


# 달리를 찾아라!

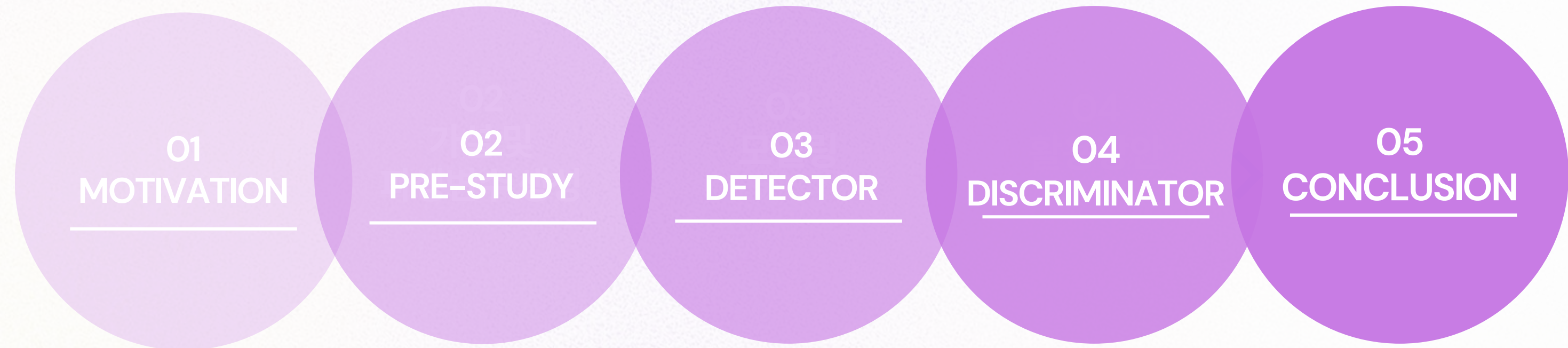
DALL-E가 생성한 Synthetic data로 학습한 Two-Stage Project

 **Dali, Van, Picasso, CV 1팀 Let's go**  
16기 이영노 / 17기 김지윤, 김희준, 문성빈 / 18기 백성은





# CONTENTS





# 01. MOTIVATION

Why Text to Image?

01  
MOTIVATION

---

02  
PRE-STUDY

---

03  
DETECTOR

---

04  
DISCRIMINATOR

---

05  
CONCLUSION

---



# 01-1 Why Text to Image?

공통된 흥미:  
Generative Model

팀원 모두 각자의 이유로  
Generation model을  
다루고 싶었으며,  
generation 중에서도  
task를 정하기 위해  
각자 관련 자료, 논문 조사

ChatGPT의 등장  
+  
Prompt의 중요성

높은 퀄리티의 prompt가  
높은 퀄리티의 generation과  
직결된다는 생각에,  
text to image와 같은  
prompt가 중요한 task를  
from scratch로 다루기로 도전



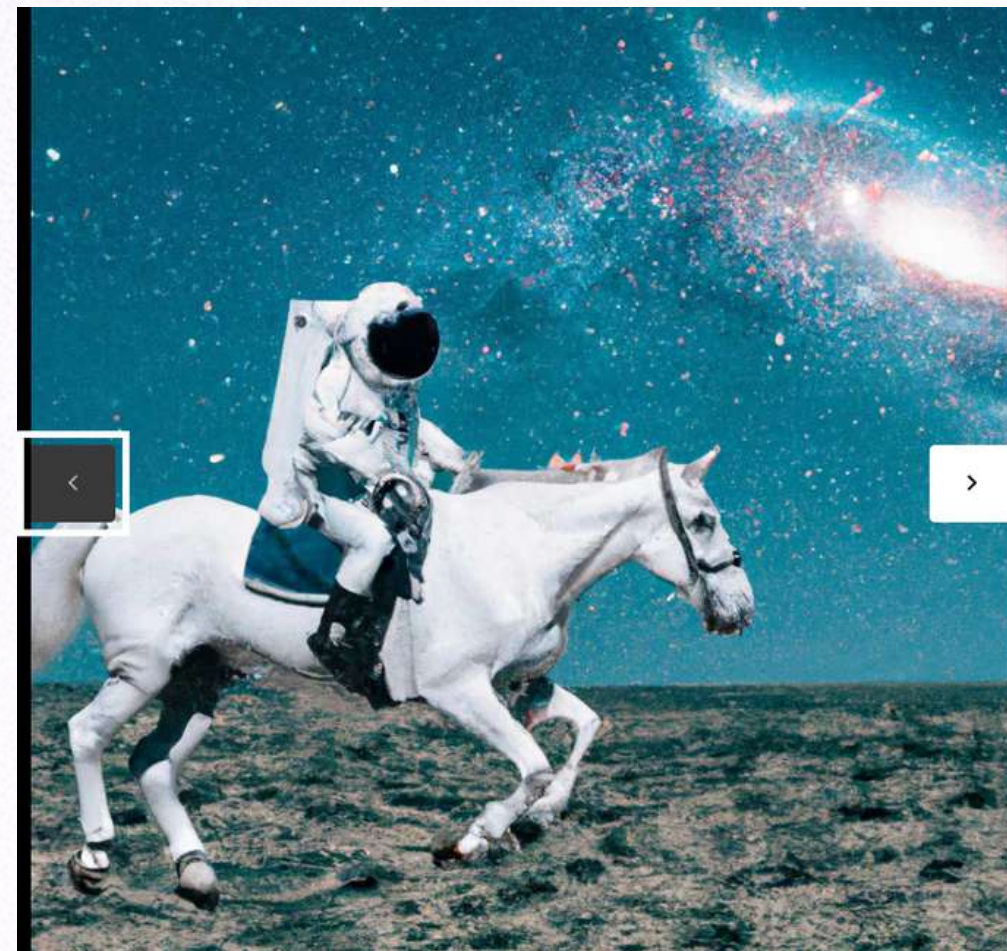
## 01-1 Why Text to Image?



input:

An astronaut riding a horse in photorealistic style.

output:



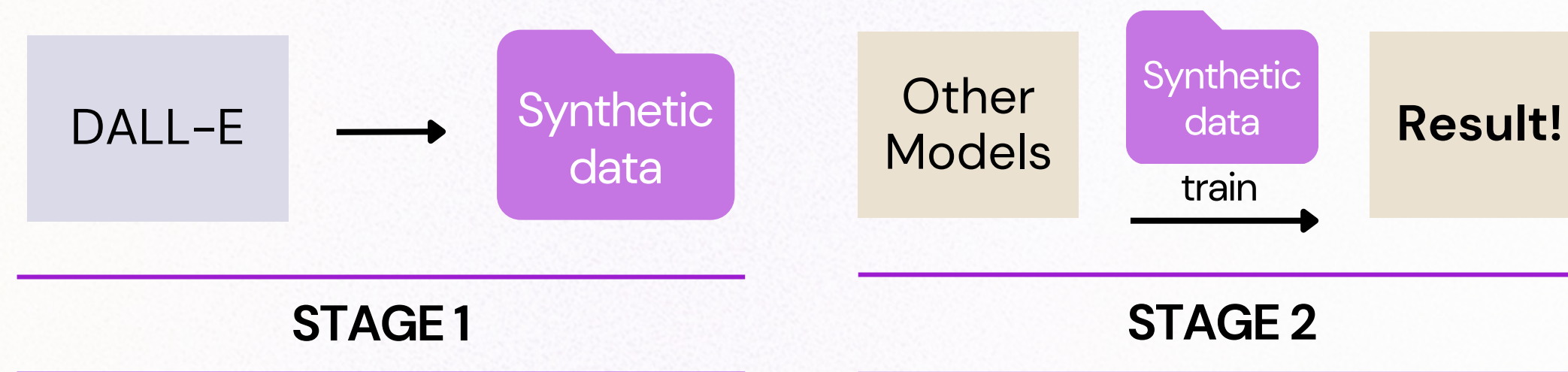
출처: OpenAI

Text-to-Image의 대표적인 모델인 **DALL-E** model 활용



## 01-2 Then, How to make use of DALL-E?

“  
DALL-E로 생성한 이미지를 **Synthetic Data**로 활용해볼까?  
”



1. 적절한 prompt → 양질의 synthetic data 생산
- +  
2. Synthetic data를 활용한 **Computer vision**의 여러 task를 **two-stage**로 시도



# 02.

## PRE-STUDY

About Diffusion, VAE, GAN, and DALL-E

01  
MOTIVATION

---

02  
PRE-STUDY

---

03  
DETECTOR

---

04  
DISCRIMINATOR

---

05  
CONCLUSION

---



## O2-1 Diffusion

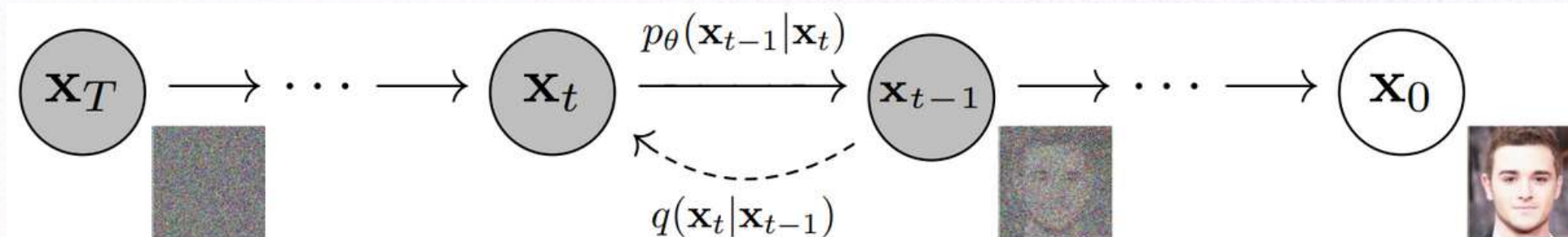


Figure 2: The directed graphical model considered in this work.

