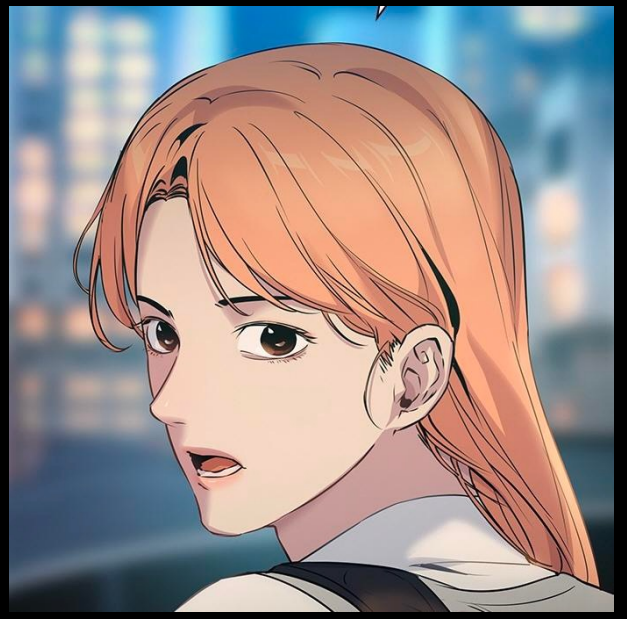
Ajou Univ 2023-Spring Machine Learning - Team Project Final Presentation

A cGAN-based Approach for Manga Colorization

Team ACV: Ajou ComicArt Visionaries

한승훈, 이헌우, 이은구, 박병하, 민동현

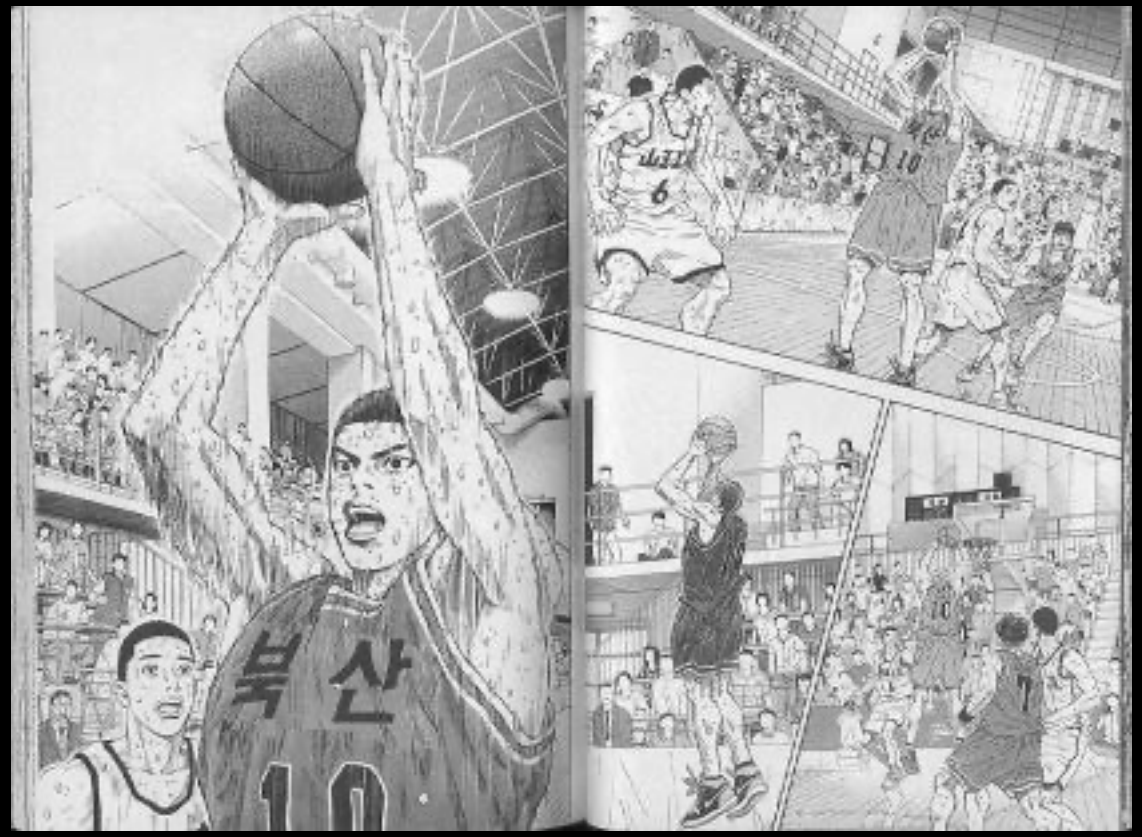
Abstract & Problems to solve



딥러닝 기술을 활용하여 흑백 만화의 채색을 자동화

Introduction

Background



Introduction

Requirements, assumptions, risks and constraints

Requirements	Assumptions	Risks and Constraints
Deep Learning Model	Coloring Quality	Insufficient computational resources
Dataset	Traning & Test Dataset	Legal Issue
Computing Resource	-	Gan Instability

Data

Index	Data name	Episode
1	NAVER Webtoon - 세상은 돈과권력	#1 ~#221
2	NAVER Webtoon - 신의탑	#400 ~ #567

