Image Recoloring with Conditional GAN

Neural Networks and Deep Learning

L. Borella

March 15, 2023

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Overview

Overview

Introduction

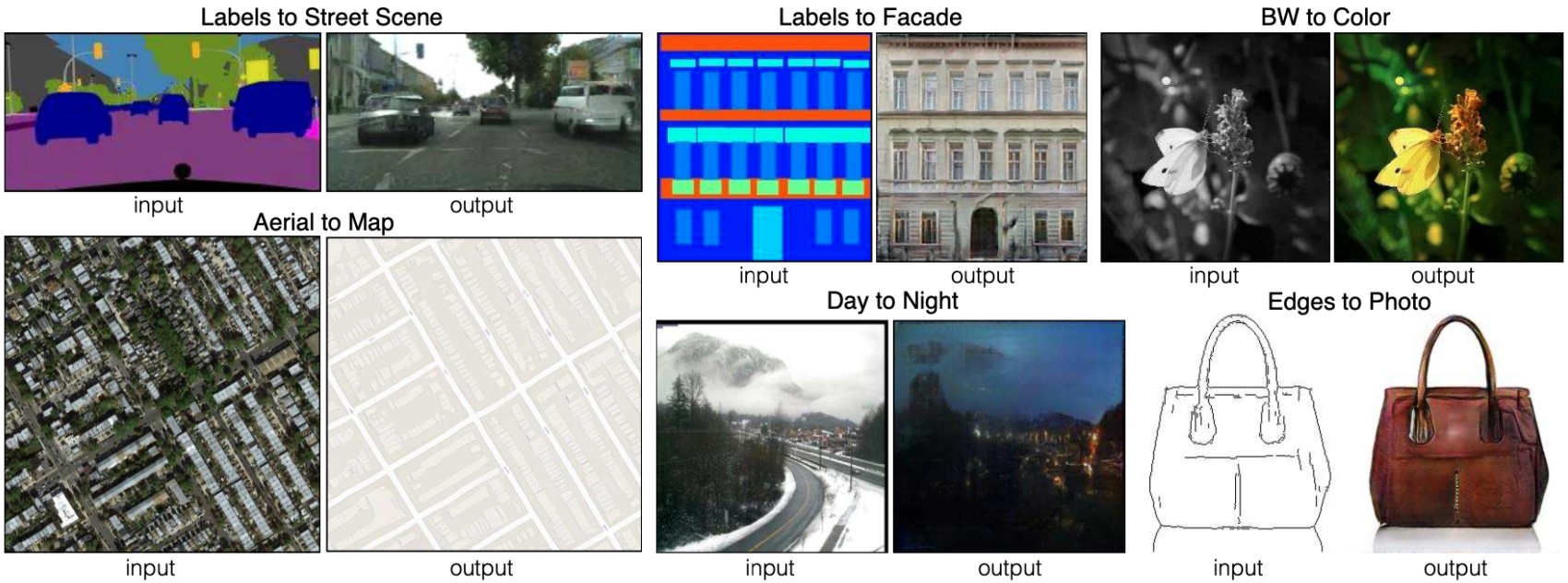


Figure: Image-to-image translation.

Figure: Image Recoloring Samples.

Generative Model

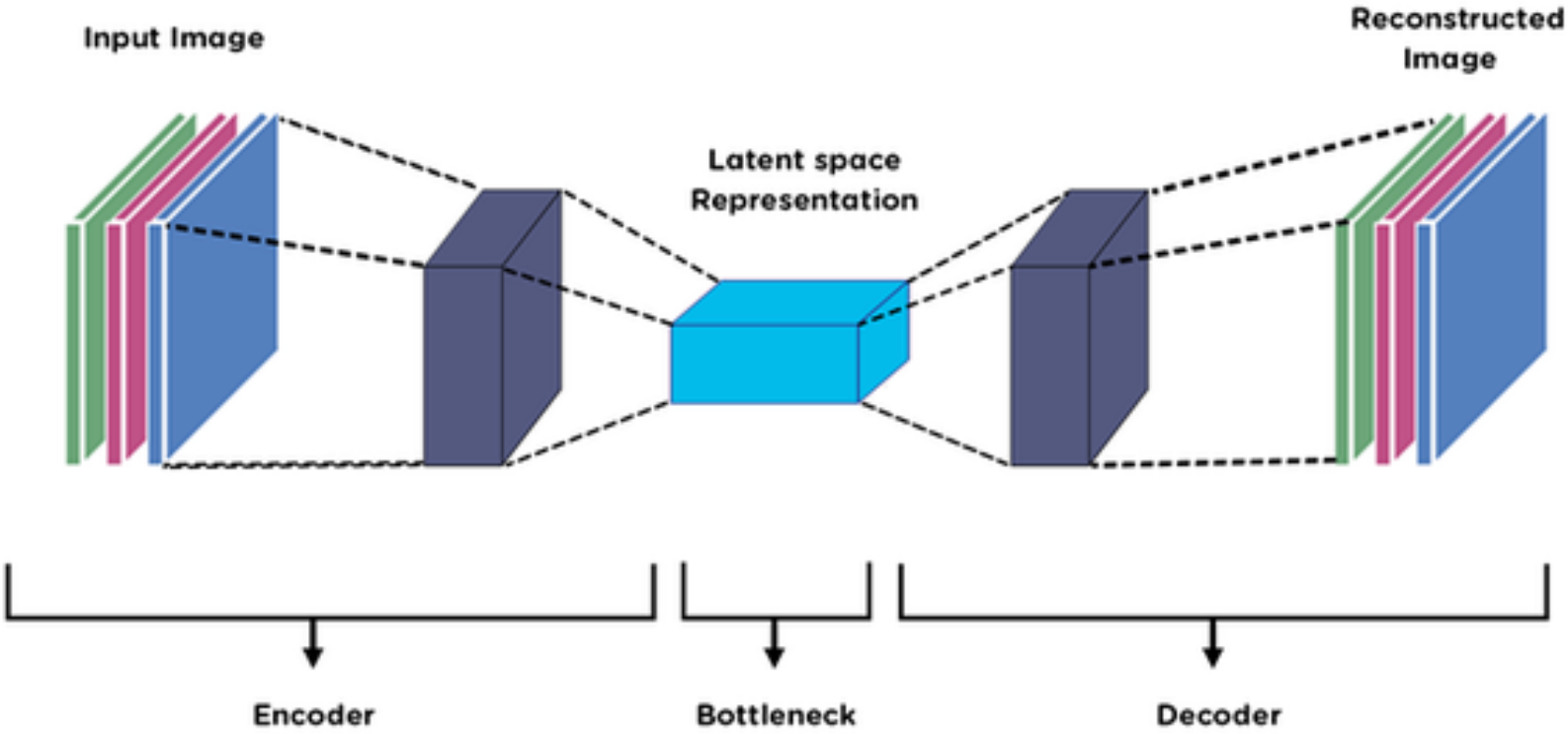


Figure: Autoencoder.

