALY

2조

김영진, 조동현, 정명훈, 임재성

GANOMALY

”

1. 데이터셋 소개
2. **Ganomaly** 소개
3. **Anomalib** 소개

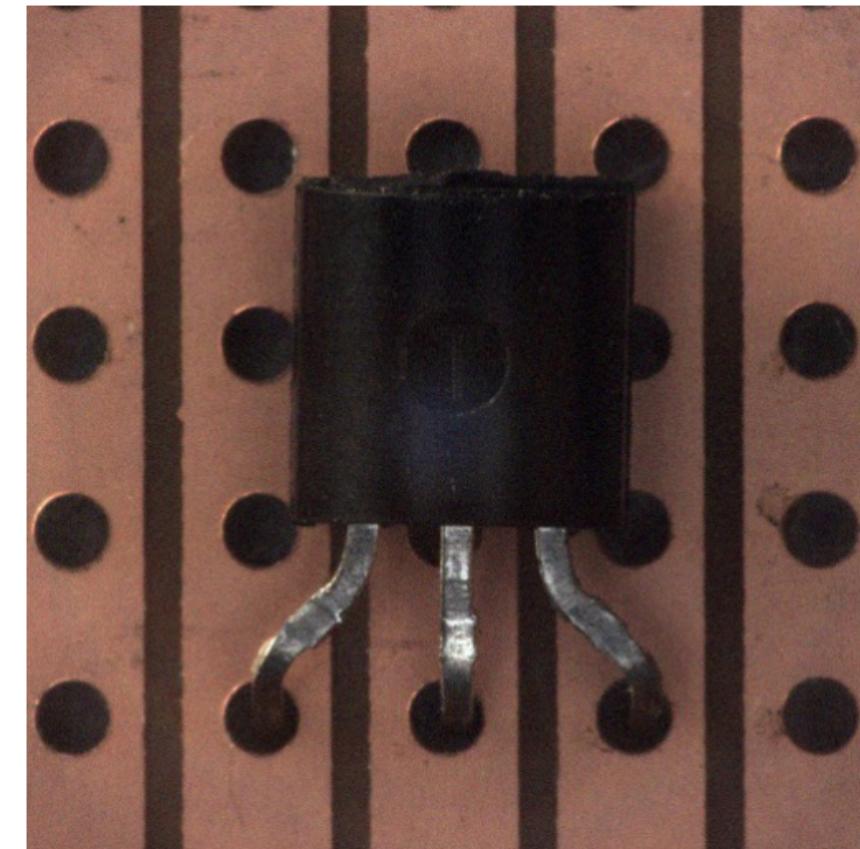
데이터 탐색

데이콘 “반도체 소자 이상 탐지”에서
반도체 이미지 데이터를 사용

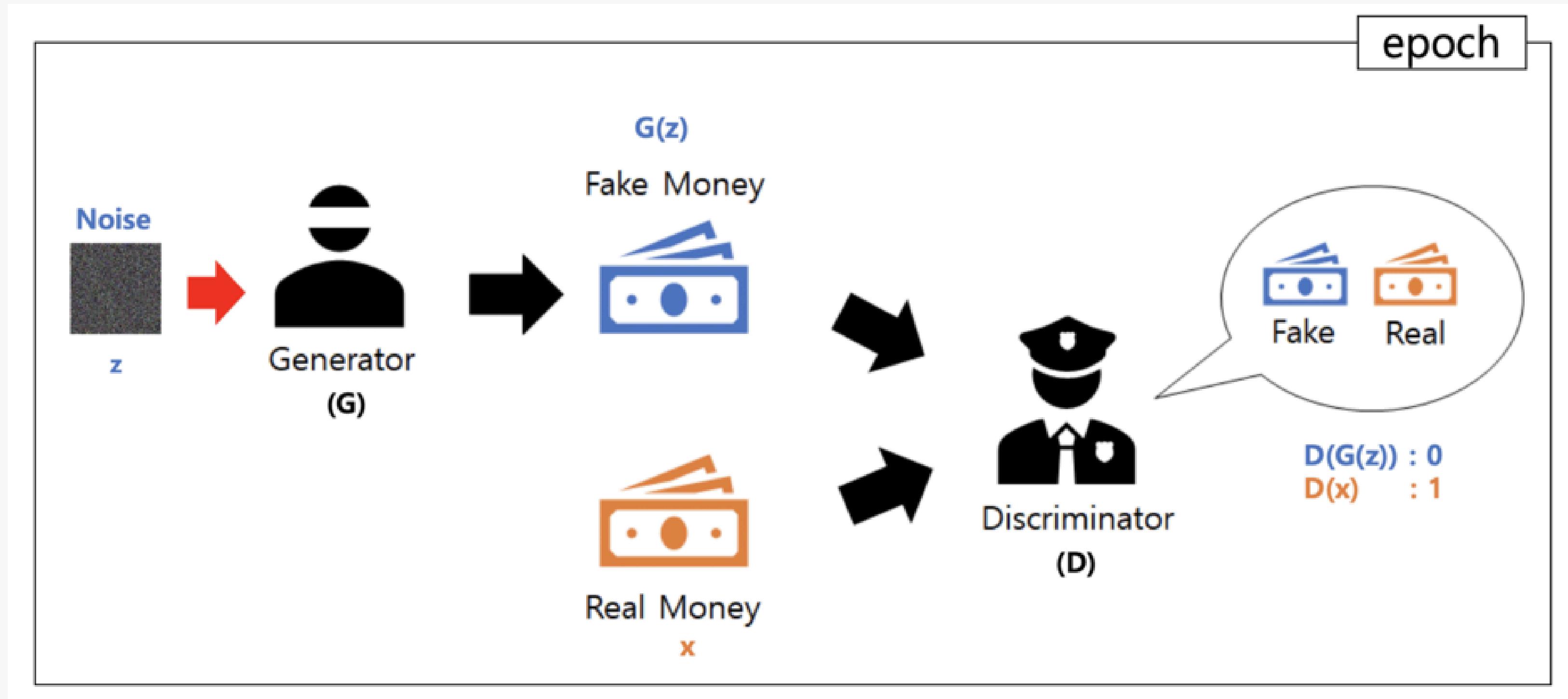
훈련 데이터는 양품 데이터 213장.
테스트 데이터는 불량을 포함한 이미지 100장이 있다.

정상 데이터 213장을 통해 모델을 학습 시키고, 불량 데이터를 감지하는 모델을 만들어야 한다.

TRAIN_000.png



GAN : SIMPLE EXPLAIN



GAN, GENERATIVE ADVERSARIAL NETWORK

- 핵심 아이디어
 - 두 개의 신경망, **GENERATOR**(생성자)와 **DISCRIMINATOR**(판별자)가 경쟁하며 학습.
 - 생성자는 데이터 생성, 판별자는 실제 데이터와 생성된 데이터 구별 역할 수행.

- 구조 및 학습 과정

1. **GENERATOR**(생성자)