## DDPM paper review

- **forward process:** time step마다 noise를 점차 추가하여 최종적으로 이미지를 알아볼 수 없게 destroy하는 과정
- **reverse process:** noise를 점차 제거해가며 원본에 가깝게 생성해내는 과정
- 위 과정들은 variational inference를 통해 훈련된 **Markov chain**을 parameterize하면서 이루어짐

## DDPM code review

**denoising-diffusion-pytorch** Public

Implementation of Denoising Diffusion Probabilistic Model in Pytorch

deep-learning artificial-intelligence generative-model

Python 6,110 789 MIT License Updated on Nov 28

**pytorch-ddpm** Public

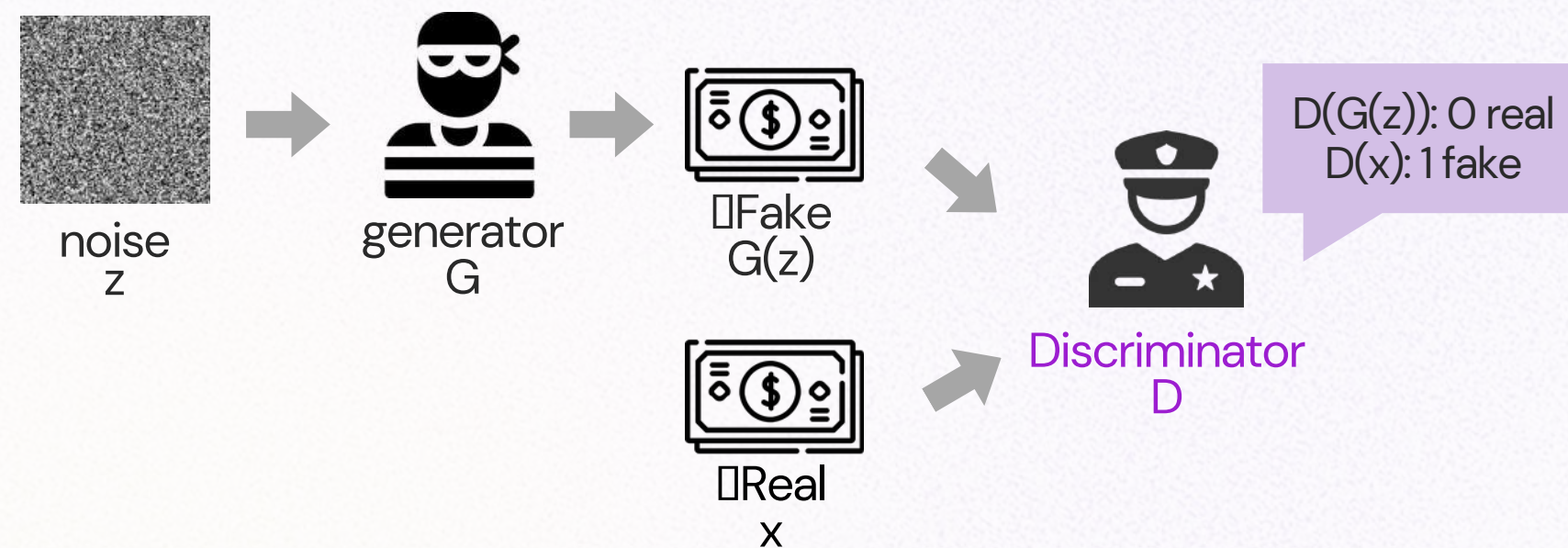
Unofficial PyTorch implementation of Denoising Diffusion Probabilistic Models

reproducible pytorch ddpm

Python 356 55 Do What The F\*ck You Want To Public License Updated on Apr 13



## 02-2 GAN



### GAN paper review

- **adversarial net:** G와 D가 서로 경쟁하며 학습. D가 real로 판별할 만큼 real같은 fake를 생성해내는 G를 만드는 것이 목표
- ddpm과 달리 Markov chain을 활용하지 않음
- back-propagation으로 비교적 간단히 학습 가능

**PyTorch-GAN**

Public

PyTorch implementations of Generative Adversarial Networks.

Python

☆ 15,147

🔗 3,943

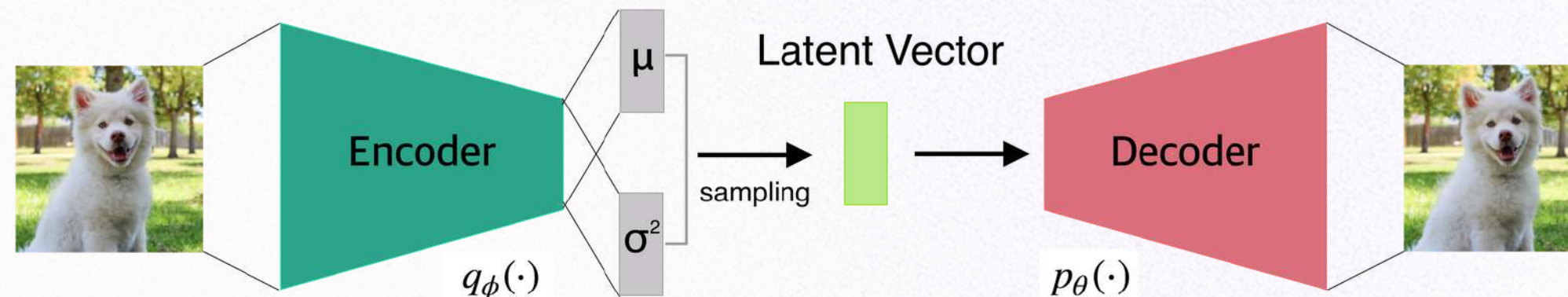
📄 MIT License

Updated on May 24

### GAN code review



## O2-3 VAE



- Obj. = regularizer + expected negative reconstruction error

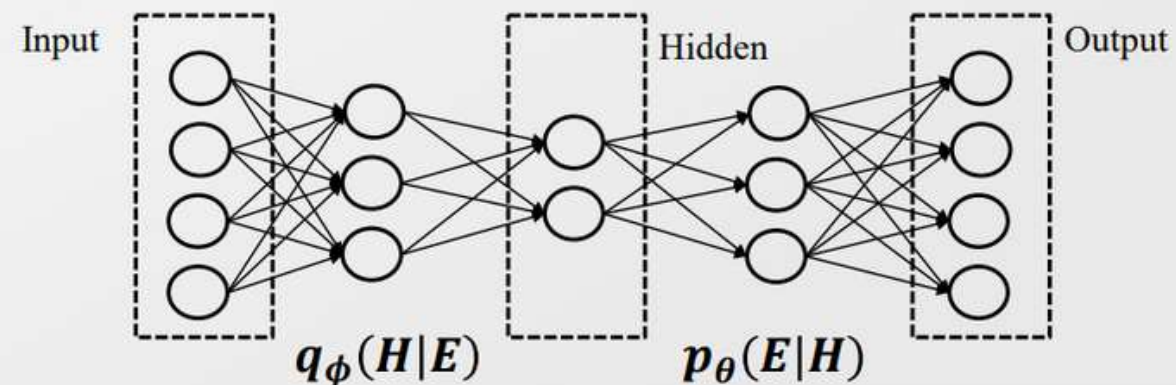
$$\mathcal{L} = -D_{KL}(q_{\phi}(H|E)||p_{\theta}(H)) + \mathbb{E}_{q_{\phi}(H|E)}[\log p_{\theta}(E|H)]$$

Variational distribution  $q_{\phi}: E \Rightarrow H$   
generates **hidden representation**  
 $H$  from data  $E$

Original distribution  $p_{\theta}: H \Rightarrow E$   
generates data  $E$  from latent  
(hidden) variable  $H$