Discriminator

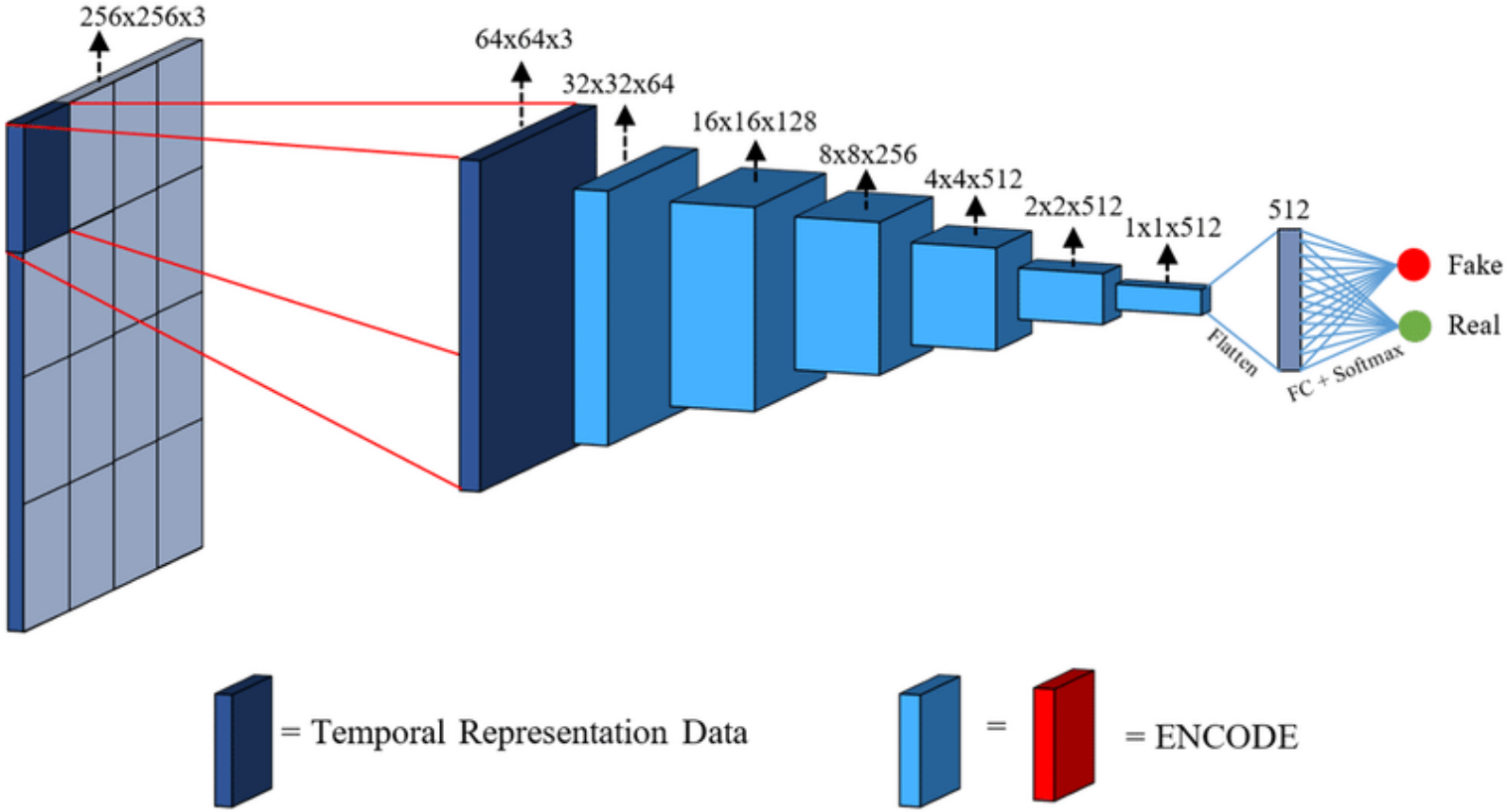


Figure: patch-GAN.

Conditional GAN

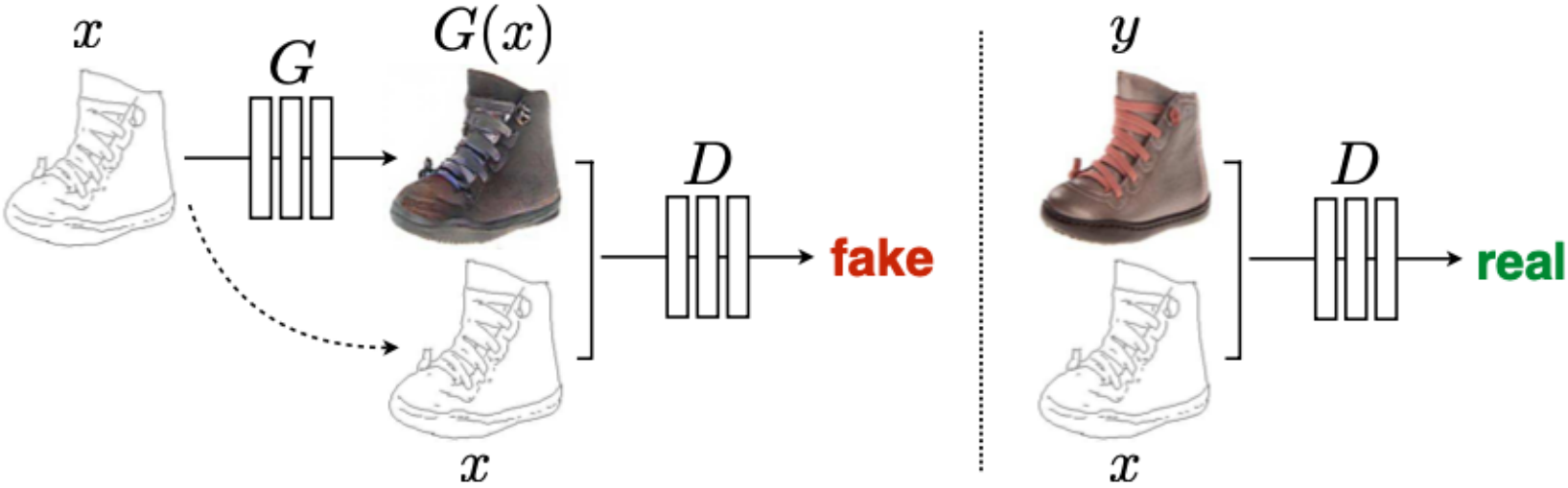
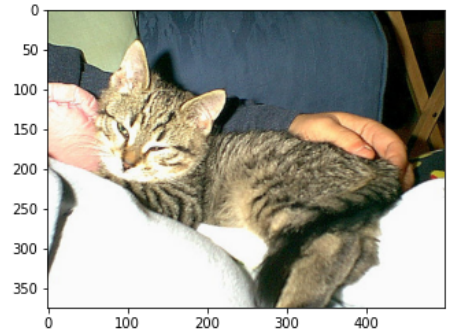
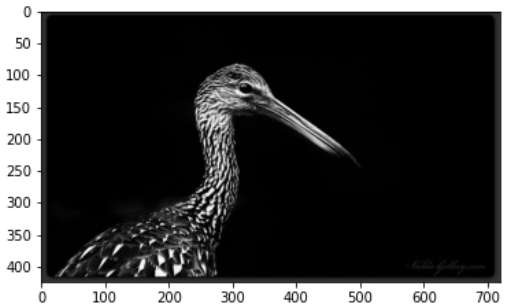


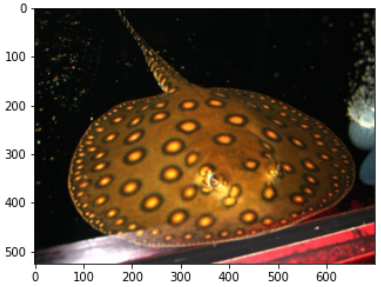
Figure: Conditional GAN.

Overview

Imagenet1000







L*a*b* Space

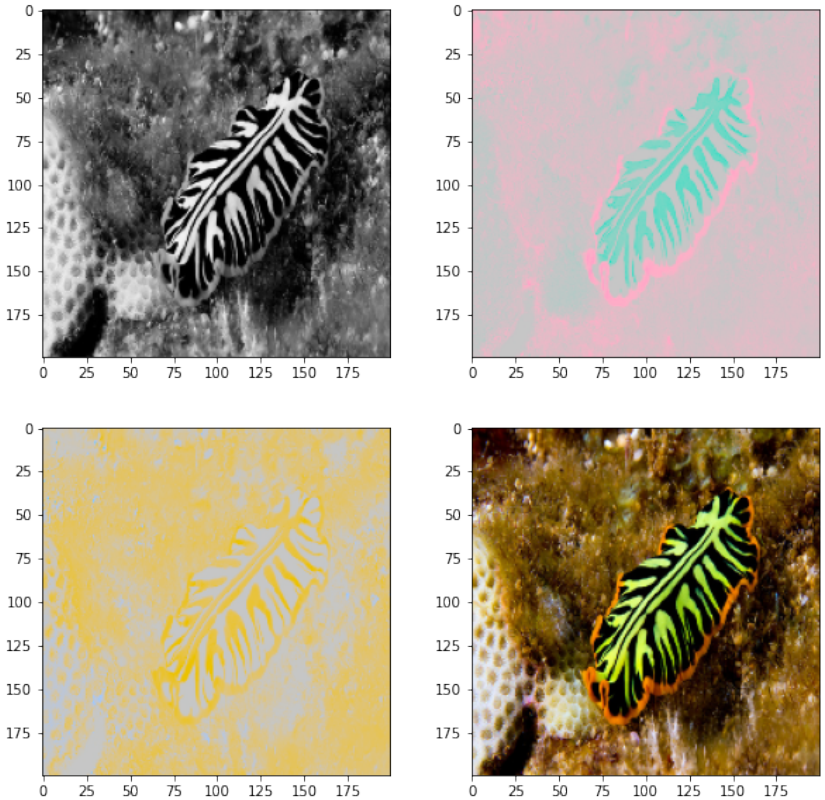


Figure: L*, a* and b* layers with RGB image.