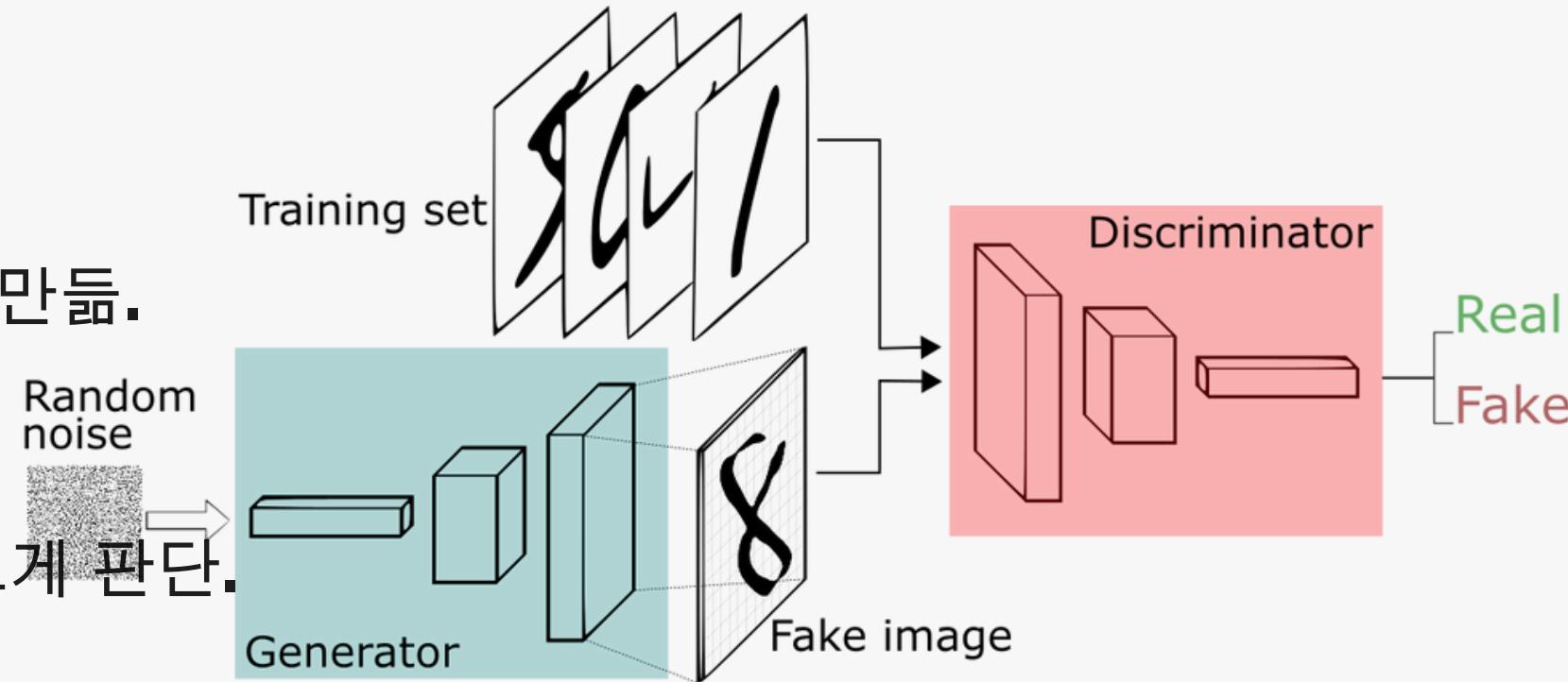
- 랜덤 노이즈를 입력받아 실제 데이터와 유사한 샘플 생성.
- 목표: 판별자가 생성 데이터를 실제 데이터로 판단하도록 만듦.

2. **DISCRIMINATOR**(판별자)

- 실제 데이터와 생성 데이터를 구별하는 이진 분류기 역할.
- 목표: 생성 데이터를 가짜로, 실제 데이터를 진짜로 올바르게 판단.

3. 경쟁적 학습

- **GENERATOR**는 판별자를 속이기 위해 점점 더 정교한 데이터를 생성.
- **DISCRIMINATOR**는 속지 않기 위해 점점 더 정확하게 구분.
- 두 네트워크가 경쟁하며 동시에 성능이 향상.