Probabilistic (stochastic) **encoder**

Probabilistic (stochastic) **decoder**



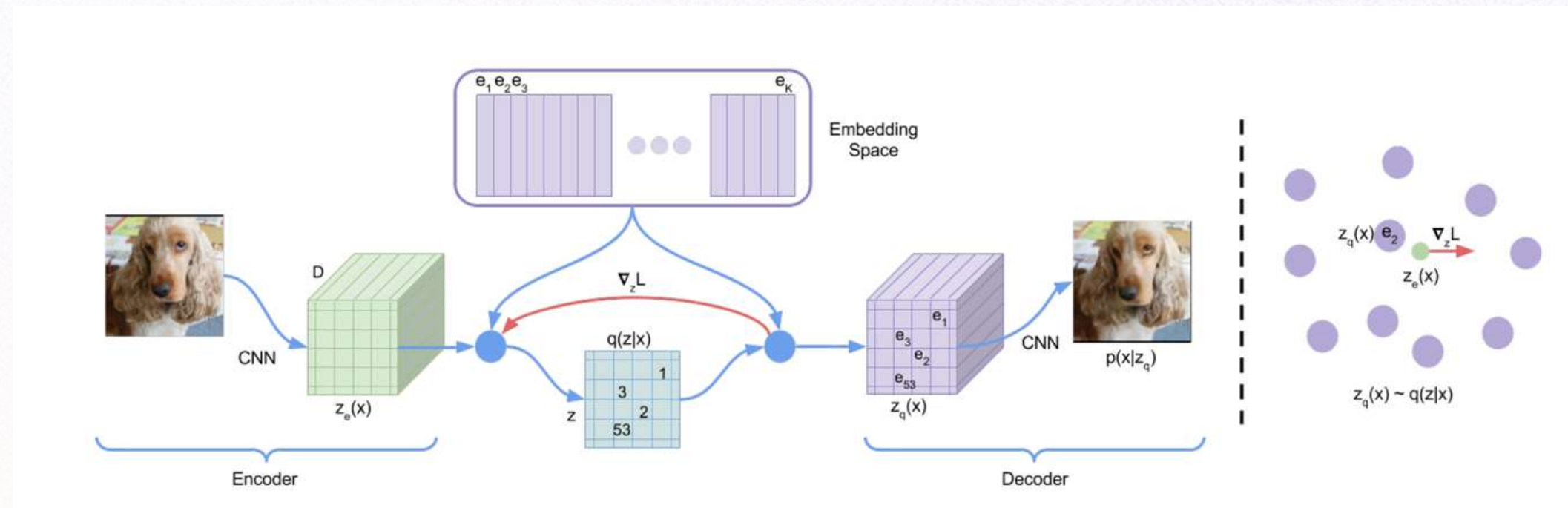
## VAE paper review

- **Variational Inference**: 연산량이 많거나 intractable한 probability density를 구하지 못하는 대신, 근사한 값인 lower bound를 찾을 수 있으면, probability density도 근사하게 구할수 있음.
- Encoder: Variational Distribution  $q$ 를 학습 Evidence (Data) 를 통해 Latent feature (Hidden) 학습.
- $q$ 에서는  $p$ 와 가장 유사한  $q$ 를 학습하기 위해 KL Divergence를 사용함.
- Decoder: Original Distribution  $p$ 를 학습 Latent feature (Hidden) 을 통해 Evidence (Data) 를 reconstruct

## VAE code review



## O2-4 VQ-VAE



### VQ-VAE paper review

- 기존 VAE에 **VQ**(Vector Quantisation)을 사용하여 이미지의 이산표현을 가능하게 함
- encoder가 이미지를 받으면, 이를 codebook에서 가장 유사한 embedding vector로 찾아와 변환함
- 이 때 대체된 vector는 원본과 유사한 이미지로 reconstruction
- 이러한 VQ방식은 **DALL-E의 key idea**로 쓰임

### vq-vae-2-pytorch Public

Implementation of Generating Diverse High-Fidelity Images with VQ-VAE-2 in PyTorch

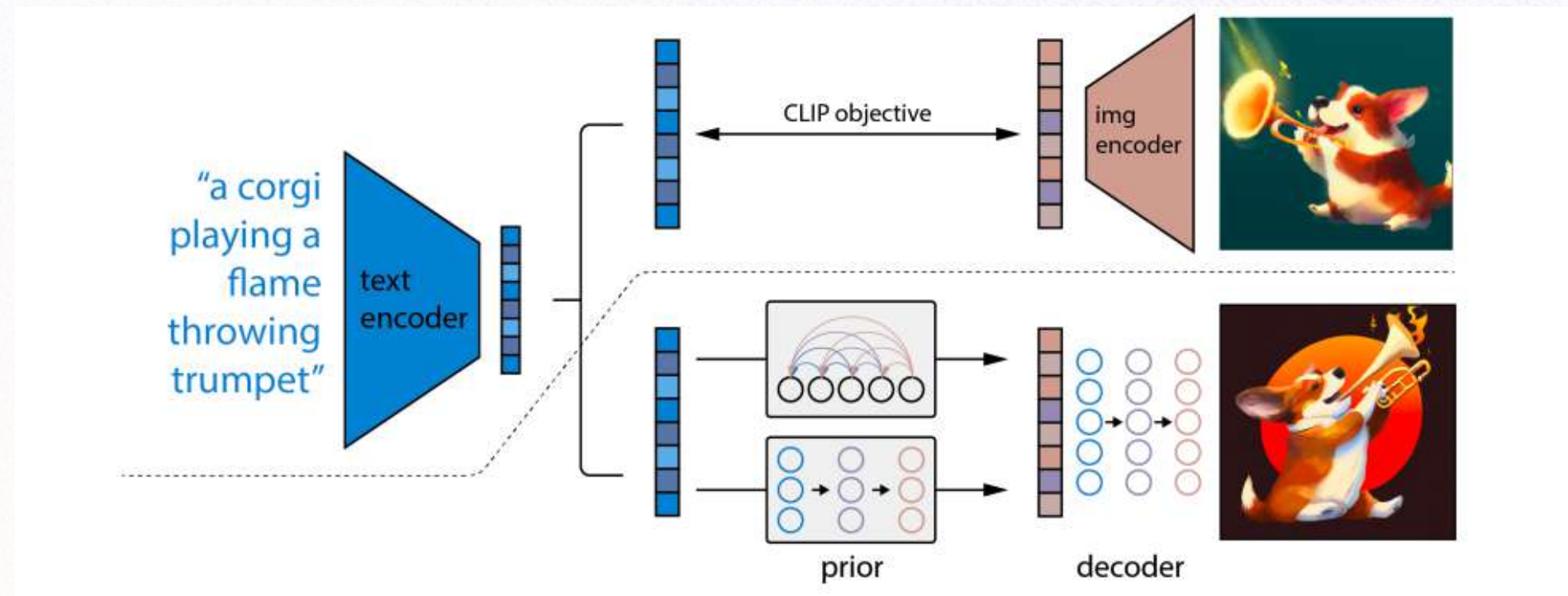
[vq-vae](#) [vq-vae-2](#)

Python 1,410 270 Other Updated on Feb 15, 2023

### VQ-VAE code review



## O2-5 DALL-E



### DALL-E paper review

- training label 없이 **zero shot**으로 뛰어난 성과.
- 120억 개의 파라미터를 가진 **transformer**를 학습.
- **Stage1:** 이미지를 픽셀이 아닌 **토큰**(using dVAE)으로 입력. VQ-VAE의 code book 방식 차용. transformer를 freeze한 채로 encoder, decoder만 학습.
- **Stage2:** 이미지 토큰과 텍스트 토큰(using BPE)을 결합하여 ELBO를 최대화하는 방식으로 학습. encoder, decoder를 freeze하고 transformer만 학습.

### DALLE2-pytorch Public

Implementation of DALL-E 2, OpenAI's updated text-to-image synthesis neural network, in Pytorch

[deep-learning](#) [artificial-intelligence](#) [text-to-image](#)

[Python](#) [10,481](#) [1,021](#) [MIT License](#) Updated on Oct 19

### DALL-E code review



# 03. DETECTOR

Let's Detect Object in Dali's Painting!