Splitting

Set	Percentage	Samples
Train	80%	16340
Val	10%	2042
Test	10%	2044
Test	100%	20426

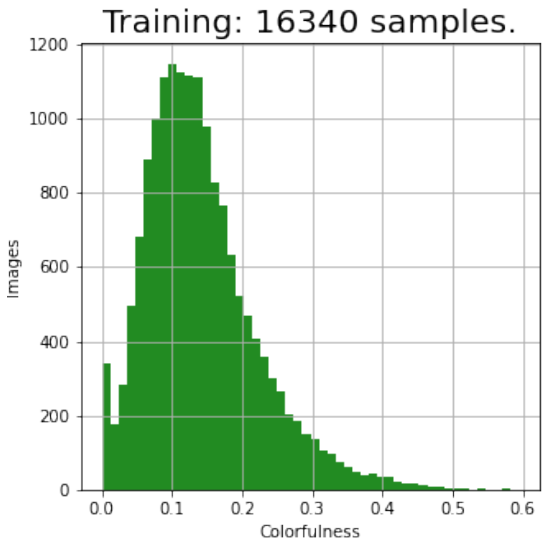
Table: Trainable Parameters.

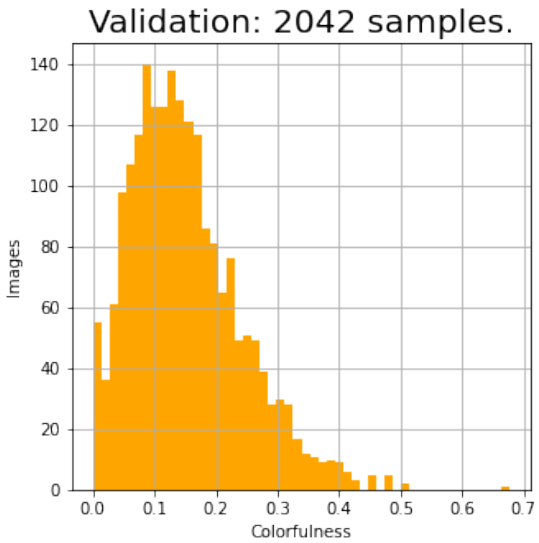
$$\mathbf{rg} = |R - G|, \mathbf{yb} = |\frac{R+G}{2} - B|$$

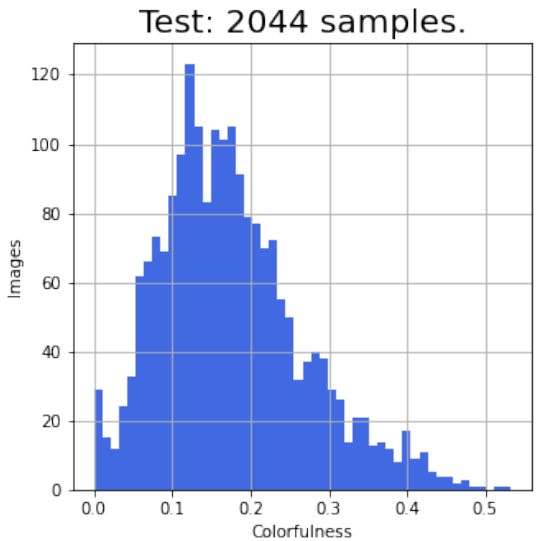
$$\mu^2 = \mu_{rg}^2 + \mu_{yb}^2, \sigma^2 = \sigma_{rg}^2 + \sigma_{yb}^2$$

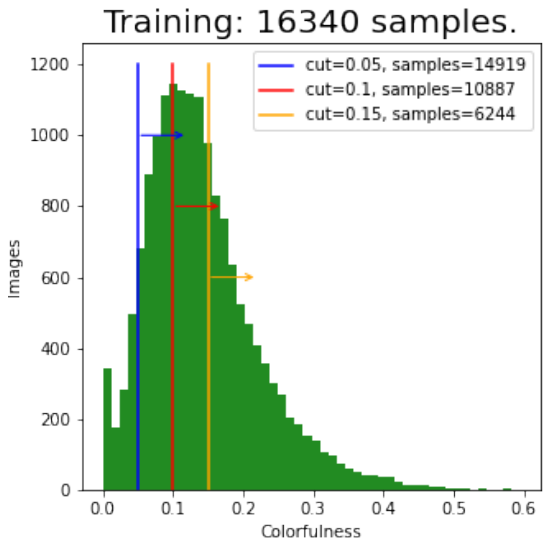
$$\mathbf{metric} = 0.3 * \mu + \sigma$$

Splitting


Training: 16340 samples.
Images
Colorfulness


Validation: 2042 samples.
Images
Colorfulness

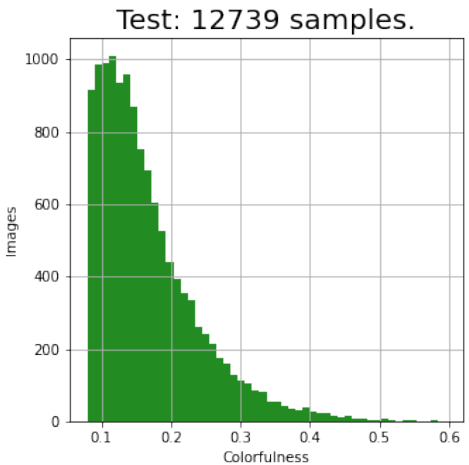

Test: 2044 samples.
Images
Colorfulness

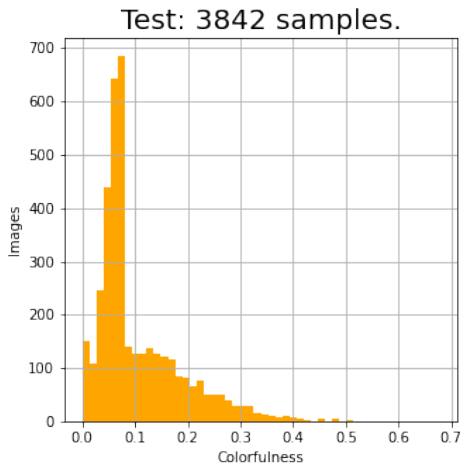

Training: 16340 samples.
cut=0.05, samples=14919
cut=0.1, samples=10887
cut=0.15, samples=6244
Images
Colorfulness

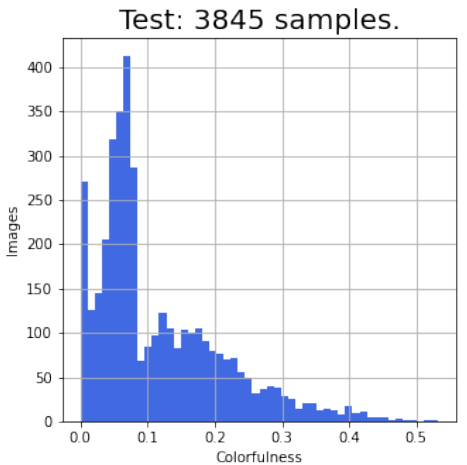
Splitting

Set	Percentage	Samples	Batch	Num
Train	60%	12739	128	100
Val	20%	3842	128	31
Test	20%	3845	128	31
Total	100%	20426		

Table: Data Splitting.