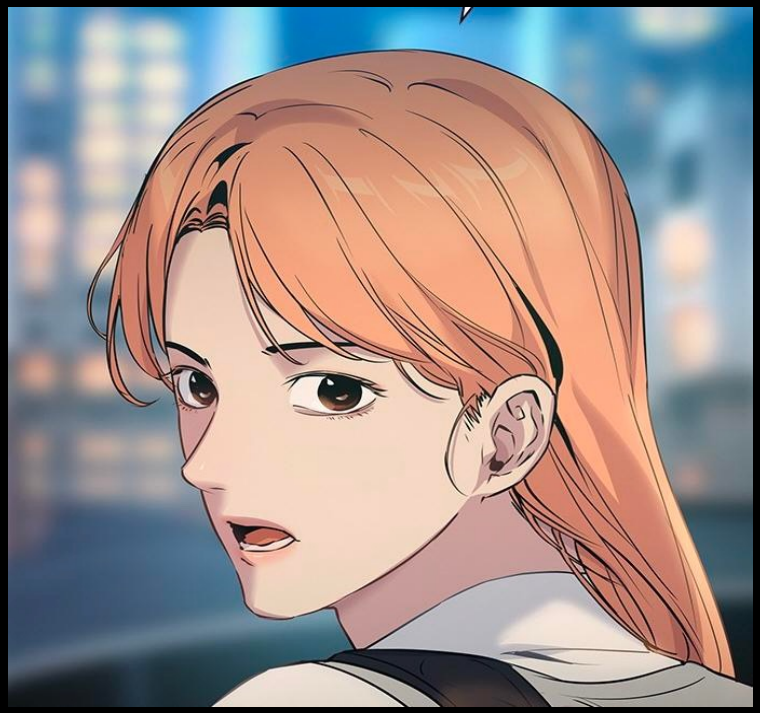
Data

Data Collection Method - 1

NAVER Webtoon에서 크롤링한 웹툰 이미지 중 일부

Data

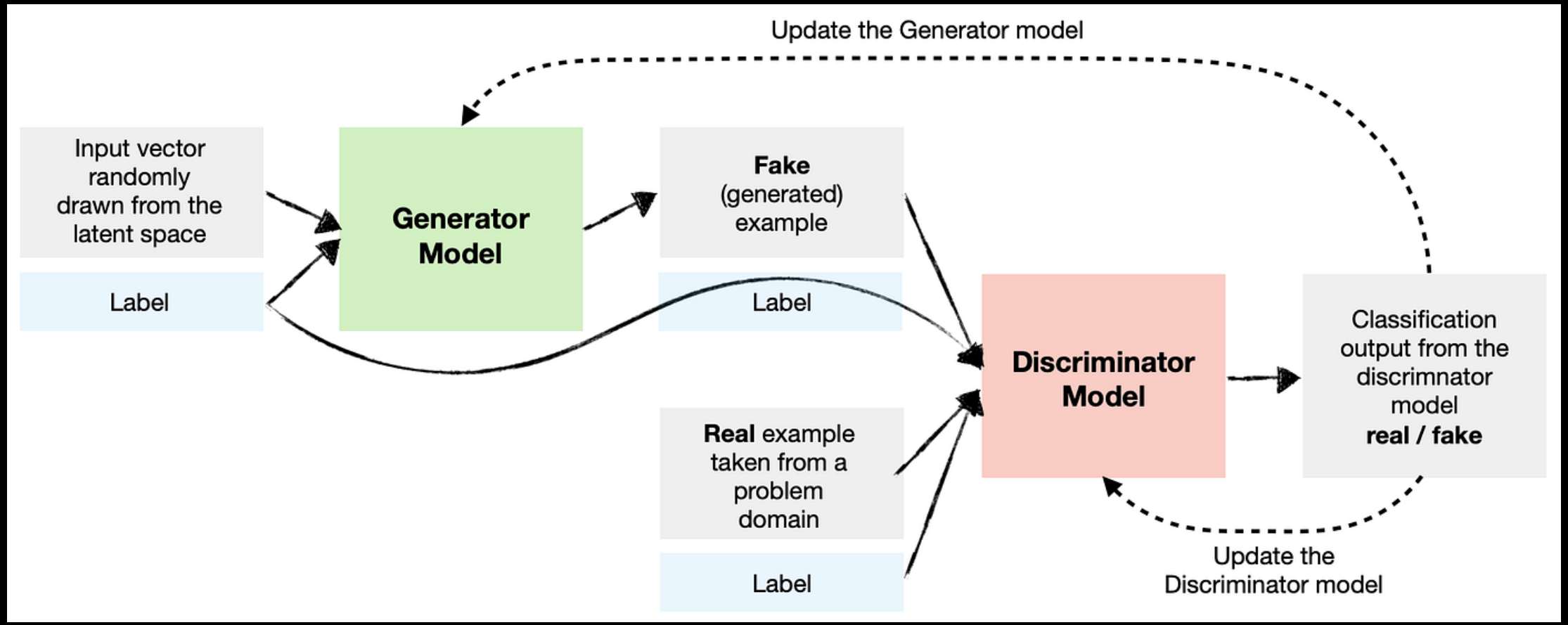
Data Collection Method - 2



OpenCV활용 전처리를 진행한 데이터 셋 중 일부

Method

Model / Algorithm description



Reference: https://towardsdatascience.com/cgan-conditional-generative-adversarial-network-how-to-gain-control-over-gan-outputs-b30620bd0cc8

Method

Model / Algorithm description

PyTorch

Get Started | Ecosystem | Mobile | Blog | Tutorials | Docs | Resources | GitH

2.0.1+cu117

Search Tutorials

PyTorch Recipes [+]

Introduction to PyTorch [-]

Learn the Basics

Tutorials > DCGAN Tutorial

Run in Google Colab | Download Notebook | View on GitHub

DCGAN TUTORIAL

Author: Nathan Inkawhich

cGAN-based Manga Colorization Using a Single Training Image

Paulina Hensman, Kiyoharu Aizawa

The Japanese comic format known as Manga is popular all over the world. It is traditionally produced in black and white, and color
methods generally rely on greyscale values, which are not present in manga. Furthermore, due to copyright protection, colorized n
colorization method based on conditional Generative Adversarial Networks (cGAN). Unlike previous cGAN approaches that use mar
requires only a single colorized reference image for training, avoiding the need of a large dataset. Colorizing manga using cGANs
limited. We therefore also propose a method of segmentation and color-correction to mitigate these issues. The final results are s
character's original color scheme.

Comments: 8 pages, 13 figures
Subjects: **Graphics (cs.GR)**; Computer Vision and Pattern Recognition (cs.CV)
MSC classes: 68U10, 68U05
Cite as: arXiv:1706.06918 **[cs.GR]**
(or arXiv:1706.06918v1 **[cs.GR]** for this version)
https://doi.org/10.48550/arXiv.1706.06918

Submission history
From: Kiyoharu Aizawa Dr. Prof. [view email]
[v1] Wed, 21 Jun 2017 14:11:32 UTC (9,737 KB)

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs

Ting-Chun Wang[1] Ming-Yu Liu[1] Jun-Yan Zhu[2] Andrew Tao[1] Jan Kautz[1] Bryan Catanzaro[1]
[1]NVIDIA Corporation [2]UC Berkeley

Reference
1. https://arxiv.org/abs/1706.06918
2. https://tcwang0509.github.io/pix2pixHD/
3. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html#training