01  
MOTIVATION

---

02  
PRE-STUDY

---

03  
DETECTOR

---

04  
DISCRIMINATOR

---

05  
CONCLUSION

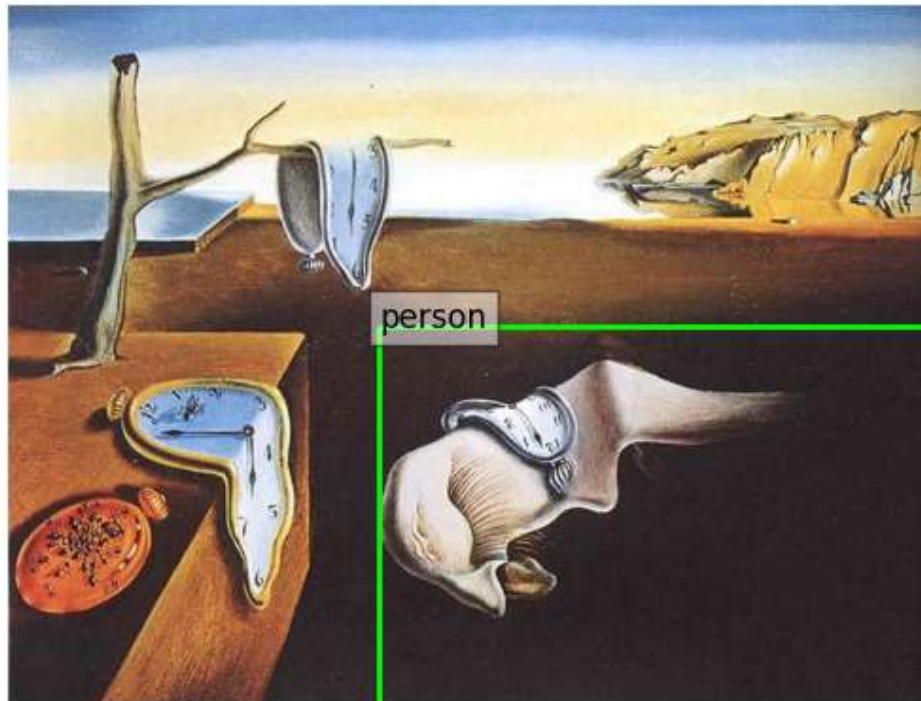
---



## 03-1 Detector Ideation

YOLO와 같은 detection 모델들은  
살바도르 달리의 그림에서  
초현실적인 시계를 '시계'라고  
detect해낼 수 있을까?

YOLO's result:



기존 모델은 detection 실패!

그렇다면, DALL-E 모델로  
synthetic data를 다량으로 생산한 후  
이 데이터들로 학습을 시킨 Faster R-CNN이  
달리의 그림 속 object를 올바르게 detect하는지  
확인해보자!

prompt:

a surrealism melting clock drawn by salvador dali

DALL-E's output:

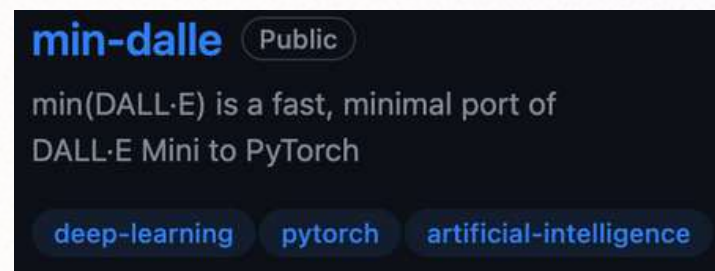


Synthetic  
data



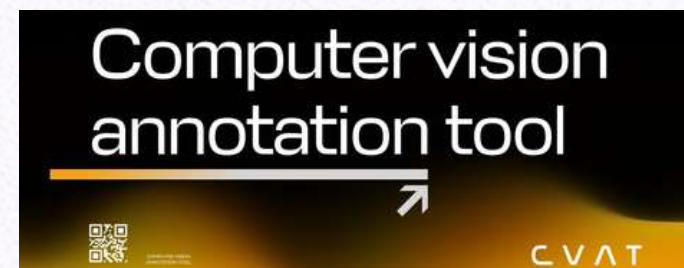
## 03-2 Detector Pipeline

### DALL-E로 Synthetic data 생성



- DALL-E의 경량화 버전인 **min-dalle** 모델 사용
- 달리의 그림에 자주 등장하는 object 9가지(elephant, horse, clock, human, car, angel, violin, bed, cloud)를 class로 지정
- 총 370장의 이미지 생성

### Annotation



- 생성한 이미지들의 bounding box를 CVAT로 annotation
- fasterRCNN 학습시 PASCAL format (xmin, ymin, xmax, ymax, class) 로 학습함.

### Faster R-CNN 학습

FASTERRCNN\_RESNET50\_FPN

```
torchvision.models.detection.fasterrcnn_resnet50_fpn(*, weights:
Optional[FasterRCNN_ResNet50_FPN_Weights] = None, progress:
bool = True, num_classes: Optional[int] = None, weights_backbone:
Optional[ResNet50_Weights] = ResNet50_Weights.IMAGENET1K_V1,
trainable_backbone_layers: Optional[int] = None, **kwargs: Any)
→ FasterRCNN [SOURCE]
```

- detection task에 적합하고, 정확도 면에서 우수한 RPN 기반 **Faster R-CNN** 선택
- resnet\_50\_fpn을 backbone으로 하는 모델 구성



## 03-3 Generate Synthetic Data



Hyperparameter Experiment 과정: temperature parameter 0.1, 1, 3, 5, 8, 10 나열 비교

### prompt:

Surrealism Clouds for main topic of image, additionally small horse, car, angels, melting clock drawn by Salvador Dali. Objects' locations are diverse. Objects' sizes are diverse

- 하나의 object만 생성하는 경우, object의 위치가 정중앙으로 고정되다보니 location에 대한 학습이 자칫 잘못될 수 있음.
- 이를 해결하기 위해 "여러 class"의 object를 "다양한 위치"에 생성하는 prompt를 작성하려고 노력함.



### Hyperparameter

- temperature: 8
- supercondition\_factor: 64
- top\_k: 128

### DALL-E's output:





## 03-4 Train Faster R-CNN

□ image와 bounding box 정보를 담은  
**custom dataset** 정의

```
import os
import torch
from PIL import Image
import torchvision.transforms as T

class CustDat(torch.utils.data.Dataset):
    def __init__(self, df, unique_imgs, indices):
        self.df = df
        self.unique_imgs = unique_imgs
        self.indices = indices

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

    def __getitem__(self, idx):
        image_name = self.unique_imgs[self.indices[idx]]
        # 이미지 ID에 해당하는 데이터만 필터링
        image_data = self.df[self.df.image_id == image_name]

        # 바운딩 박스 좌표를 추출 (x1, y1, x2, y2 형태로)
        boxes = image_data[['x1', 'y1', 'x2', 'y2']].values.astype('float')

        # class 열에서 레이블을 추출
        labels = image_data['class'].values.astype('int64')

        # 이미지 파일을 열고, RGB로 변환
        img = Image.open(os.path.join(os.getcwd(), "data_jiyoan/", image_name)).convert("RGB")

        # 타겟 딕셔너리 생성
        target = {}
        target["boxes"] = torch.tensor(boxes, dtype=torch.float32)
        target["labels"] = torch.tensor(labels, dtype=torch.int64)

        # 이미지를 텐서로 변환하여 반환
        return T.ToTensor()(img), target
```

```
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
num_classes = 9
in_features = model.roi_heads.box_predictor.cls_score.in_features # classification score
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes) # bbox predictor
```

### Architecture

- **transform**: GeneralizedRCNNTransform
- **feature extract**: 1~5 layer constructed by bottleneck
- **FPN**: FeaturePyramidNetwork
- **RPN**: RegionProposalNetwork
- **Rol**: RolHeads
- **box\_predictor**: FastRCNNPredictor

### Environment

- GPU:
  - epoch:
  - batch size:
  - learning rate:
- 
- Backbone : resnet50\_fpn (pre-trained)