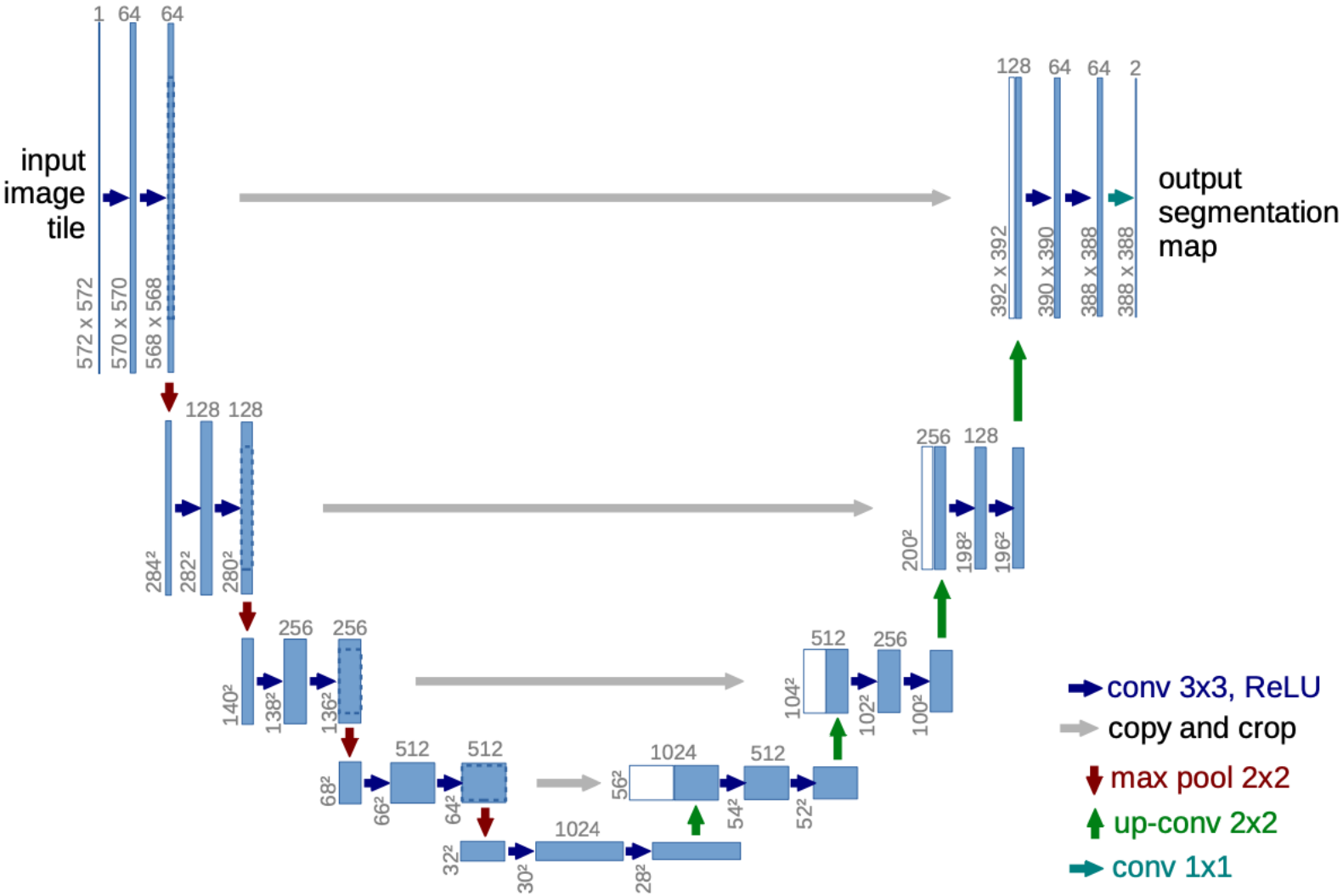






Overview

U-NET Architecture



Simple U-NET

Level	Block	In-Out Channels
1	Conv2d	(1,64)
1	Conv2d	(64, 64)
1	Conv2d	(64, 64)
2	Conv2d	(64,128)
2	Conv2d	(128,128)
2	Conv2d	(128,128)
1	ConvTranspose2d	(128,64)
1	Conv2d	(128,64)
1	Conv2d	(64, 32)
1	Conv2d	(32, 2)
	Total Parameters	570.825

TABLE 4: Simple UNET.

Normal U-NET

Level	Block	In-Out Channels
1	Conv2d	(1,64)
1	Conv2d	(64,64)
2	Conv2d	(64,128)
2	Conv2d	(128, 128)
3	Conv2d	(128,256)
3	Conv2d	(256,256)
3	Conv2d	(256,256)
2	ConvTranspose2d	(256,128)
2	Conv2d	(256,128)
2	Conv2d	(128, 64)
1	ConvTranspose2d	(64, 64)
1	Conv2d	(128,64)
1	Conv2d	(64, 2)
	Total Parameters	2.327.558

TABLE 5: Normal UNET.

Complex U-NET

Level	Block	In-Out Channels
1	Conv2d	(1,64)
2	Conv2d	(64,128)
3	Conv2d	(128,256)
4	Conv2d	(256, 512)
5	Conv2d	(512,512)
6	Conv2d	(512,512)
7	Conv2d	(512,512)
8	Conv2d	(512,512)
7	ConvTranspose2d	(1024,512)
6	ConvTranspose2d	(1024,512)
5	ConvTranspose2d	(1024,512)
4	ConvTranspose2d	(1024,256)
3	ConvTranspose2d	(512,128)
2	ConvTranspose2d	(256,64)
1	ConvTranspose2d	(128,2)
	Total Parameters	54.409.858

TABLE 7: Complex UNET.

Discriminator

Blocks	Ch-In	Ch-Out
Conv2d	4	32
Conv2d	32	64
Conv2d	64	128
Linear	var	1
	Params	140.129

Table: Simple Architecture.

Blocks	Ch-In	Ch-Out
Conv2d	4	32
Conv2d	32	64
Conv2d	64	128
Conv2d	128	256
Conv2d	256	256
Linear	var	1
	Params	1.431.841

Table: Normal Architecture.

Complex Discriminator

Level	Block	In-Out Channels
1	Conv2d	(3,64)
1	Conv2d	(64, 128)
1	Conv2d	(128, 256)
1	Conv2d	(256,512)
1	Conv2d	(512,1)
	Total Parameters	2.765.633

TABLE 6: Complex PatchGAN.