Method

Model / Algorithm description

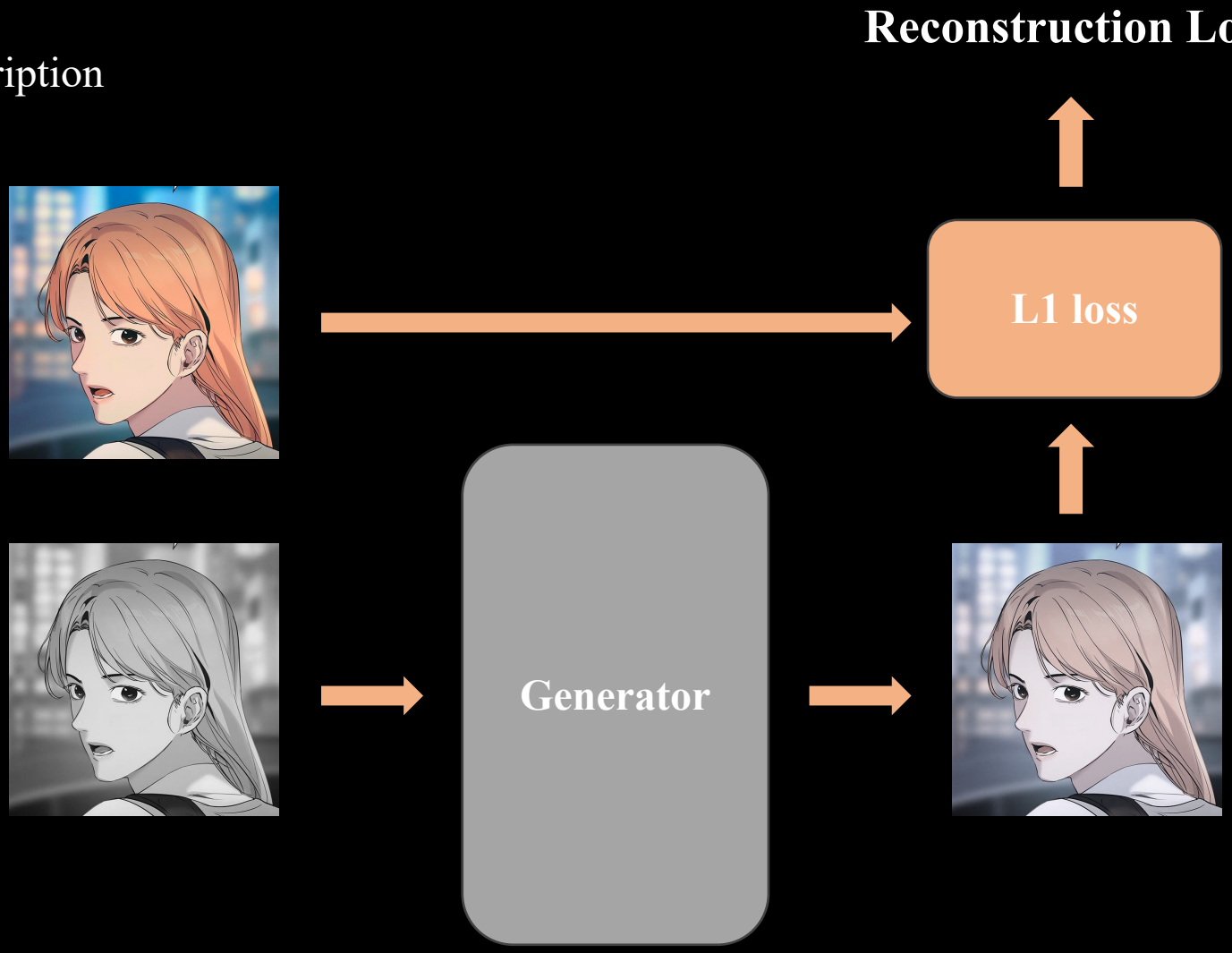
Objective function of cGAN structure

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

G()	**Generator**
D()	**Discriminator**
Z	**원본 컬러 만화 Dataset을 흑백 변환한 만화 Dataset**
X	**원본 컬러 만화 Dataset**
Log(1 – D(G(z)))	**흑백 변환한 만화 Dataset을 Generator에 입력해 컬러 만화를 생성하고 이를 Discriminator에 입력한 경우**
Log(D(x))	**원본 컬러 만화 Dataset을 Discriminator에 입력한 경우**

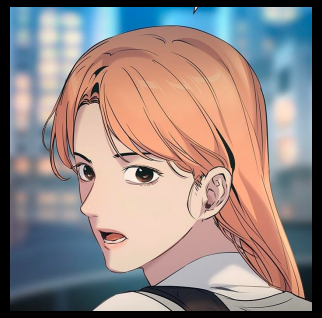
Method

Model / Algorithm description

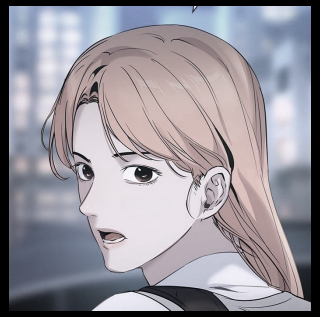


Method

Model / Algorithm description



Generator



Discriminator

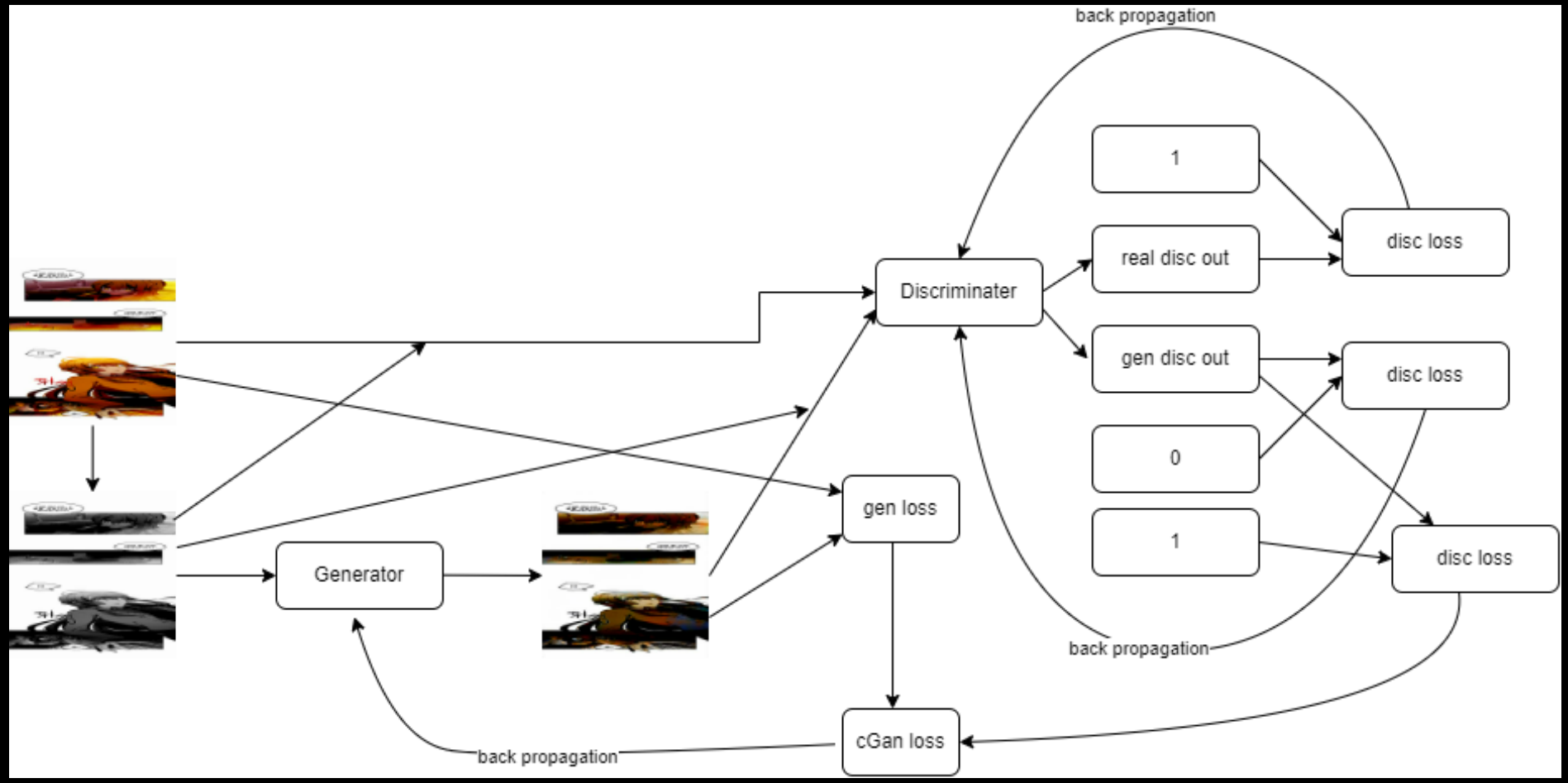
Loss on probability of REAL

Adversarial Loss (Loss on probability of FAKE)