## 03-5 Train Faster R-CNN

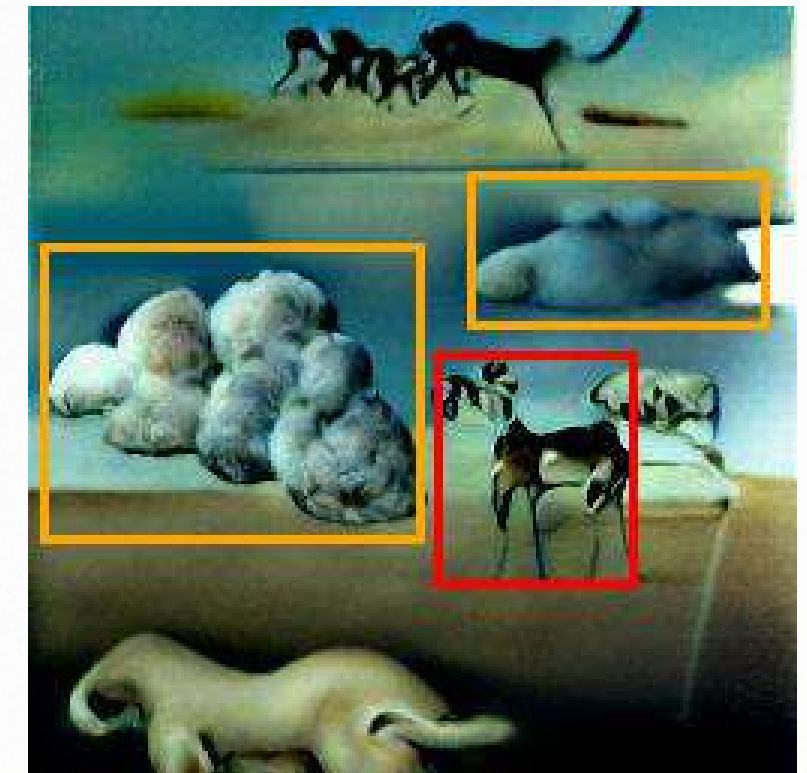
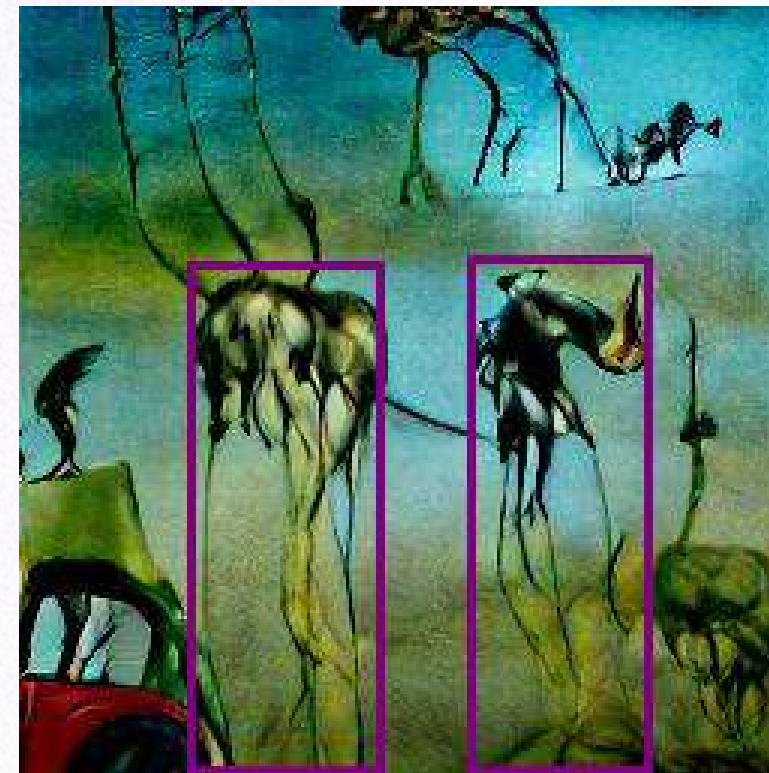
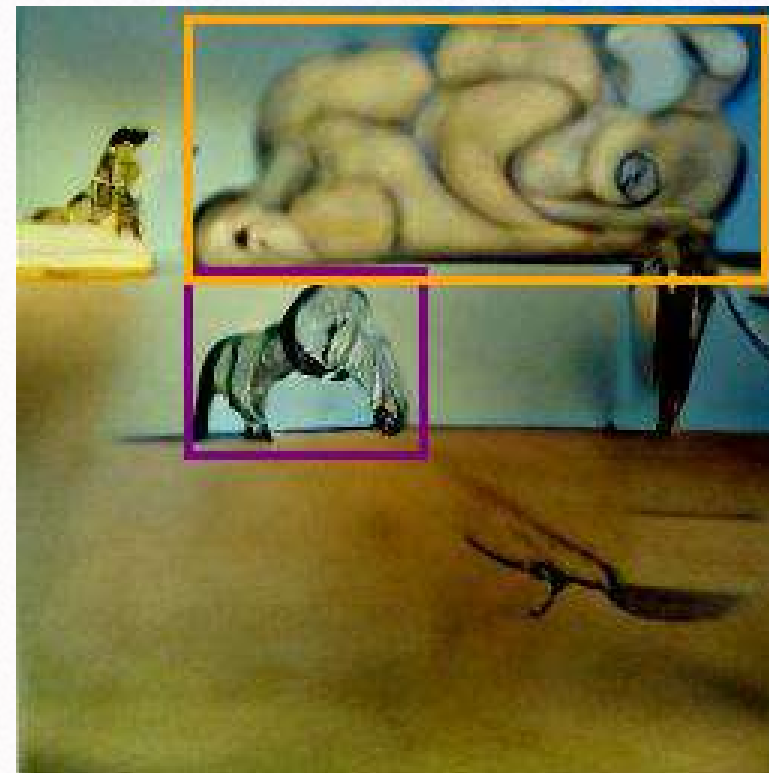
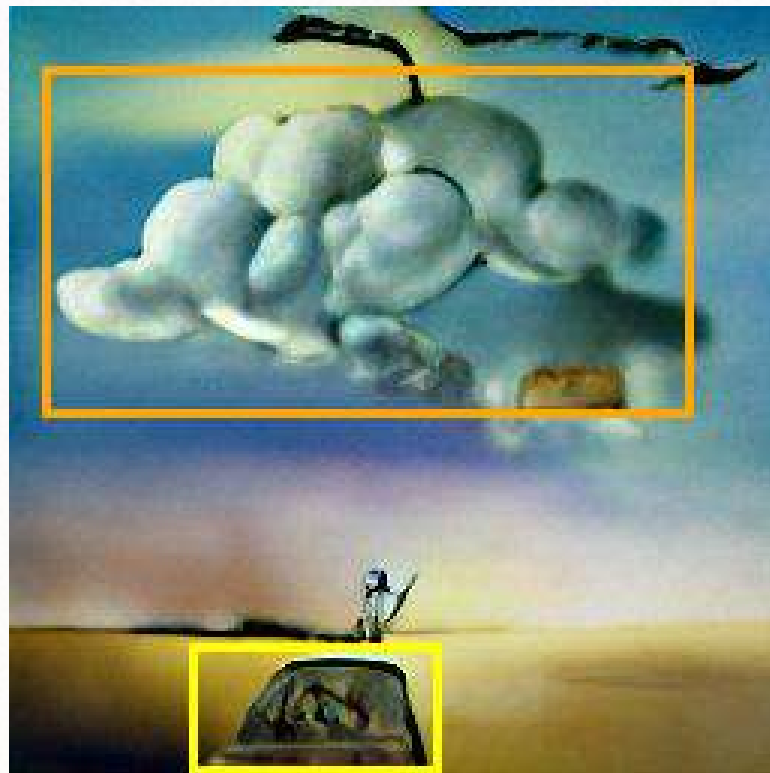
---