Before Training

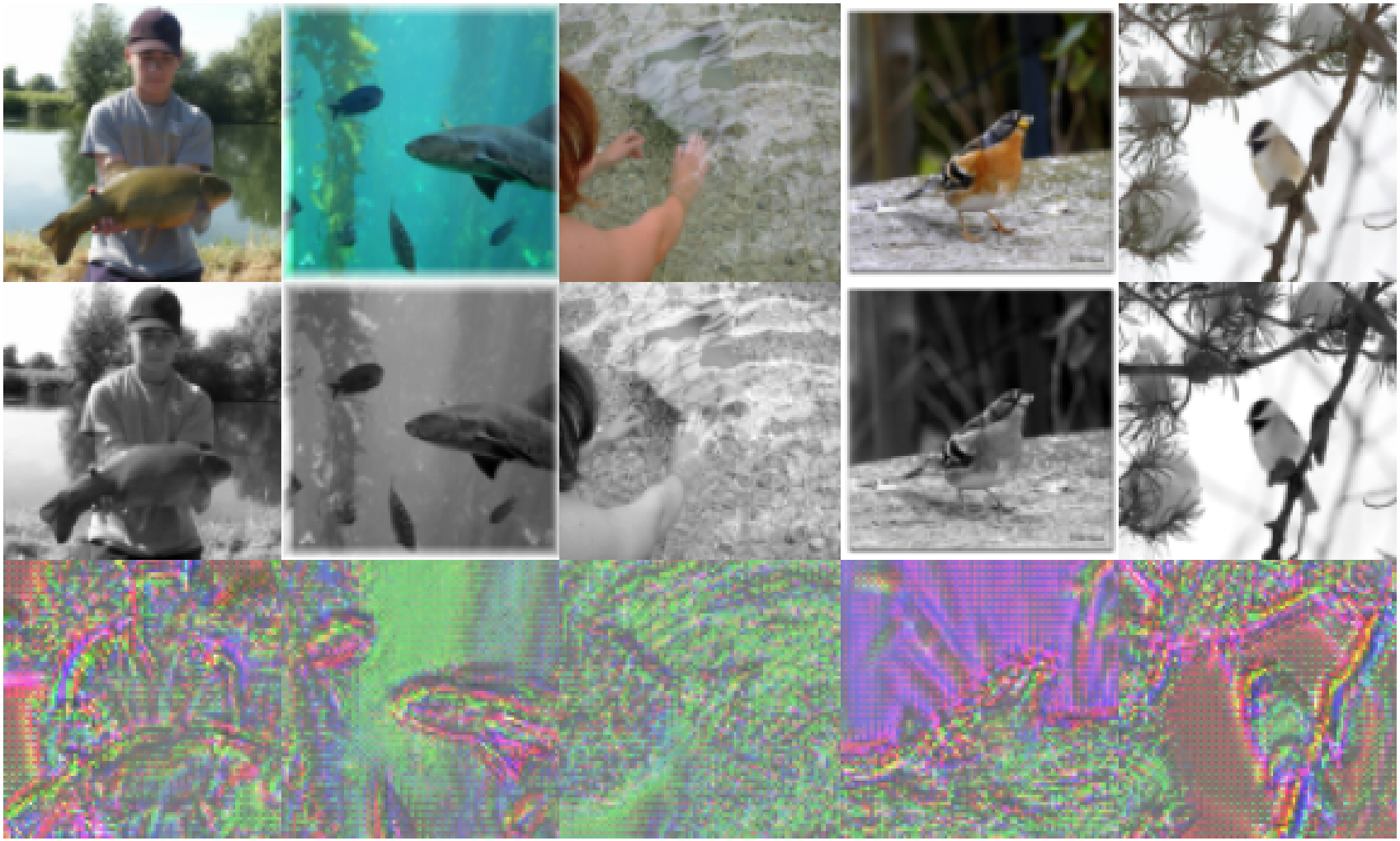


Figure: Simple Architecture.

Before Training

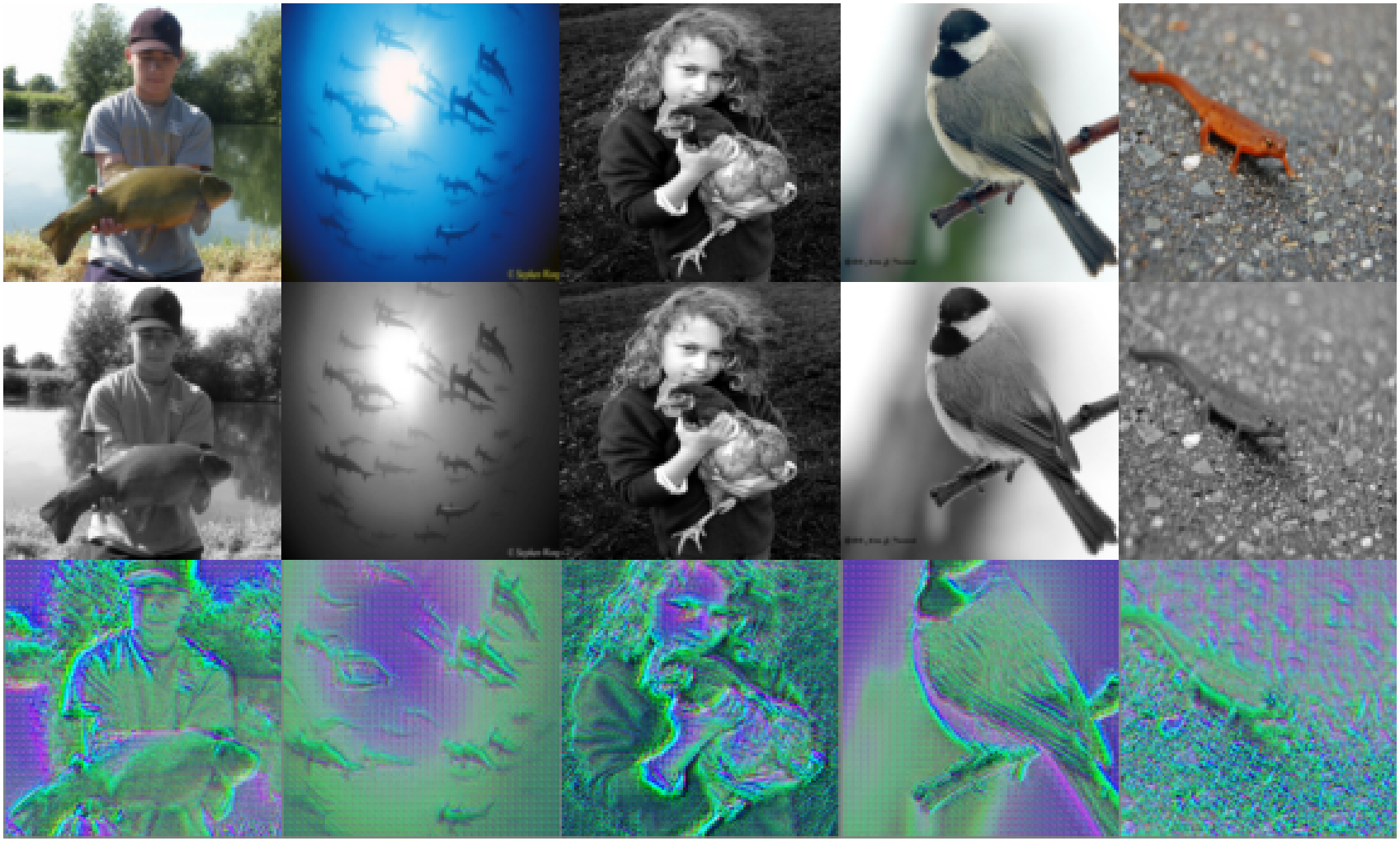


Figure: Complex Architecture.

Loss function

- Objective Function:

$$\mathcal{L}_{CGAN}(G, D) = \log D(x, y) + \log(1 - D(x, G(x, z)))$$

- Maximize for D, minimize for G.
- Add L_1 regularization term: $\mathcal{L}_{L_1}(G) = ||y - G(x, z)||_1$.

$$G^* = arg \min_G \max_D \mathcal{L}_C GAN + \lambda \mathcal{L}_{L_1}(G)$$

- Metrics: `nn.BCEWithLogitsLoss()`

$$H(p, q) = -\sum_{x \in X} p(x) \log q(x)$$

Learning Algorithm

Training process:

1. Iterate over number of epochs.
2. Process data batch-per-batch.
3. Feed **Real Data** to the **Discriminator** and update its weights.
4. Generate **Fake Images**, feed them to the **Discriminator** and update its weights.
5. Train the **Generator**. Compute its error based on the evaluation of the updated Discriminator.

WGAN

Main differences with previous models:

1. Discriminator outputs values in [-1,1] range instead of [0,1].
2. Exploits Wasserstein Loss (y= labels):
   $W_{loss} = \frac{1}{n} \sum_0^n y_{n,true} * y_{n,gen}$
3. Discriminator becomes Critic.
4. Uses `RMSProp` as optimizer.
5. Clipping gradients to avoid exploding values.

The major drawback is a huge computational time.

Overview

Simple Architecture: Adam

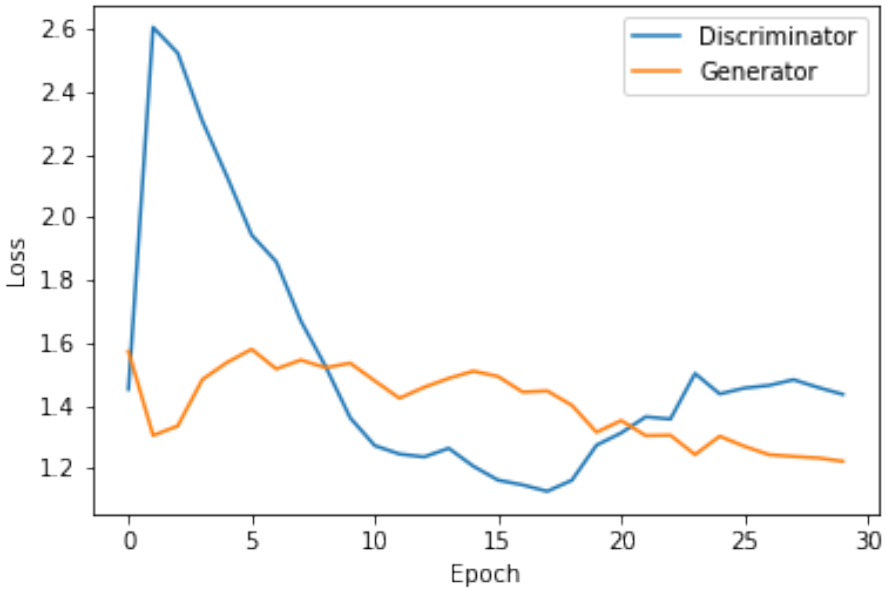


Figure: Discriminator and Generator's loss.