Binary Cross Entropy Loss

Method

Model / Algorithm description



Method

Model / Algorithm description

Loss on probability of REAL + **Adversarial Loss (Loss on probability of FAKE)** = **Binary Cross Entropy Loss** = **Discriminator Loss**

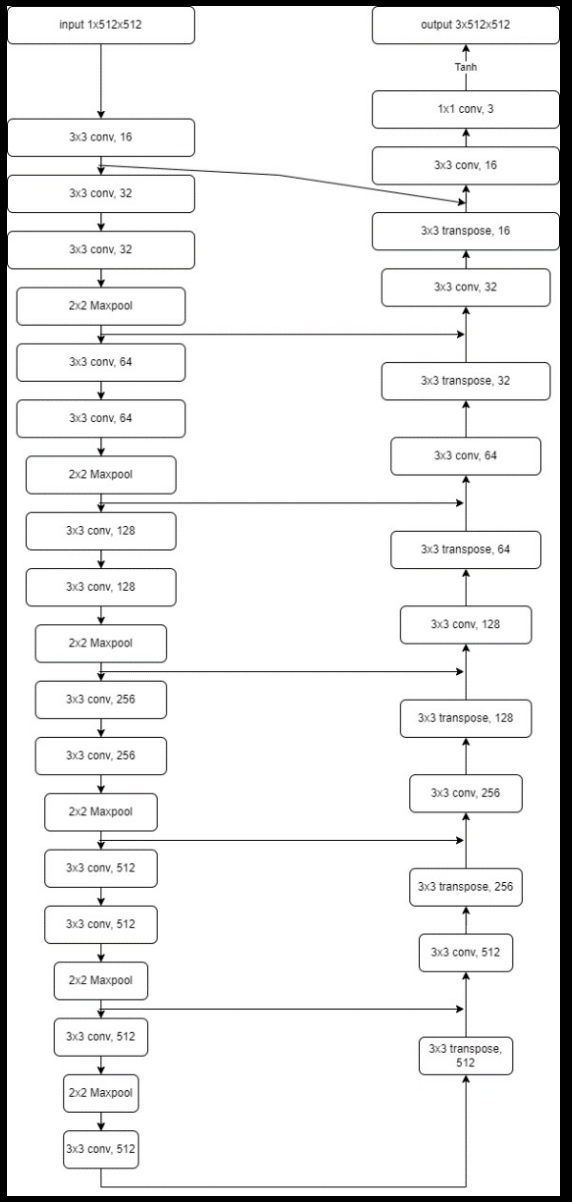
Adversarial Loss (Loss on probability of FAKE) + **Reconstruction Loss** = **Generator Loss**

Method

Model / Algorithm description

Generator

layer	outputsize	(LeakyLeLU, InstanceNorm)
conv1	512x512	3x3, 16, pad = 1
conv2	512x512	3x3, 32, pad = 1
	512x512	3x3, 32, pad = 1
	256x256	maxpool 2x2
conv3	256x256	3x3, 64, pad = 1
	256x256	3x3, 64, pad = 1
	128x128	maxpool 2x2
conv4	128x128	3x3, 128, pad = 1
	128x128	3x3, 128, pad = 1
	64x64	maxpool 2x2
conv5	64x64	3x3, 256, pad = 1
	64x64	3x3, 256, pad = 1
	32x32	maxpool 2x2
conv6	32x32	3x3, 512, pad = 1
	32x32	3x3, 512, pad = 1
	16x16	maxpool 2x2
conv7	16x16	3x3, 512, pad = 1
	8x8	maxpool 2x2
conv8	8x8	3x3, 512, pad = 1
conv9	16x16	transpose 3x3, 512, stride = 2, pad = 1
	16x16	concat(conv6)
	16x16	3x3, 512, pad = 1
conv10	32x32	transpose 3x3, 256, stride = 2, pad = 1
	32x32	concat(conv5)
	32x32	3x3, 256, pad = 1
conv11	64x64	transpose 3x3, 128, stride = 2, pad = 1
	64x64	concat(conv4)
	64x64	3x3, 128, pad = 1
conv12	128x128	transpose 3x3, 64, stride = 2, pad = 1
	128x128	concat(conv3)
	128x128	3x3, 64, pad = 1
conv13	256x256	transpose 3x3, 32, stride = 2, pad = 1
	256x256	concat(conv2)
	256x256	3x3, 32, pad = 1

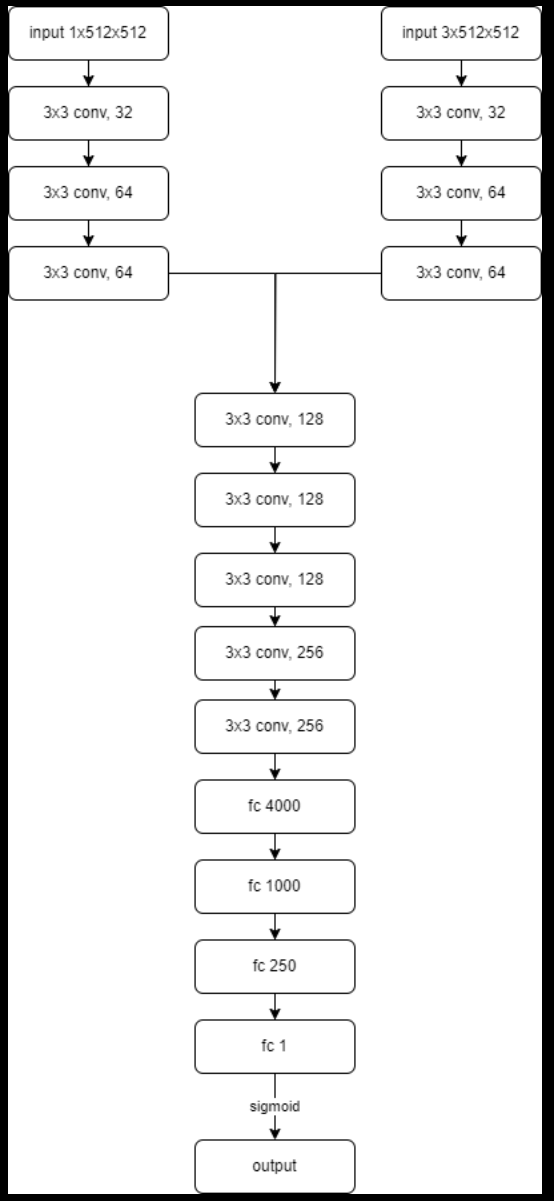


Method

Model / Algorithm description

Discriminator

layer	outputsize		
		3x3, 32, pad = 1	3x3, 32, pad = 1
		3x3, 64, pad = 1	3x3, 64, pad = 1
		3x3, 64, stride = 2, pad = 1	3x3, 64, stride = 2, pad = 1
		3x3, 128, stride = 2, pad = 1	
		3x3, 128, stride = 2, pad = 1	
		3x3, 128, stride = 2, pad = 1	
		3x3, 256, stride = 2, pad = 1	
		3x3, 256, stride = 2, pad = 1	
		fc 4000	
		fc 1000	
		fc 250	
		fc 1, Sigmoid	