SEMI-SUPERVISED LEARNING

1. 첫 번째 단점: Labeling Cost

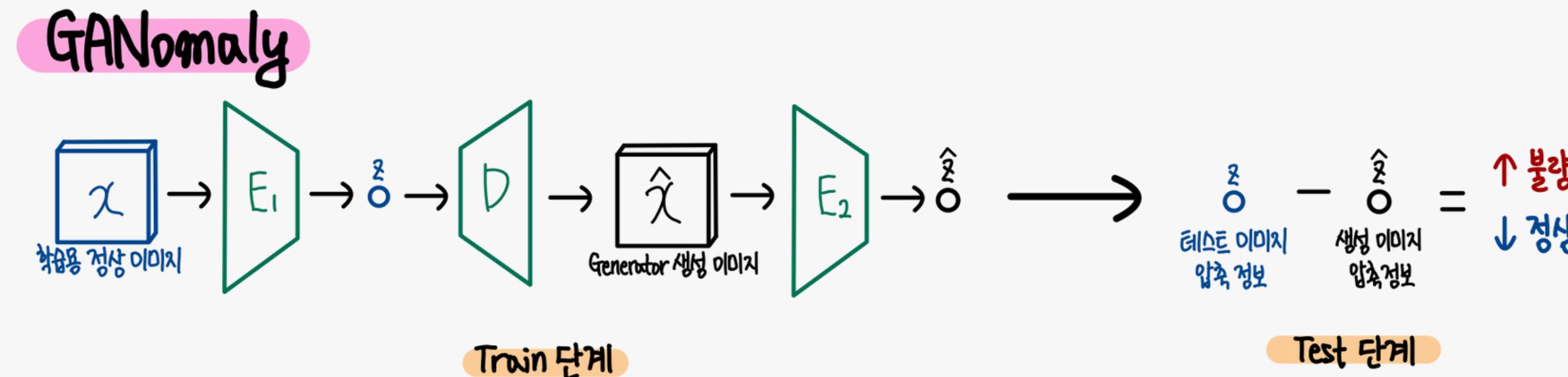
- Labeling 작업의 비용 문제
 - Supervised Learning 방식을 사용하려면 Labeling Dataset이 필요합니다.
 - Labeling 비용은 데이터셋의 분야에 따라 큰 차이가 존재합니다.
 - Anomaly Detection의 특성상 스마트 팩토리, 의료 환경 등 전문 지식이 필요한 경우가 많음.

2. 두 번째 단점: 새로운 불량 검출 불가

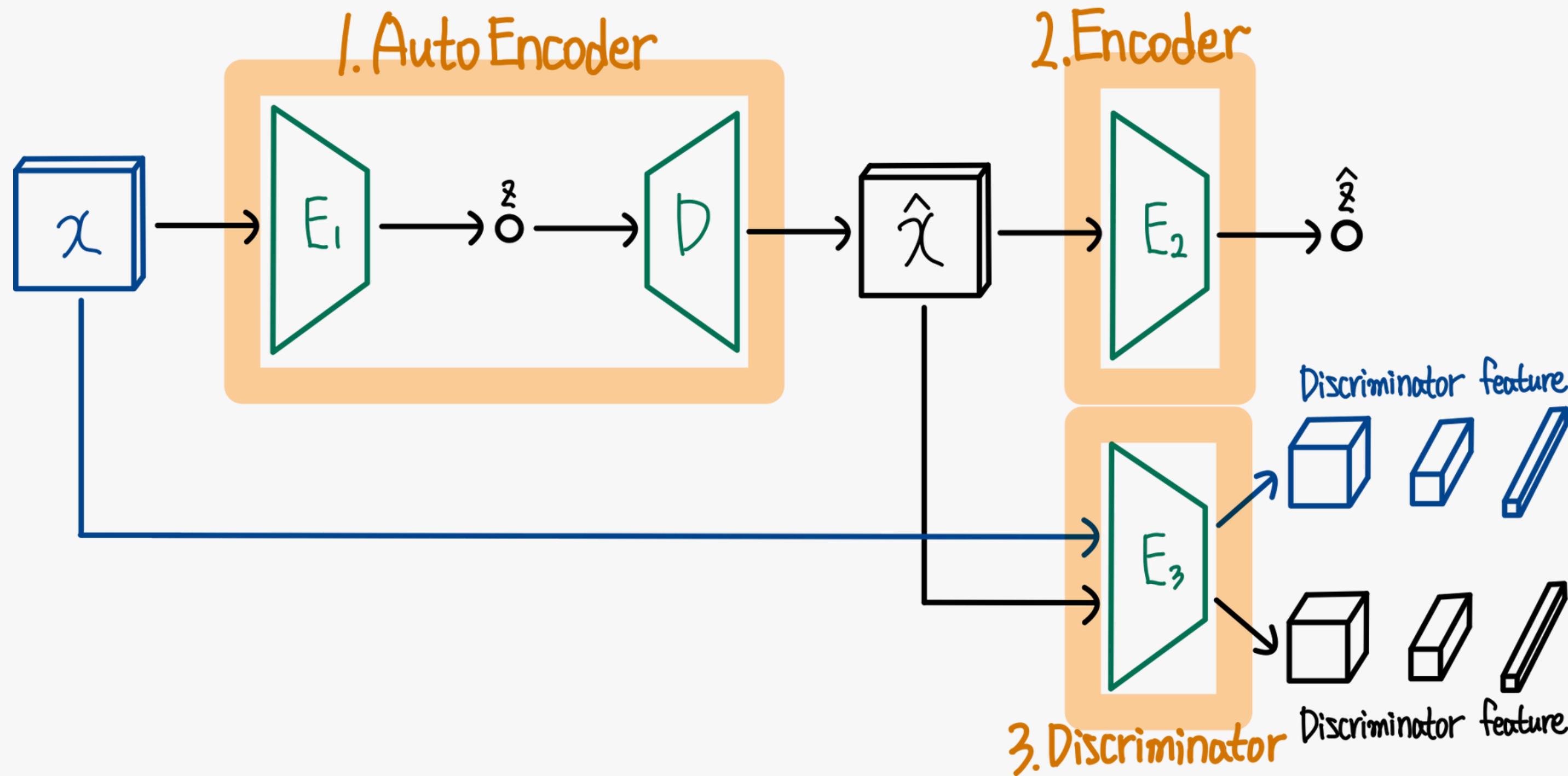
- 데이터셋 외 불량 검출 한계
 - Supervised Learning 방식은 주어진 불량 데이터를 학습합니다.
 - → “이런 특성을 가진 데이터만 불량이다”라고 모델이 인식.