---

### Result

---

- Validation Set 에 대한 Detection Result

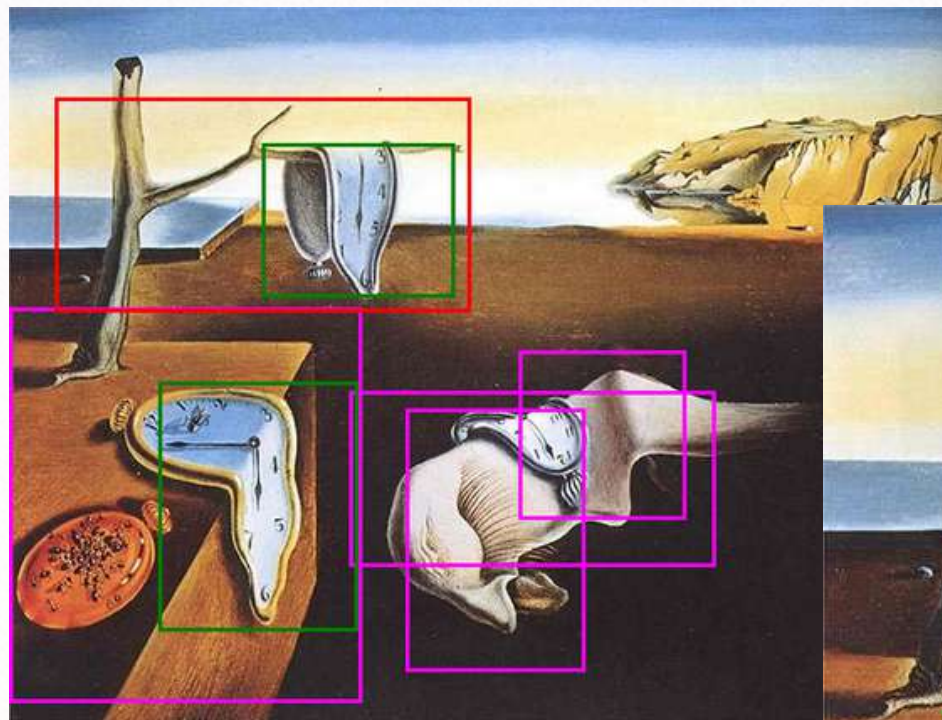




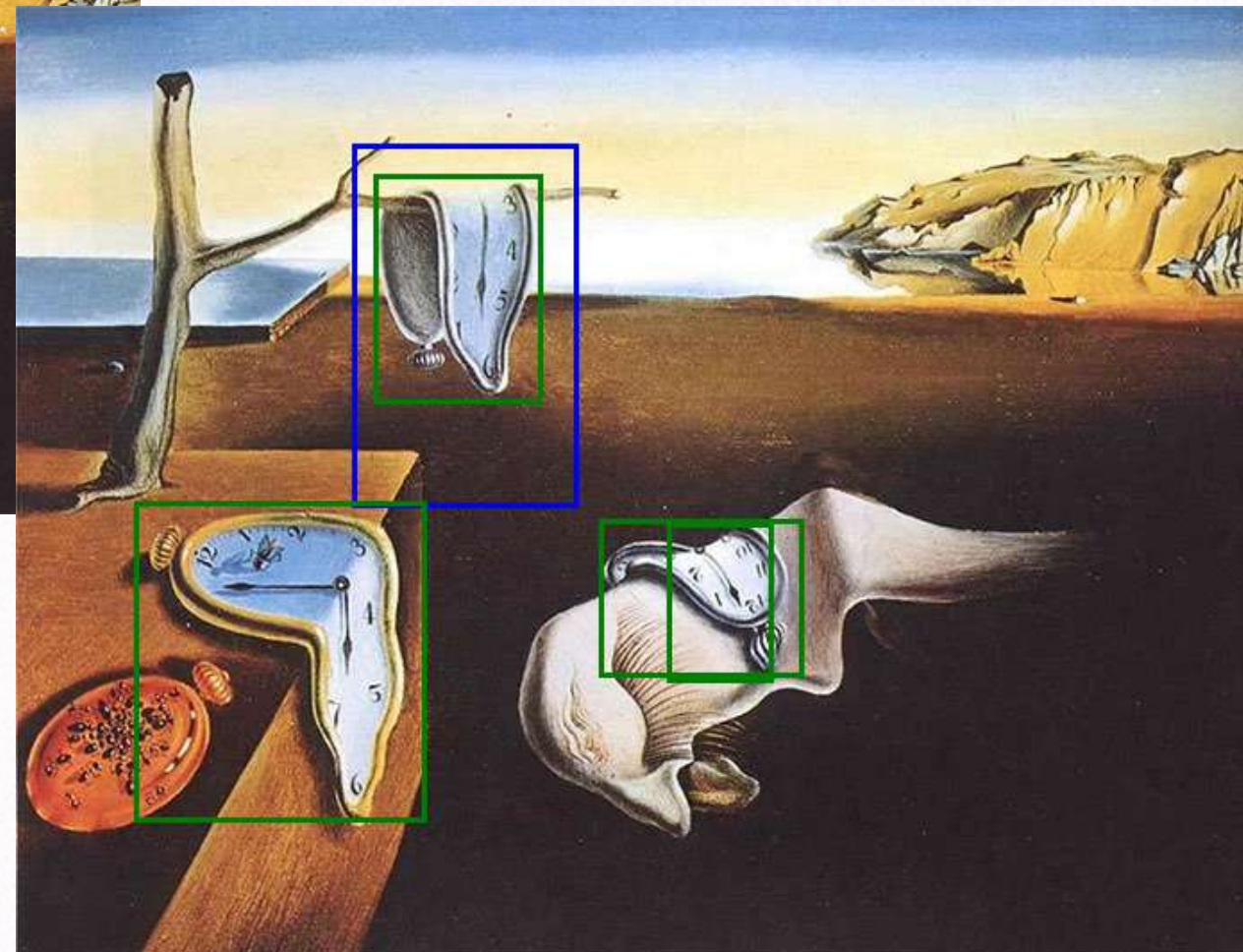
## 03-5 Train Faster R-CNN

### Result

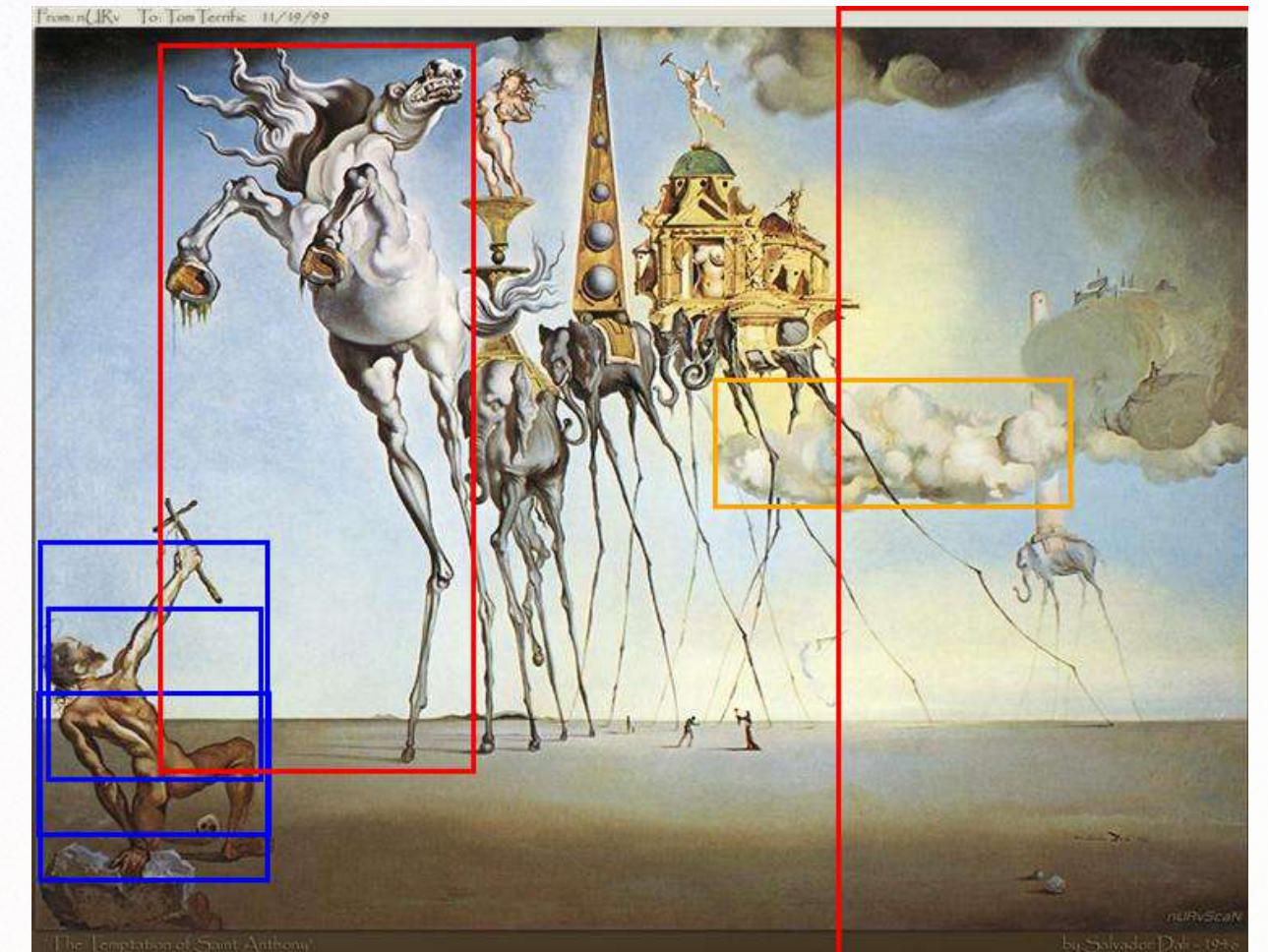
- Test Set (달리의 작품) 에 대한 Detection Result



<- 실제 작품 추가 전



실제 작품 추가 후





# 04. DISCRIMINATOR

Let's Discriminate Real or Fake!

01  
MOTIVATION

---

02  
PRE-STUDY

---

03  
DETECTOR

---

04  
DISCRIMINATOR

---

05  
CONCLUSION

---



## 04-1 Discriminator Ideation

---

### **DALL-E2로 만든 이미지는 누구 것? 법률 전문가들이 답하다**

---

글로벌 로펌 베이커 매킨지의 AI와 머신러닝을 담당하는 브래드퍼드 뉴먼(Bradford Newman) 변호사는 "DALL-E 이미지를 누가 소유하느냐는 질문에 대한 답은 분명하지 않다, 시장에 많은 참여자가 있고 문제가 공통적으로 걸렸을 때 소송 가능성이 높다"고 밝혔다.

출처 : AI타임스

2022. 08. 17

### **Fake AI images keep going viral – here are eight that have caught people out**

---

AI로 생성된 이미지들이 실제로는 존재하지 않는 유명 인물들의 사진이나 주요 사건들을 묘사하고 있으며 이로 인해 많은 사람들에게 혼란을 주는 사태가 발생하고 있다. 크리스 모리스 CEO는 "온라인에서 보는 것의 사실 파악이 부족할 경우, 민주주의를 악화시킬 우려가 있다"고 밝혔다.