Method

Model / Algorithm description

Loss function

- BCE Loss
- L1 Loss

Optimizer

- Adam optimizer

Activation function

- Leaky LeLU
- Tanh
- Sigmoid

Method

Model / Algorithm description

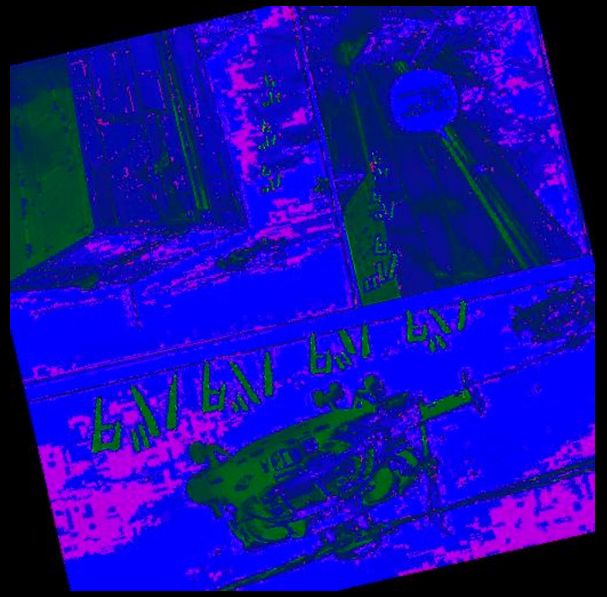
Learning rate : 0.0001

Batch size : 1, (메모리의 한계로 가장 작은 batch size를 사용)

Normalize : Instance norm

Method

Implementation & Struggles



Method

Implementation & Struggles

Epoch 수 부족

⬇

dropout

⬇

fully connected layer 완충

⬇

InstanceNorm

⬇

Custom Reconstruction loss function => L1 loss function

⬇

Model weight Save & Load

Method

Implementation & Struggles

Original

Generated

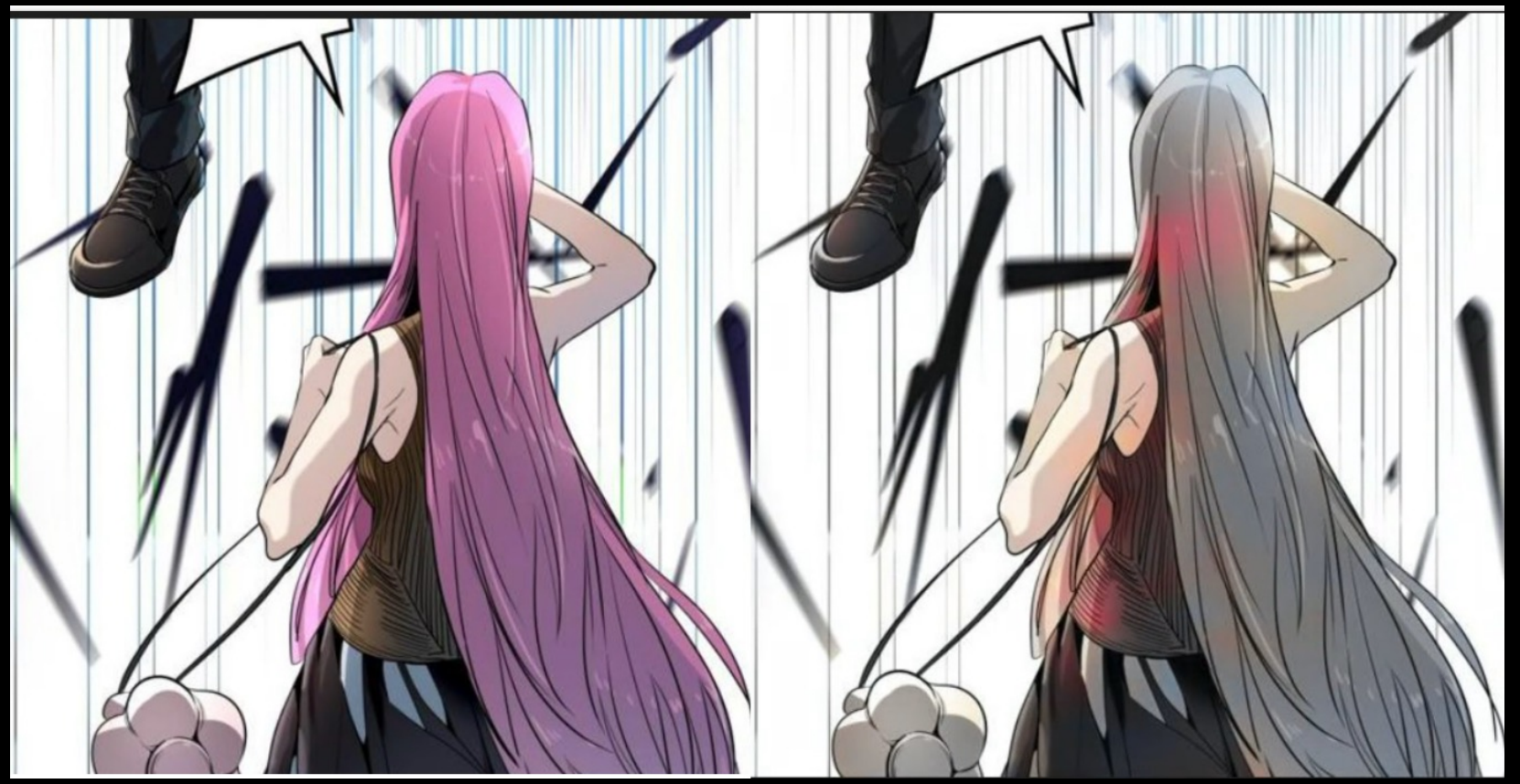
Method

Implementation & Struggles

Kaiming He Initialization

Method

Implementation & Struggles



Mode collapse: grayish color

Method

Implementation & Struggles

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Encoder-decoder (L1)	0.35	0.12	0.08
Encoder-decoder (L1+cGAN)	0.29	0.09	0.05
U-net (L1)	0.48	0.18	0.13
U-net (L1+cGAN)	**0.55**	**0.20**	**0.14**