GANOMALY : PROCESS

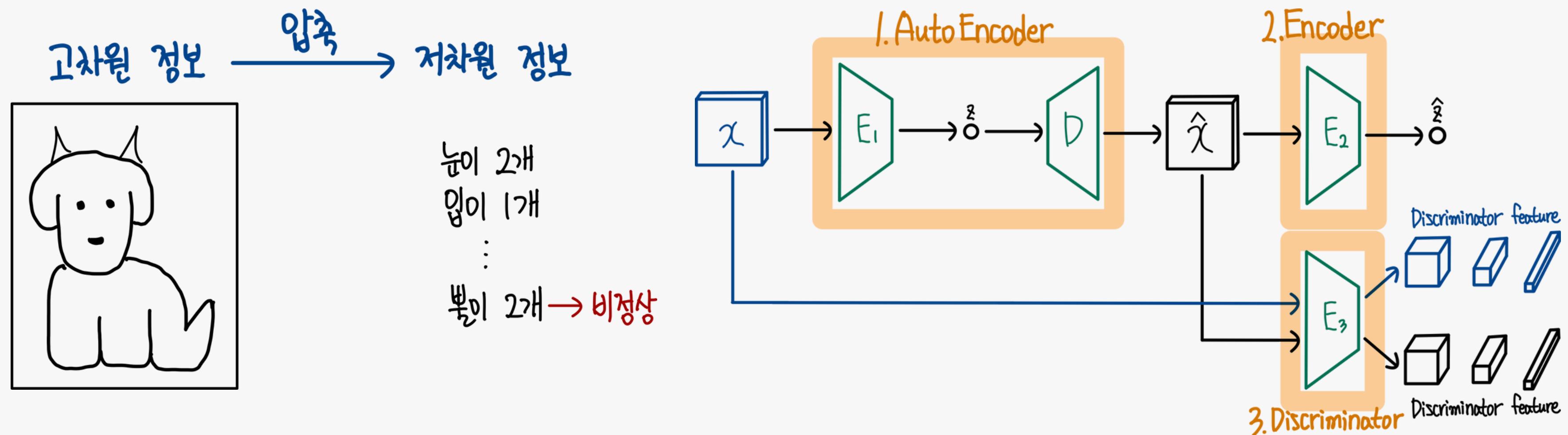
- 1단계로 ANOMALY SCORE 측정: GANOMALY는 ANOGAN의 2단계 학습 문제를 해결하여, 1단계에서 바로 ANOMALY SCORE를 계산할 수 있도록 구조를 변경함.
- $X \rightarrow Z, Z \rightarrow X$ 직렬 연결: ANOGAN은 이미지를 잠재 공간으로 변환 후 다시 원래 이미지로 변환하는 두 단계를 따랐으나, GANOMALY는 이를 직렬로 연결하여 한 번에 학습.
- 한번에 학습, 바로 ANOMALY SCORE 계산: GANOMALY는 두 단계를 나누지 않고 한 번에 학습하며, 즉시 ANOMALY SCORE를 계산할 수 있음.