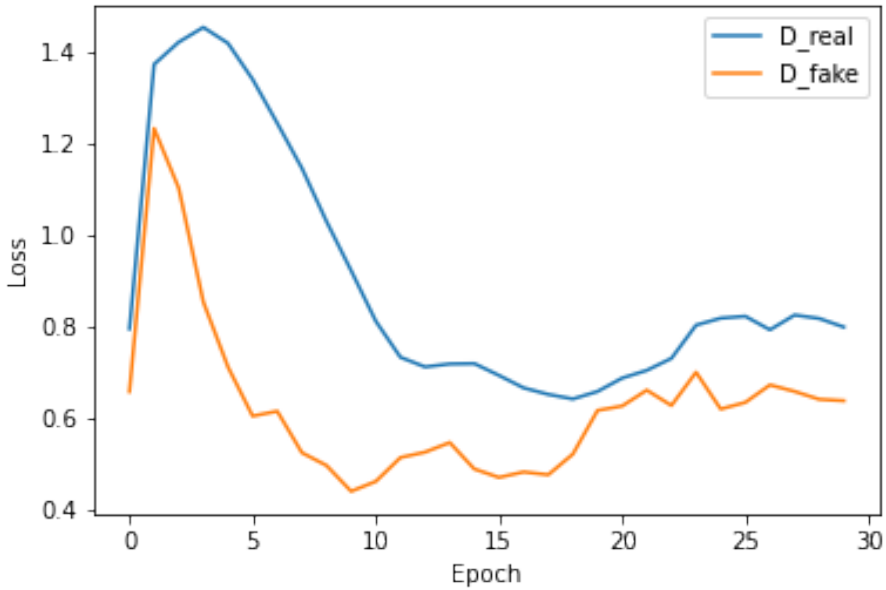


Figure: Real and Fake Discriminator loss.

Simple Architecture: SGD

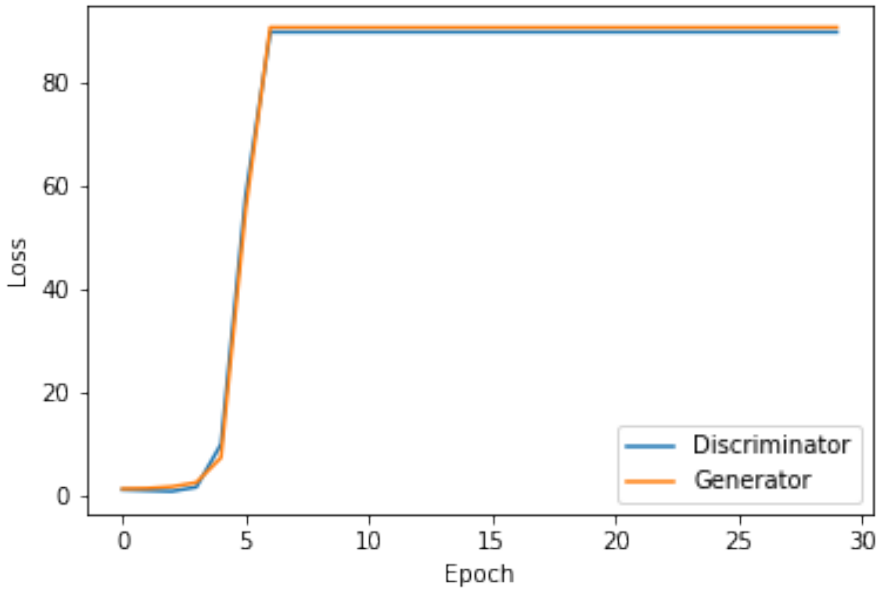


Figure: Discriminator and Generator's loss.

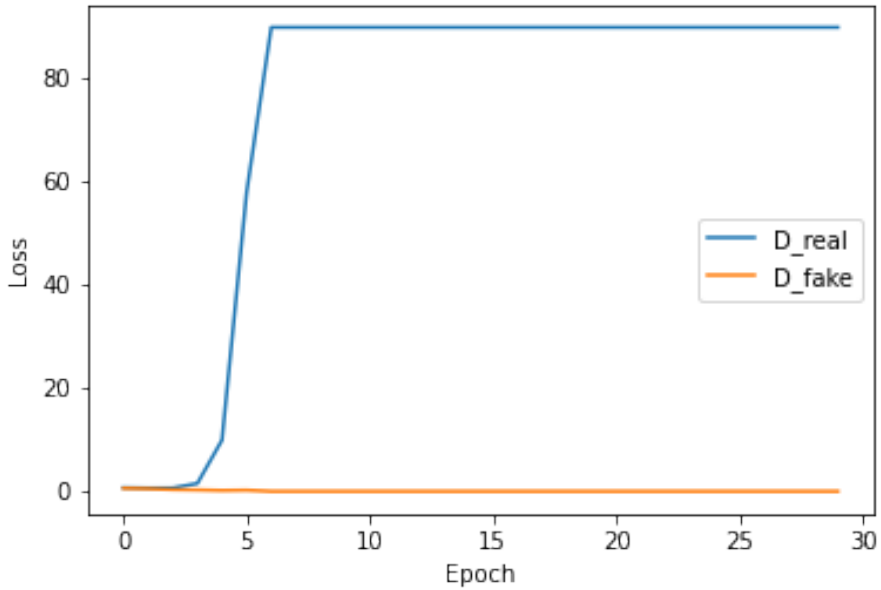


Figure: Real and Fake Discriminator loss.

Normal Architecture

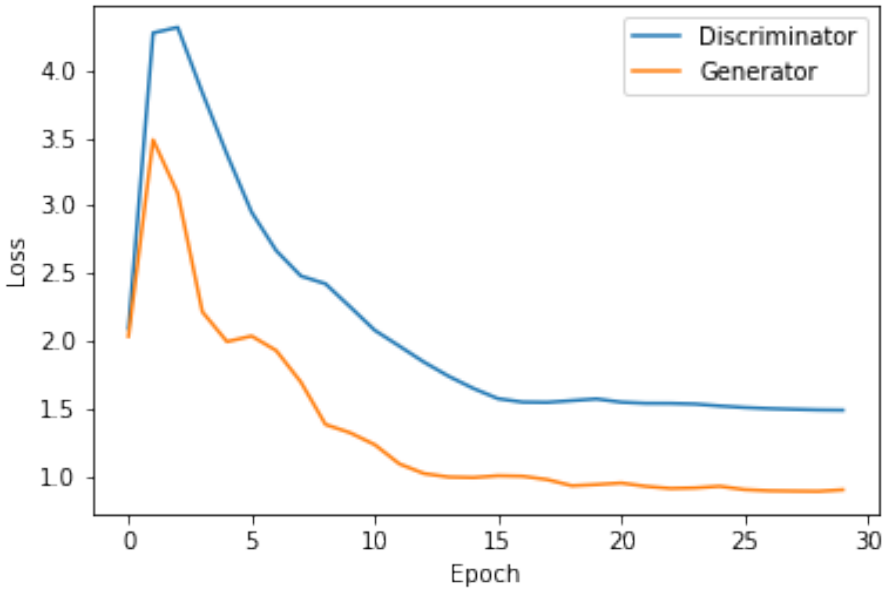


Figure: Discriminator and Generator's loss.