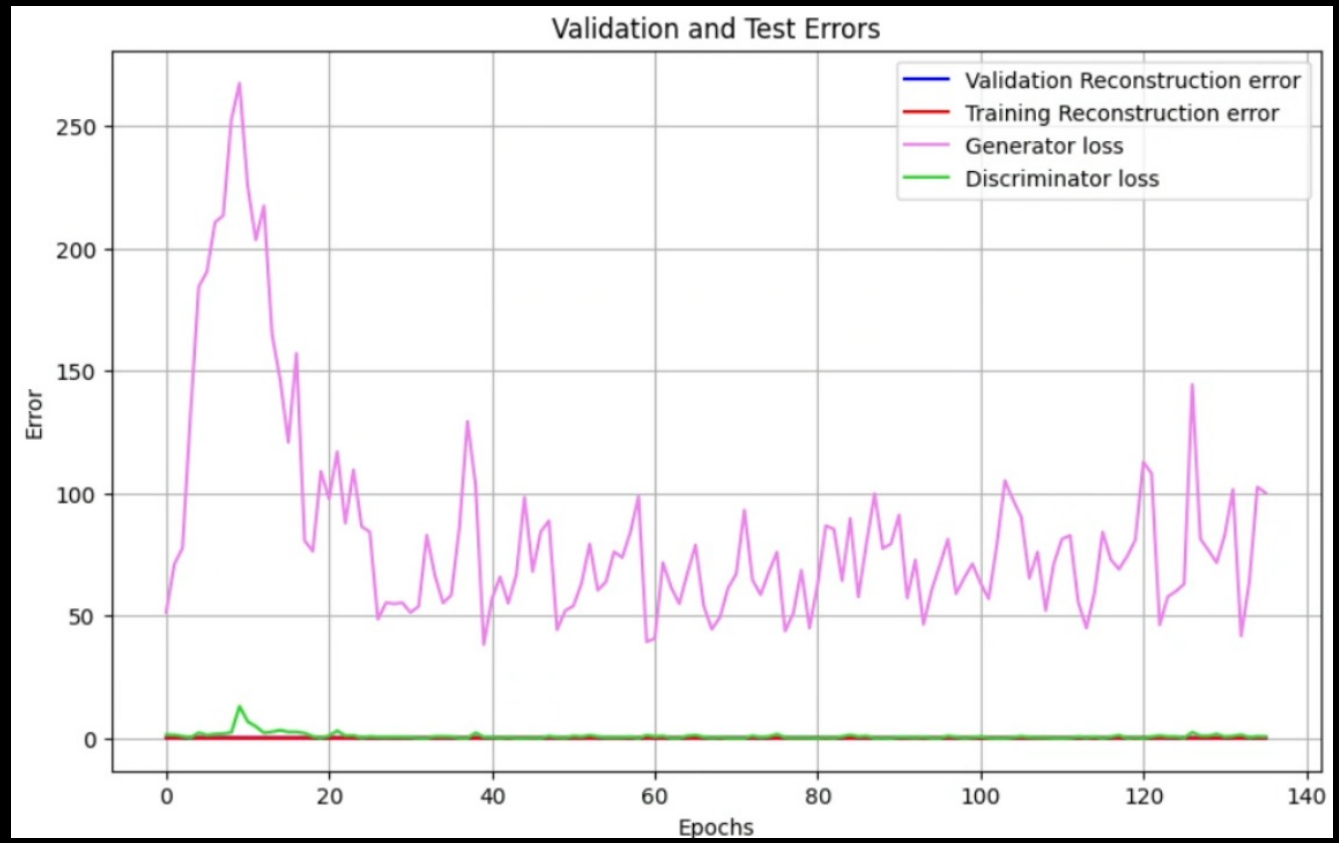
Table 2: FCN-scores for different generator architectures (and objectives), evaluated on Cityscapes labels↔photos. (U-net (L1-cGAN) scores differ from those reported in other tables since batch size was 10 for this experiment and 1 for other tables, and random variation between training runs.)

Applying the Unet Structure

Reference
1. [1611.07004] Image-to-Image Translation with Conditional Adversarial Networks (arxiv.org)