GANOMALY : STRUCTURE

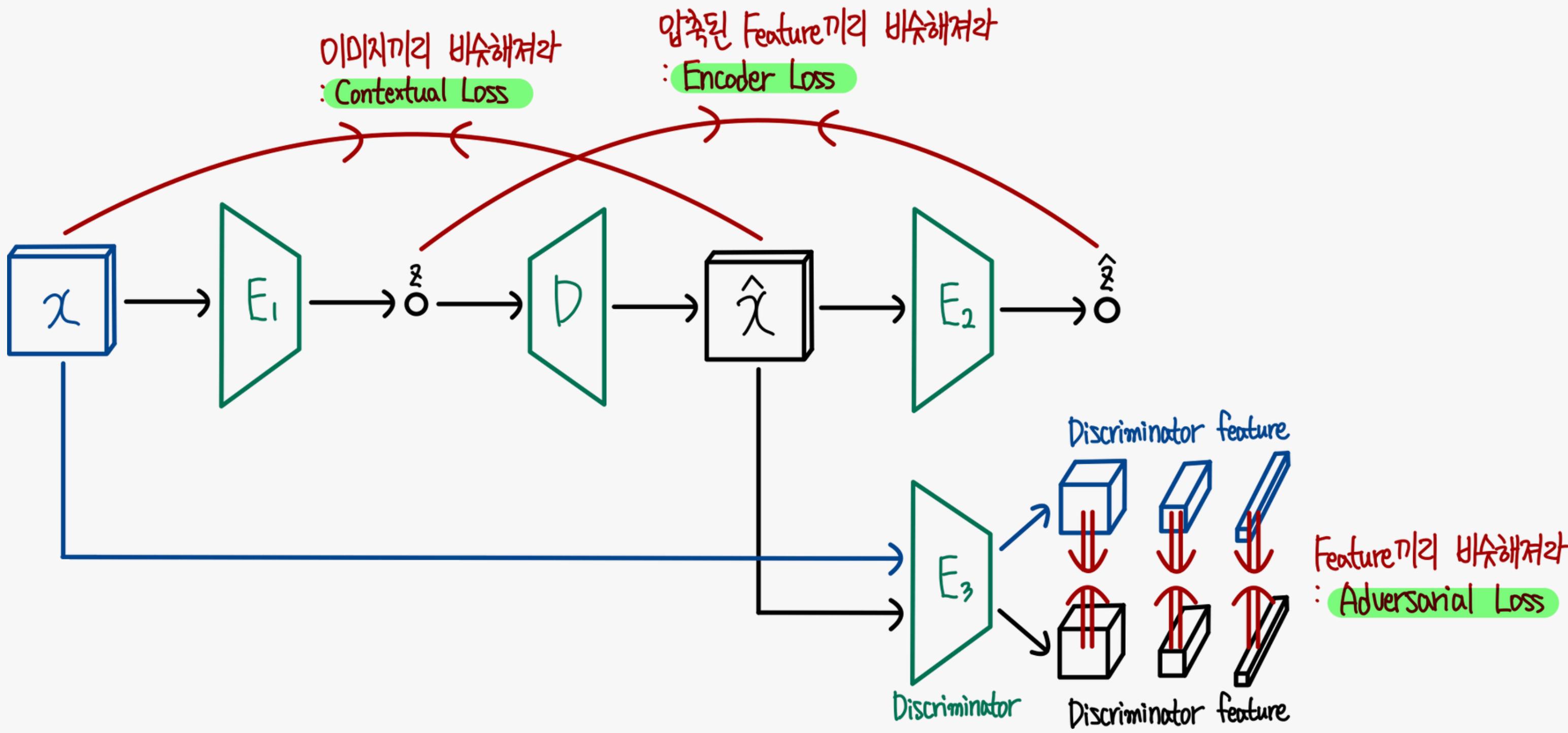


GANOMALY : WHY ENCODE?