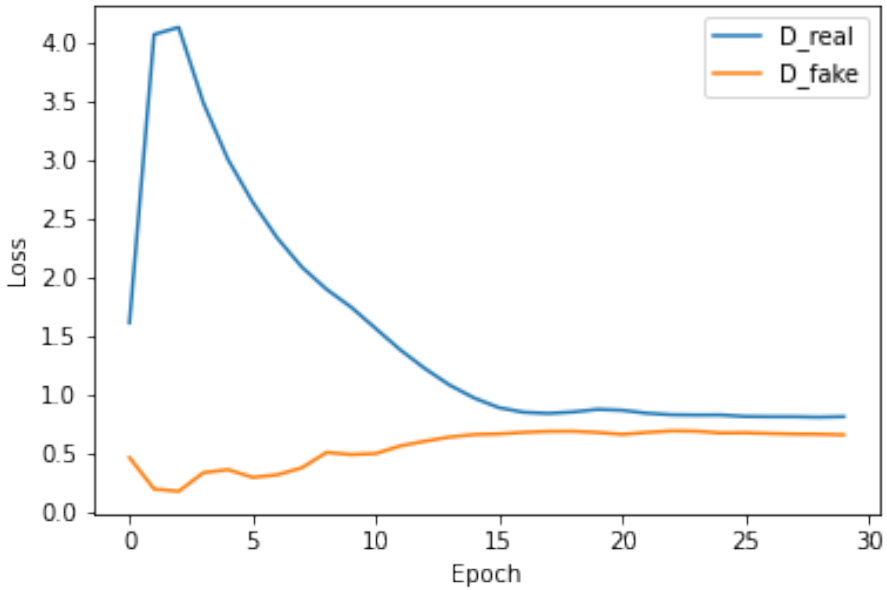


Figure: Real and Fake Discriminator loss.

Image Generation

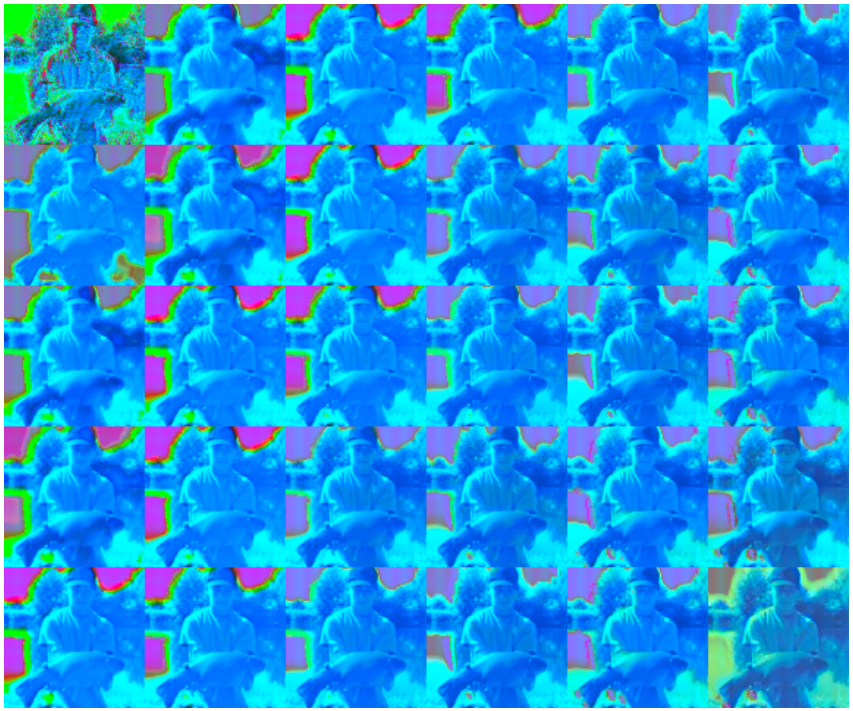


Figure: Simple Architecture

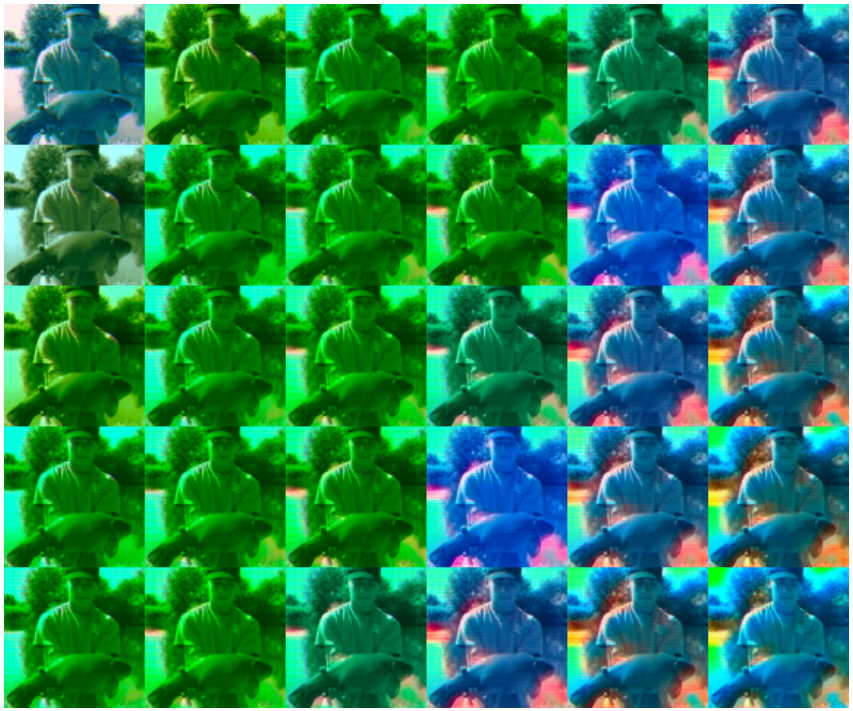


Figure: Normal Architecture.

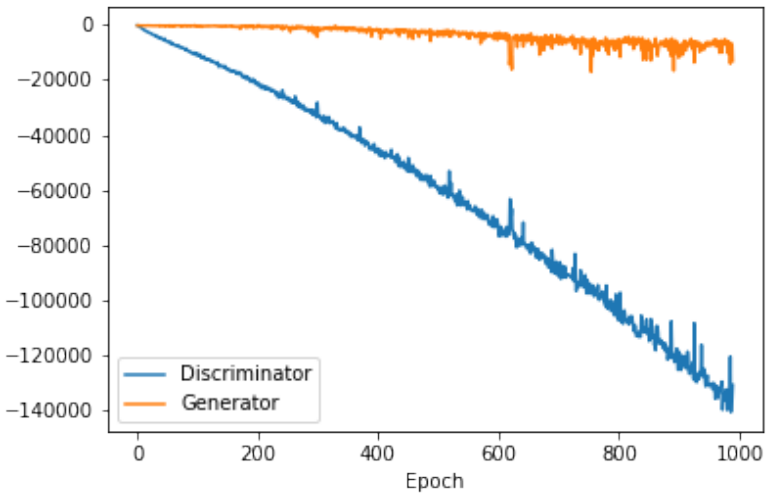


Figure: Learning curves for WGAN Training.

Figure: Generated Images.

Best Model: Complex Architecture

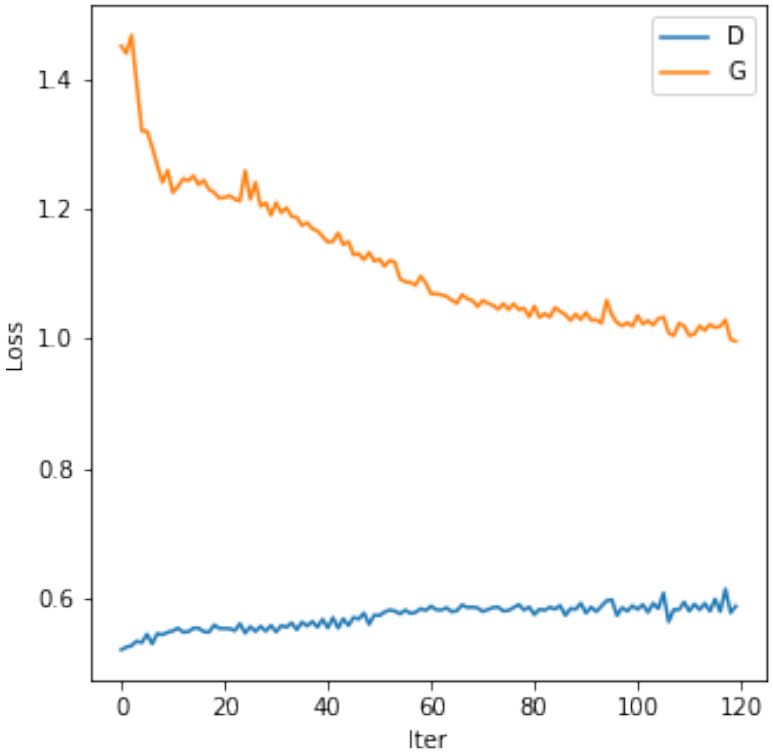


Figure: Discriminator and Generator loss.