Method

Implementation & Struggles

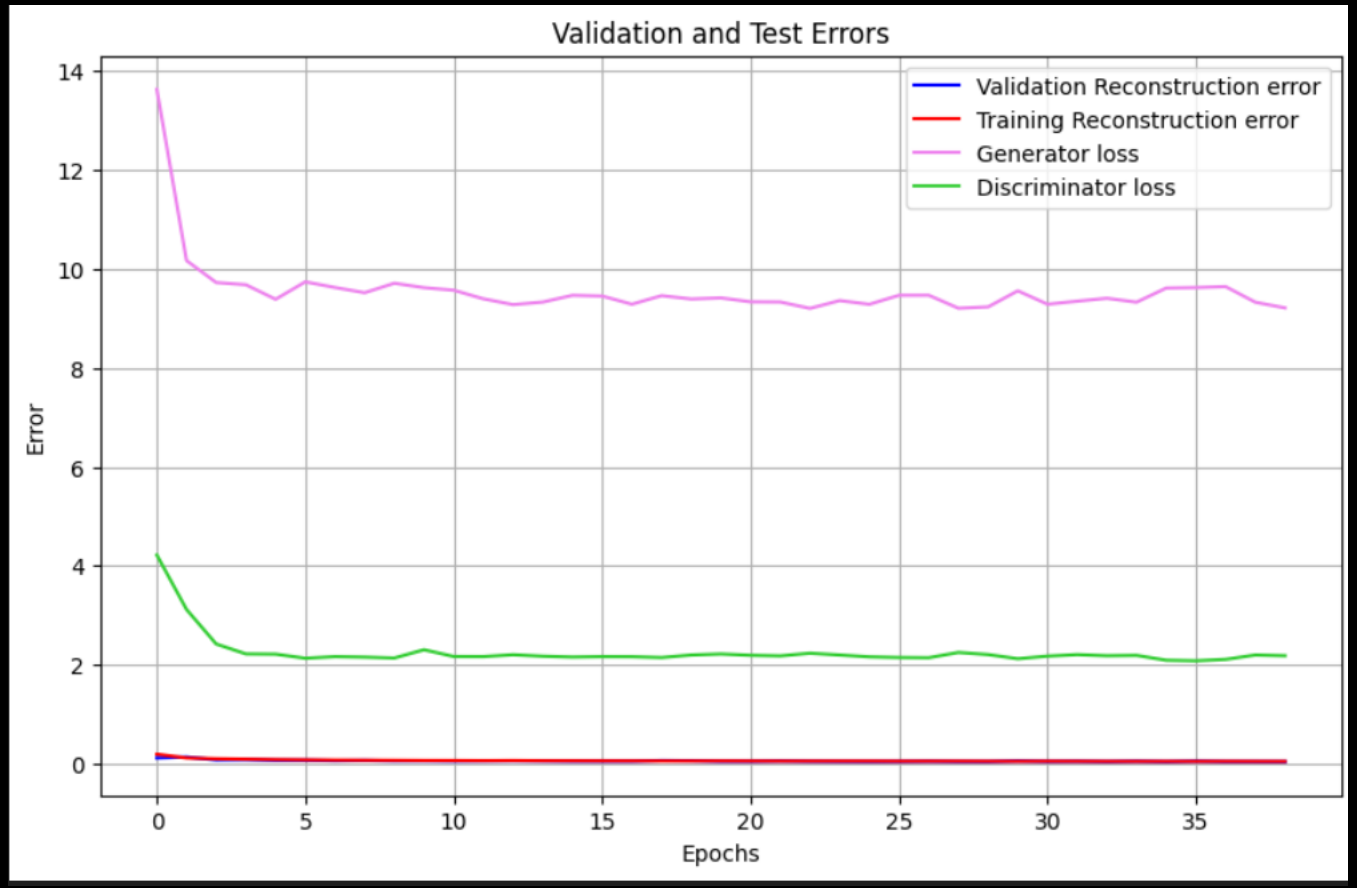


Problem: Discriminator loss converges to zero.

Solution:Weakening the discriminator by giving noise to the lables, layers of Disc.

Method

Implementation & Struggles

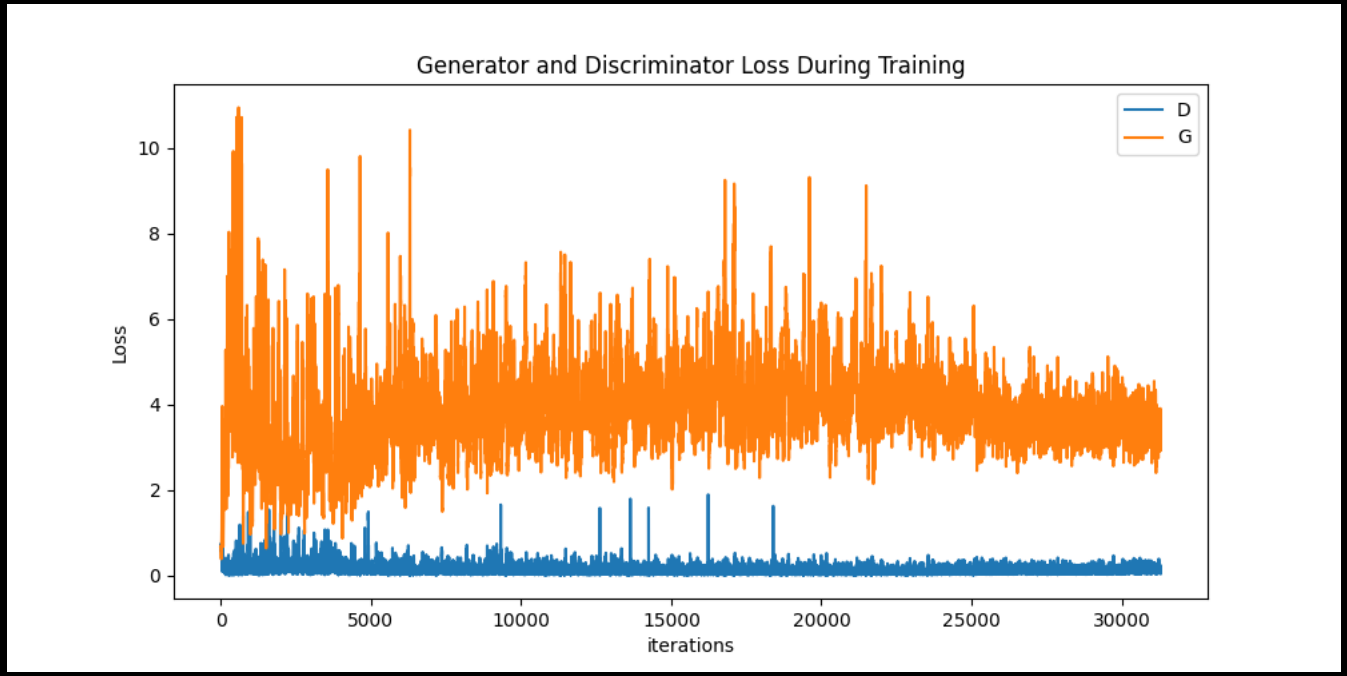


Result : Discriminator loss does not converge to zero

By giving noise to the lables, but still has some problems

Method

Implementation & Struggles



Ideal fluctuating graph

Reference
1. https://stackoverflow.com/questions/72815092/gan-loss-of-generator-and-discriminator-lowest-at-first-epoch-is-that-normal

Method

Implementation & Struggles

SOLUTION

(1) Add Discriminator Noise.

- Adding Gaussian Noise to each Disc Layer weakens learning

- The corresponding Gaussian Noise has a decay rate, and the noise is weakened through the layer

(2) Discriminator adds Noise to the correct answer label / sometimes flips the labe

- Current: [1, 0]

- Improvement: True label 0.7 ~1, fake label : 0 ~ 0.3

Method

Implementation & Struggles

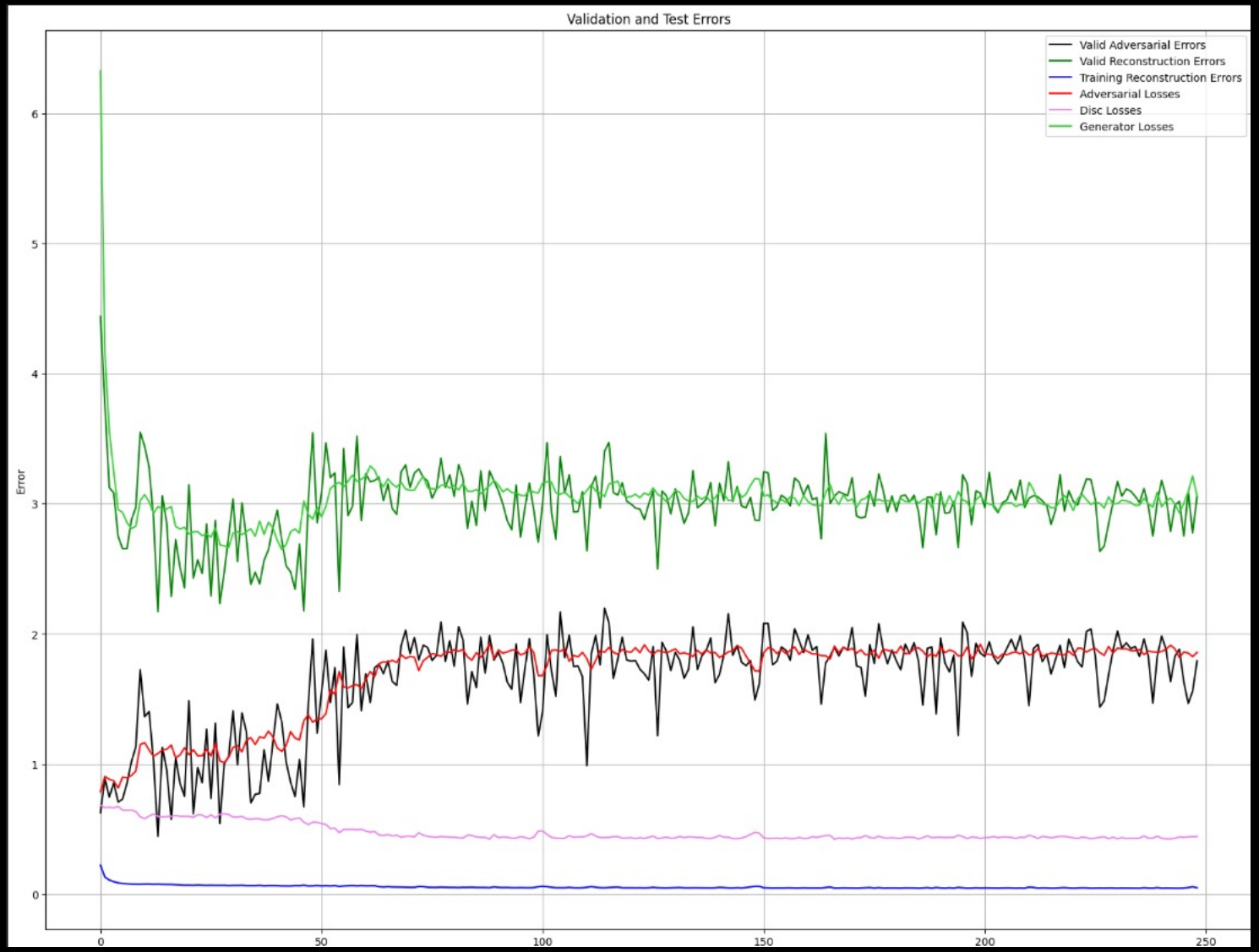
Evidence (1) Gaussian Noise:
- [1606.03498] Improved Techniques for Training GANs (arxiv.org)