1. 압축된 정보를 입력 받으면 판단해야 하는 정보의 개수 자체가 적어진다(차원축소).
2. 자잘한 **NOISE** 들은 하나의 큰 정보로 압축되어 **NOISE**는 큰 영향을 주지 못한다.

GANOMALY : LOSS FUNCTION(FOR LEARNING)



$$\mathcal{L} = w_{adv}\mathcal{L}_{adv} + w_{con}\mathcal{L}_{con} + w_{enc}\mathcal{L}_{enc}$$

ANOMAL SCORE (FOR DETECTION)

시그모이드 함수를 이용해서 **0~1** 값으로 **NORMALIZATION** 하면
최종 **ANOMAL SCORE**가 나오게 됩니다.

$$A(\hat{x}) = \left\| G_E(\hat{x}) - E(G(\hat{x})) \right\|_1$$

해스트 이미지의 압축한 feature와 생성한 이미지의 압축한 feature의 차이

GANOMALY CODE

```
▶ from google.colab import drive  
drive.mount('/content/drive')  
→ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

구글 드라이브 연동

```
▶ import os  
import numpy as np  
import pandas as pd  
import torch  
import torch.nn as nn  
from torch.utils.data import Dataset, DataLoader  
from torchvision import transforms  
from PIL import Image  
from tqdm import tqdm
```

필요한 라이브러리 import

GANOMALY CODE

```
▶ class SemiconductorDataset(Dataset):
    def __init__(self, folder_path, transform=None):
        self.folder_path = folder_path
        self.file_names = sorted(os.listdir(folder_path))
        self.transform = transform

    def __len__(self):
        return len(self.file_names)

    def __getitem__(self, idx):
        img_path = os.path.join(self.folder_path, self.file_names[idx])
        image = Image.open(img_path).convert("RGB")
        if self.transform:
            image = self.transform(image)
        return image, 0 # label은 필요하지 않음

▶ # 데이터 경로 설정
train_folder = "/content/drive/MyDrive/open/train"
test_folder = "/content/drive/MyDrive/open/test"

# 이미지 전처리 파이프라인
image_size = 256
transform = transforms.Compose([
    transforms.Resize((image_size, image_size)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]) # [-1, 1] 정규화
])

# Dataset 및 DataLoader 생성
train_dataset = SemiconductorDataset(train_folder, transform=transform)
test_dataset = SemiconductorDataset(test_folder, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

구글 드라이브 연동

필요한 라이브러리 import

GANOMALY CODE

```
▶ class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1),
            nn.Tanh()
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

Generator 정의

```
▶ class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2),
            nn.Flatten(),
            nn.Linear(256 * (image_size // 8) * (image_size // 8), 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.model(x)
```

Discriminator 정의

GANOMALY CODE

```
▶ device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
  generator = Generator()
  discriminator = Discriminator()

  trained_generator = train_ganomaly(generator, discriminator, train_loader, num_epochs=50, lr=0.0002, device=device)
```

테스트 데이터 훈련

```
▶ def detect_anomalies(generator, test_loader, threshold, device):
  generator = generator.to(device)
  generator.eval()

  anomaly_scores = []
  with torch.no_grad():
    for batch in tqdm(test_loader, desc="Detecting anomalies"):
      images, _ = batch
      images = images.to(device)

      reconstructed_images = generator(images)
      scores = torch.mean((reconstructed_images - images) ** 2, dim=[1, 2, 3]) # MSE Loss
      anomaly_scores.extend(scores.cpu().numpy())

  predictions = (np.array(anomaly_scores) > threshold).astype(int)
  return anomaly_scores, predictions
```

Anomaly 정의 이후 분류

```
▶ threshold = 0.1
  anomaly_scores, predictions = detect_anomalies(trained_generator, test_loader, threshold, device)

  # 결과 저장
  result_df = pd.DataFrame({
    "ID": [f"TEST_{i:03d}.png" for i in range(len(predictions))],
    "label": predictions
  })
  result_df.to_csv("sample_submission.csv", index=False)
  print("Results saved to sample_submission.csv")
```

GANOMALY CODE

```
▶ class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1),
            nn.Tanh()
        )