출처 : Sky News

2023. 12. 13



## 04-1 Discriminator Ideation

---

“

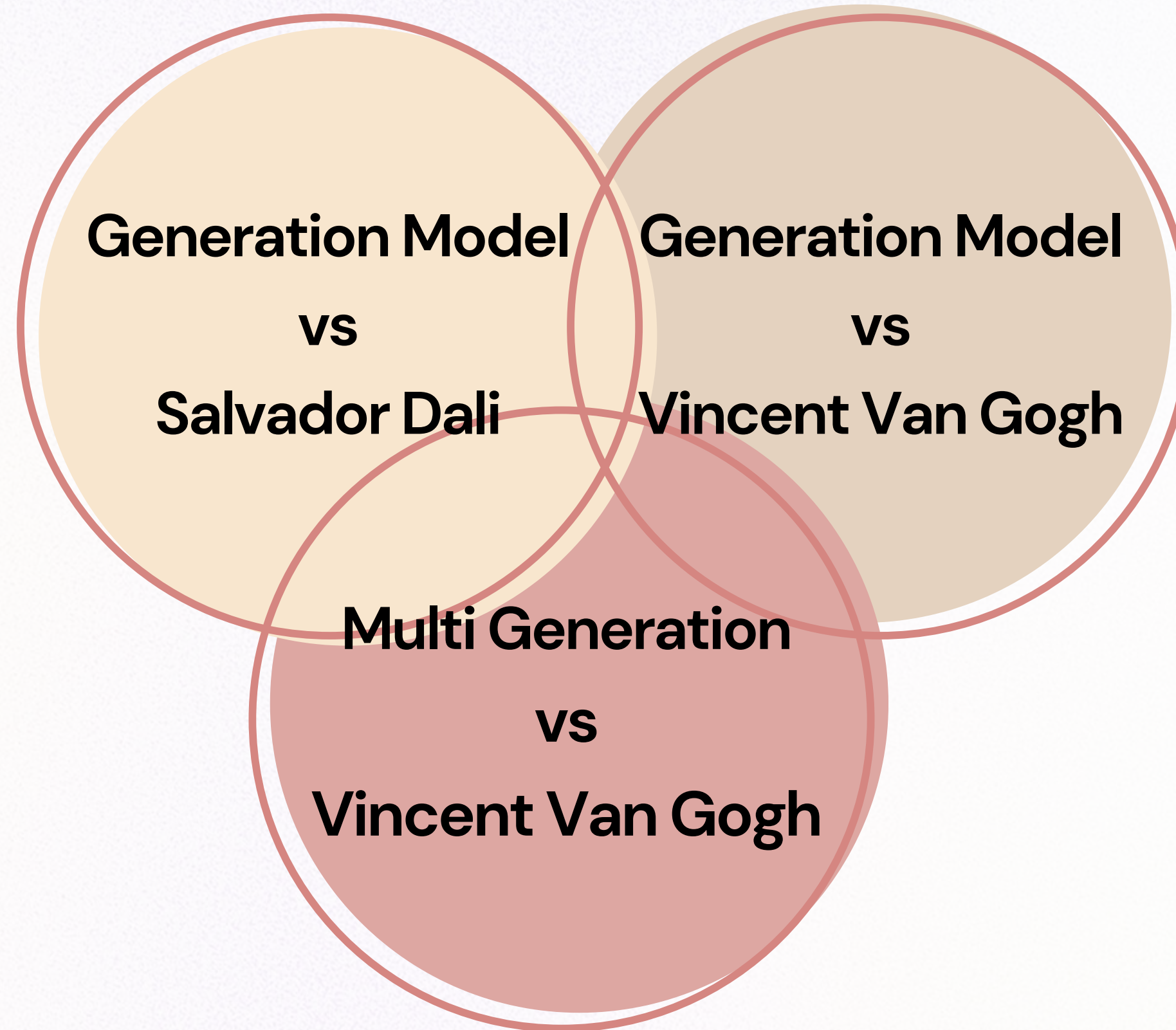
DALL-E가 만든 synthetic data의 이점을 누렸으니  
이번에는 단점을 보완할 수 있는 **Discriminator**를 만들어보자

”



## 04-2 Discriminator Pipeline

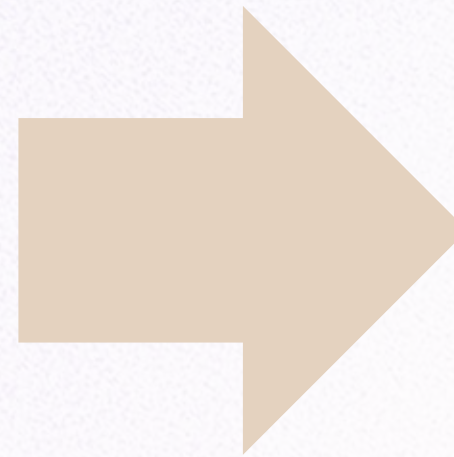
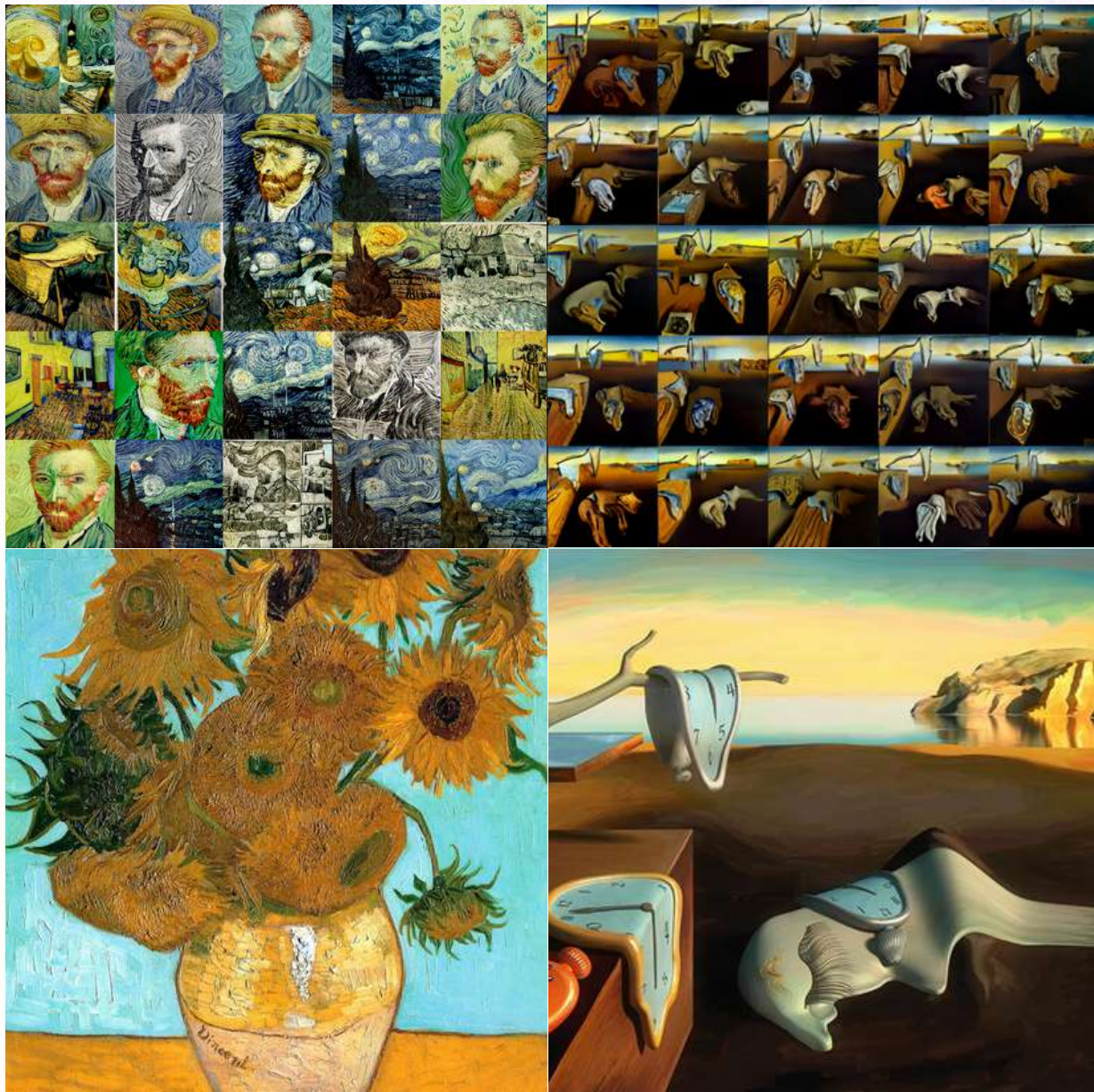
---