Evidence (2) Add Noise to Label:
- Salimans et. al. 2016

Reference
1. https://www.reddit.com/r/MachineLearning/comments/oif5dp/d_adding_guassian_noise_to_discriminator_layers/

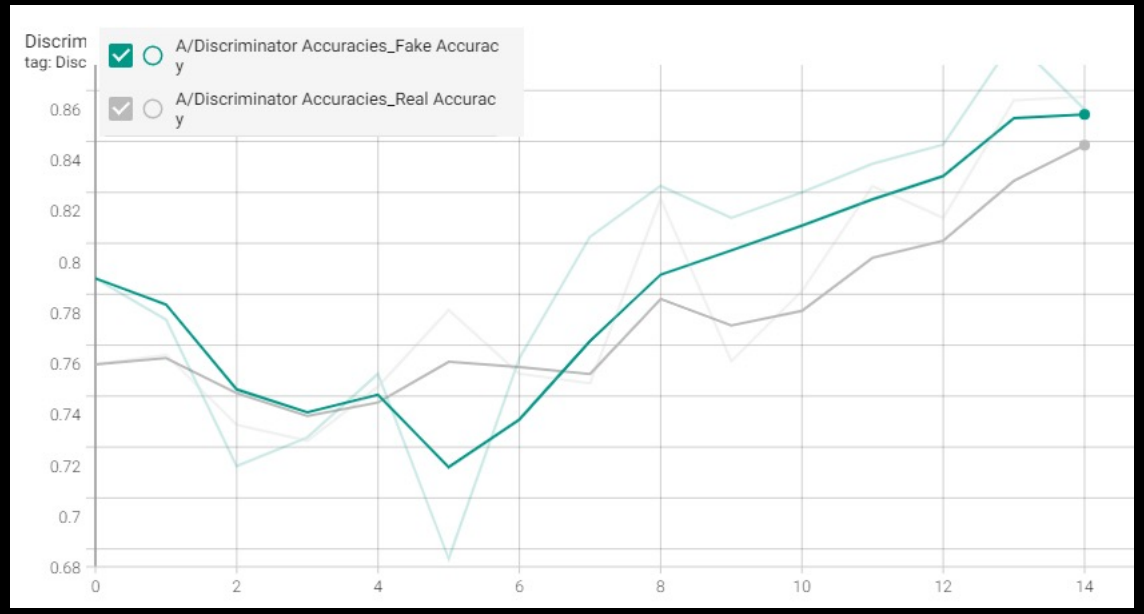
Results/Evaluation

Final Losses plot

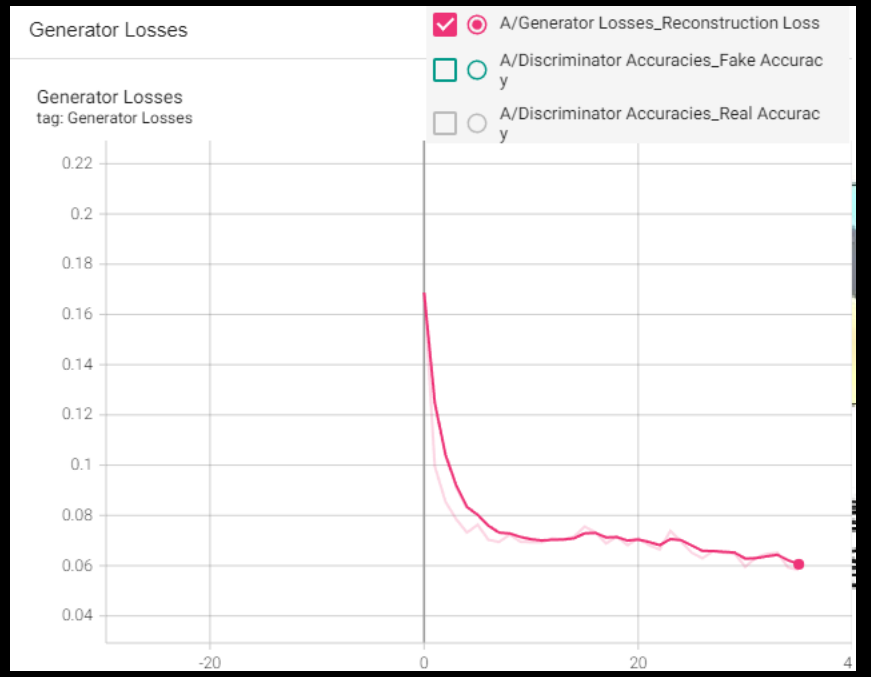


Results/Evaluation

Final Discriminator Loss Plot



Discriminator Real/Fake Accuracy Plot



Discriminator Validation Loss Plot

Results/Evaluation

Best Case

Results/Evaluation

Best Case

Results/Evaluation

Failure to perform well on a particular object

Results/Evaluation

Worst Case

Conclusion

Worst Case Cause and Resolution

PROBLEM & RESOLUTION

(1) Insufficient number of Data

(2) Custom Data Loader

(3) Gaussian Noise’s optimal Decay Rat

(4) Hyperparameter Assesment Evidence – FID, Inception

(5) Model collapse(Weight hyper parameter)

… And More