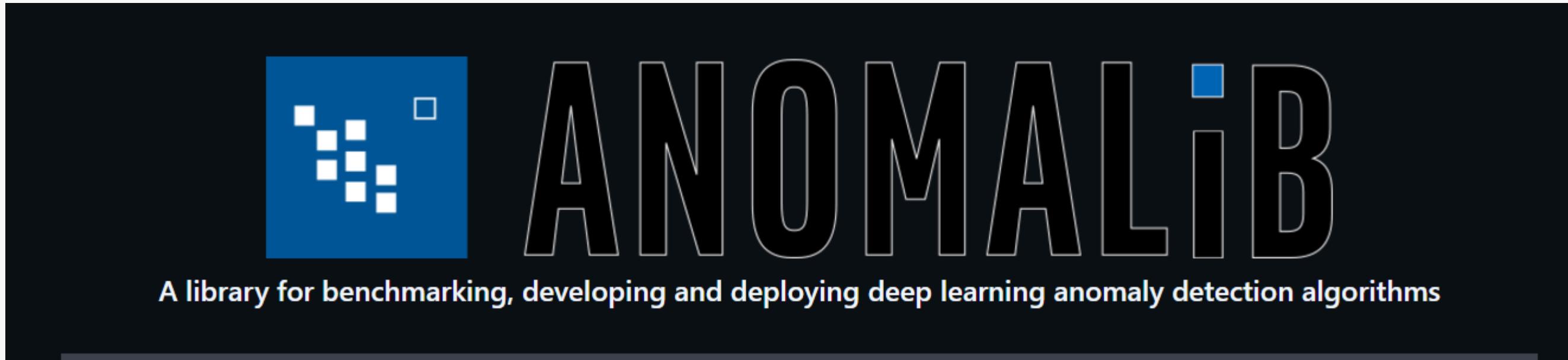
    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

Generator 정의

```
▶ class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2),
            nn.Flatten(),
            nn.Linear(256 * (image_size // 8) * (image_size // 8), 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.model(x)
```

Discriminator 정의



ANOMALIB

딥러닝 기반
Anomaly 탐지

오픈소스 파이썬 라이브러리

정상 이미지를 통해 이상 이미지를 탐지 하는 방법

Pathcore 방식을 주로 사용

아직 개발 단계에 있는 라이브러리이기 때문에 Github
Source code를 직접 이용

ANOMALIB CODE

```
import os
from anomalib.data import get_transforms
from anomalib.models import get_model
from anomalib.utils.callbacks import get_callbacks
from anomalib.utils.inference import Inferencer
from pytorch_lightning import Trainer
from pytorch_lightning.callbacks import EarlyStopping
```

필요한 라이브러리 import

```
train_data_path = "/Users/kim-yeongjin/Desktop/쿠글 2차 프로젝트/open/train"
test_data_path = "/Users/kim-yeongjin/Desktop/쿠글 2차 프로젝트/open/test"
```

데이터 경로 설정

ANOMALIB CODE

```
# GANomaly 모델 설정
model_config = {
    "name": "ganomaly",
    "image_size": 256,
    "latent_dim": 100,
    "batch_size": 32,
    "lr": 0.0002,
    "num_epochs": 50
}
model = get_model(model_config)

# 학습 콜백
callbacks = get_callbacks(
    model_name="ganomaly",
    log_dir="./logs"
)
callbacks.append(EarlyStopping(monitor="val_loss", patience=5, mode="min"))
```

```
# 모델 학습
trainer = Trainer(
    max_epochs=model_config["num_epochs"],
    gpus=1 if torch.cuda.is_available() else 0,
    callbacks=callbacks
)
trainer.fit(model, train_loader)

# 모델 저장
os.makedirs("saved_model", exist_ok=True)
trainer.save_checkpoint("saved_model/ganomaly.ckpt")

# 모델 로드 및 추론
inferencer = Inferencer(model="ganomaly", ckpt_path="saved_model/ganomaly.ckpt")
```

```
# 테스트 이미지 예측
for batch in test_loader:
    images, _ = batch
```

모델 학습 및 추론

GANomaly 모델 설정

THANK YOU

FOR COMING