## 04-3 Train Discriminator

### Image Dataset



### Discriminator

```
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, inplace=True),

            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(256, 512, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Flatten(),
            nn.Linear(512 * 16 * 16, 1),
            nn.Sigmoid()
        )

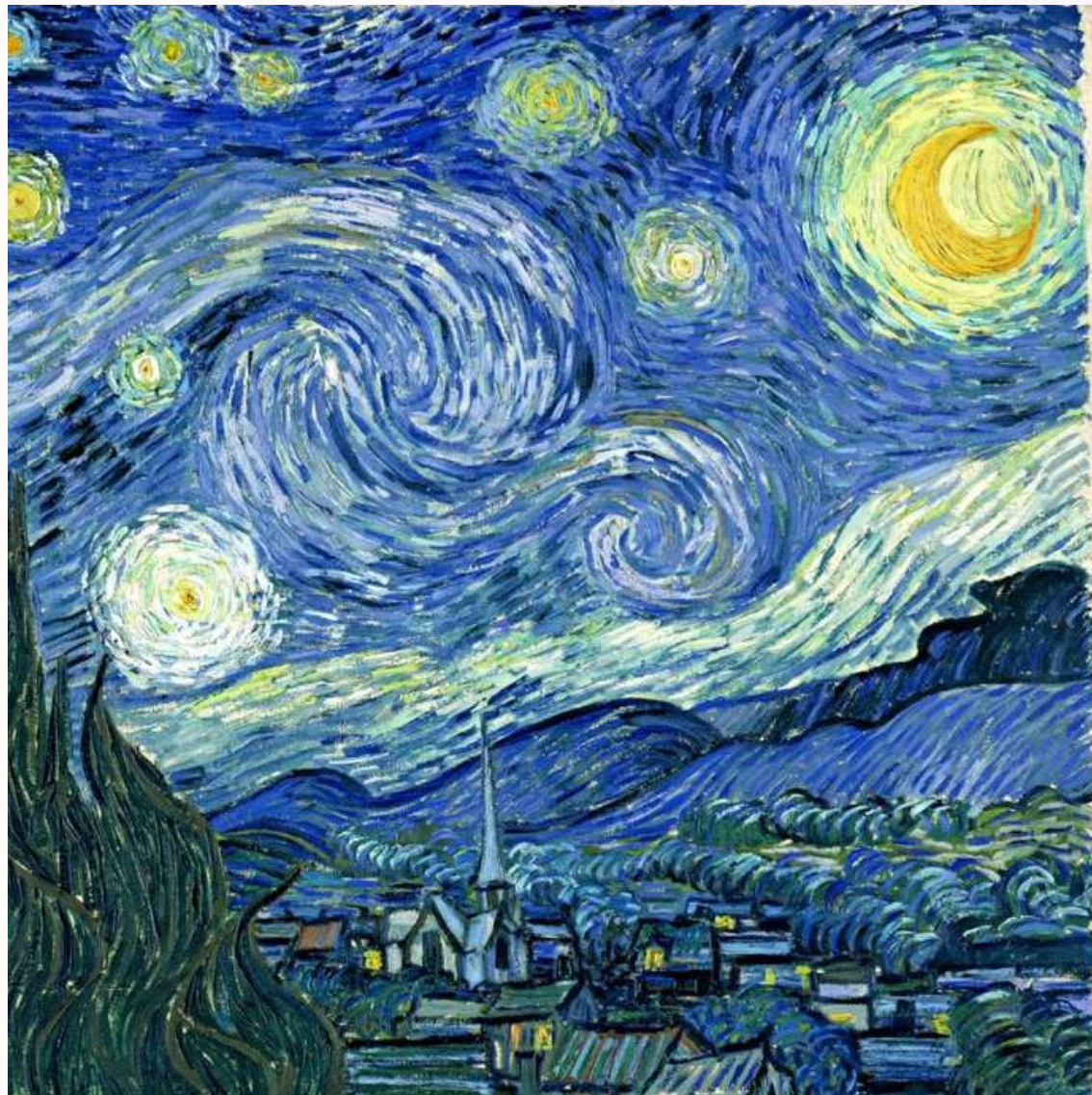
    def forward(self, x):
        return self.model(x)
```



## 04-3 Quiz

---

무엇이 진짜 '별이 빛나는 밤'일까요?





## 04-3 Discriminator (Result)

```
Label: Fake, Predicted: Fake, Confidence: 1.2974180791907526e-37
Label: Real, Predicted: Real, Confidence: 1.0
(cv) tjddms9376@datad2:~/cv$ cd /home/tjddms9376/cv ; /usr/bin/er
conda3/envs/cv/bin/python /home/tjddms9376/.vscode-server/extensio
23.22.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launc
dms9376/cv/discriminator/eval.py
Label: Fake, Predicted: Fake, Confidence: 0.0
Label: Real, Predicted: Real, Confidence: 1.0
(cv) tjddms9376@datad2:~/cv$ cd /home/tjddms9376/cv ; /usr/bin/er
conda3/envs/cv/bin/python /home/tjddms9376/.vscode-server/extensio
23.22.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launc
dms9376/cv/discriminator/eval.py
Label: Fake, Predicted: Fake, Confidence: 2.2766425582650595e-35
Label: Real, Predicted: Real, Confidence: 0.9999998807907104
(cv) tjddms9376@datad2:~/cv$ cd /home/tjddms9376/cv ; /usr/bin/er
conda3/envs/cv/bin/python /home/tjddms9376/.vscode-server/extensio
23.22.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launc
dms9376/cv/discriminator/eval.py
Label: Fake, Predicted: Fake, Confidence: 4.232624477778407e-36
Label: Real, Predicted: Real, Confidence: 1.0
(cv) tjddms9376@datad2:~/cv$ cd /home/tjddms9376/cv ; /usr/bin/er
conda3/envs/cv/bin/python /home/tjddms9376/.vscode-server/extensio
23.22.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launc
dms9376/cv/discriminator/eval.py
Label: Fake, Predicted: Fake, Confidence: 0.0
Label: Real, Predicted: Real, Confidence: 1.0
```

**REAL**



**FAKE**





## 04-3 Discriminator (EfficientNET)

---



# 05. CONCLUSION

Significance and Limitation

01  
MOTIVATION

---

02  
PRE-STUDY

---

03  
DETECTION

---

04  
DISCRIMINATOR

---

05  
CONCLUSION

---



## 05-1 □ Trial and Error

### Classification

기존의 classification 모델들은 달리의 그림에서 초현실적인 시계를 '시계'라고 분류하지 못한다.

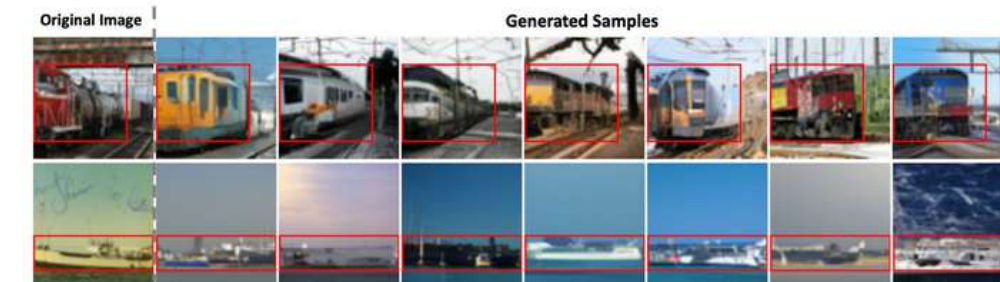
“  
반면, Synthetic data로 학습을 거친 모델은  
올바르게 분류해낼 것이다!  
”



무난하게 classification 성공!

### detection dataset 구축을 위해 self guided bounding box 시도

bounding box를 자동으로 annotate해주는 self-guided model로 분류된 '시계'가 어디 있는지 검출해보자!



self guided diffusion이라는 model을 활용!

- bounding box가 그려진 이미지를 input으로 넣으면, bbox 위치 정보를 보존한 채 새로운 이미지를 생성
- bounding box가 없는 이미지를 input으로 넣으면, 각각의 bounding box를 직접 제작
- model 불러오는 과정에서 오류 + 기술적 한계



## O5-1 □ Trial and Error

'달리 화풍의 시계'에서 화풍을 없애고, **현실같은 시계로 transfer**할 수 있을까?  
즉, 기존의 **style transfer**의 **inverse**한 과정을 수행할 수 있을까?

**Cycle GAN** 활용하여 style transfer 시도  
=> 결과가 좋지 못했으나, 향후 과제로 시도해볼 가치 충분



Content image



style image



result



## 05-2 Limitation

---



Self-guided detection & CycleGAN과 같은  
모델의 기술적 한계



Human annotation을 이용한 데이터 셋 구축으로  
훈련 데이터를 충분히 확보하기 어려움



Salvador Dali와 같이 다양한 화풍을 가진 화가에 대한  
Discriminator 성능 향상 실패



## 05-2 Significance

---



Text-to-Image 모델 활용을 통해  
Multi-modal에 대한 이해도 향상



Text 하나만으로 다양한 task에 사용 가능한  
이미지 데이터 확보




Synthetic data가 가지는 사회적 이슈를  
보완하기 위한 시도



**감사합니다!**

**달리를 찾아라!**

 Dali, Van, Picasso, CV 1팀 Let's go  
16기 이영노 / 17기 김지윤, 김희준, 문성빈 / 18기 백성은