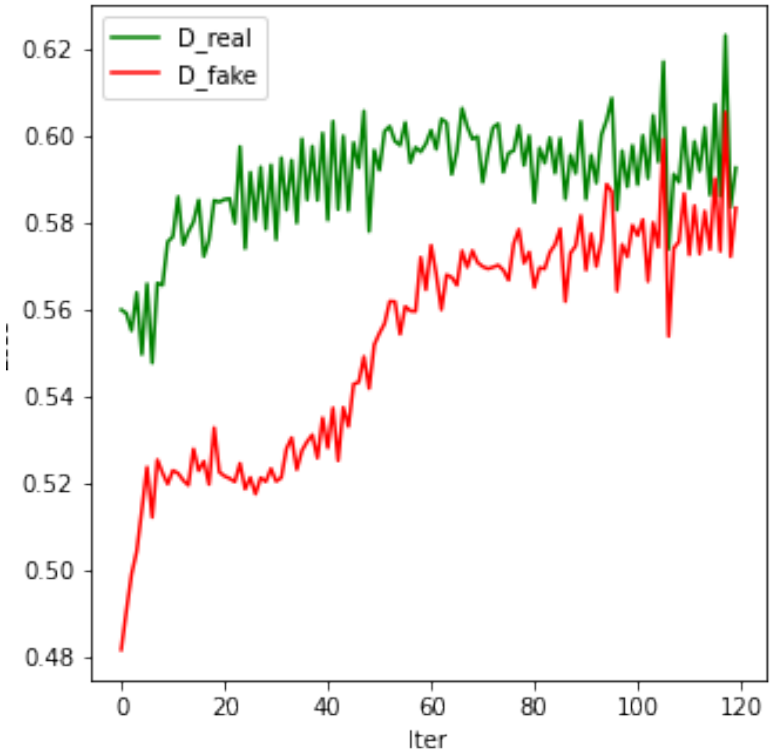


Figure: Real and Fake loss.

Best Model: Complex Architecture

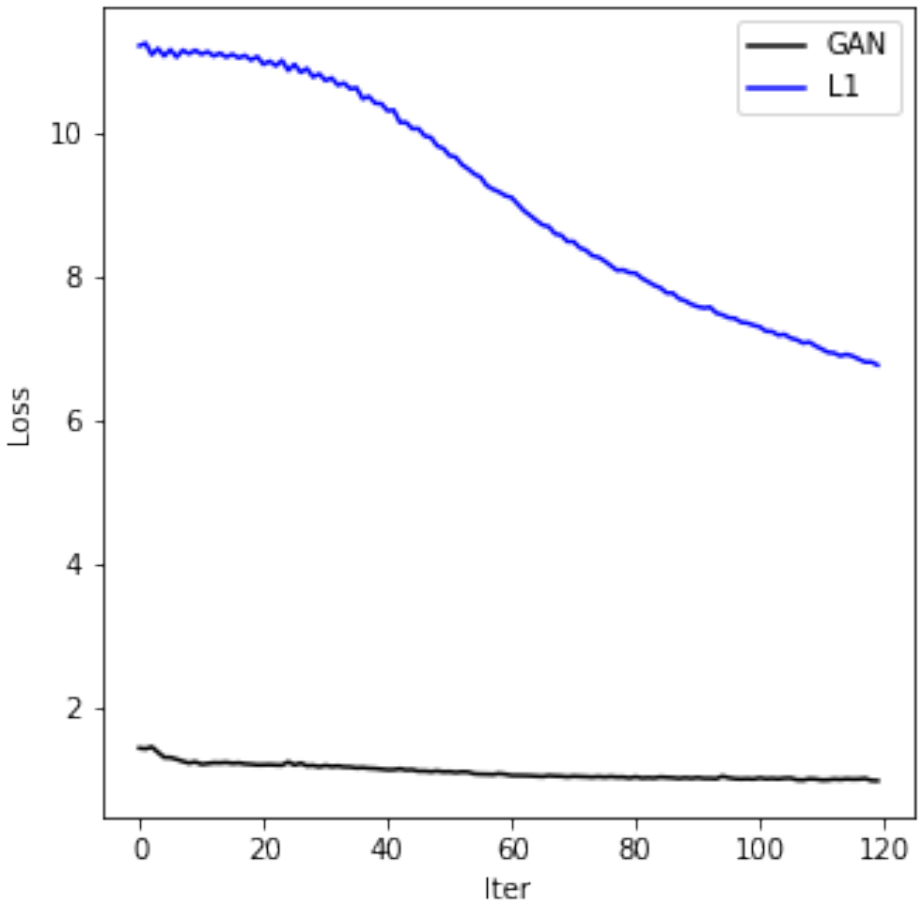
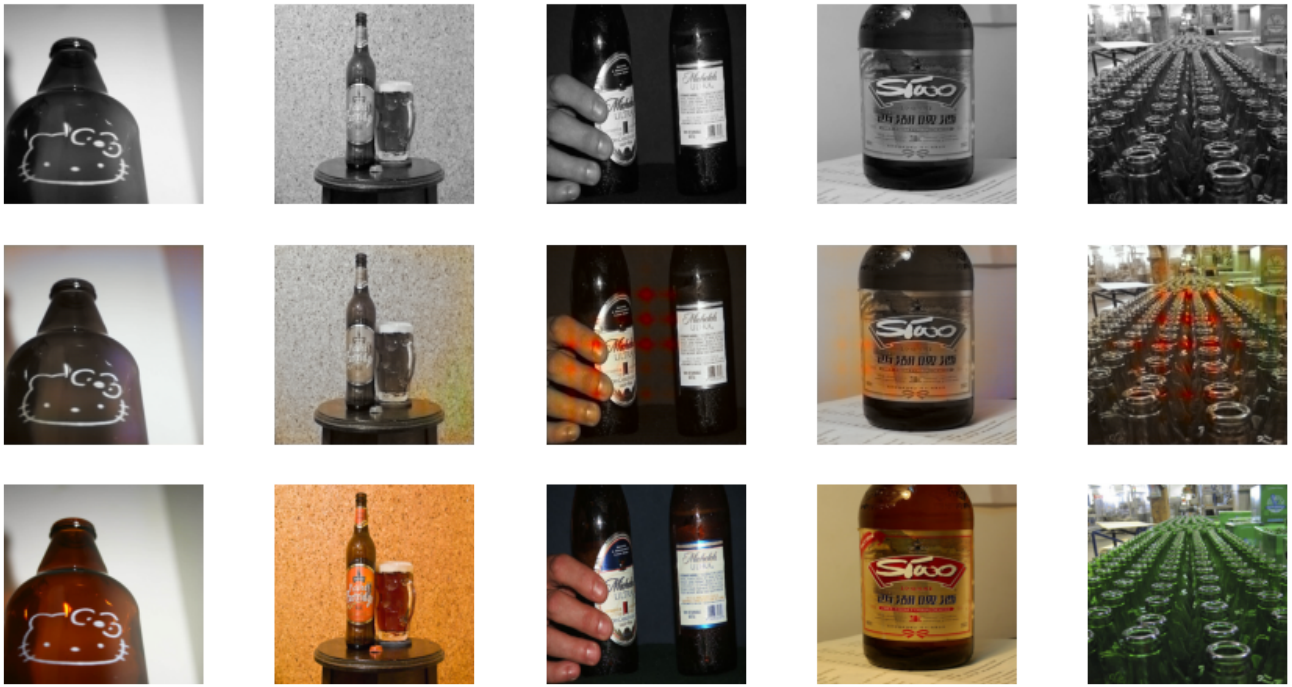


Figure: Generator L_1 and GAN losses.

Generated Images:start of Training.

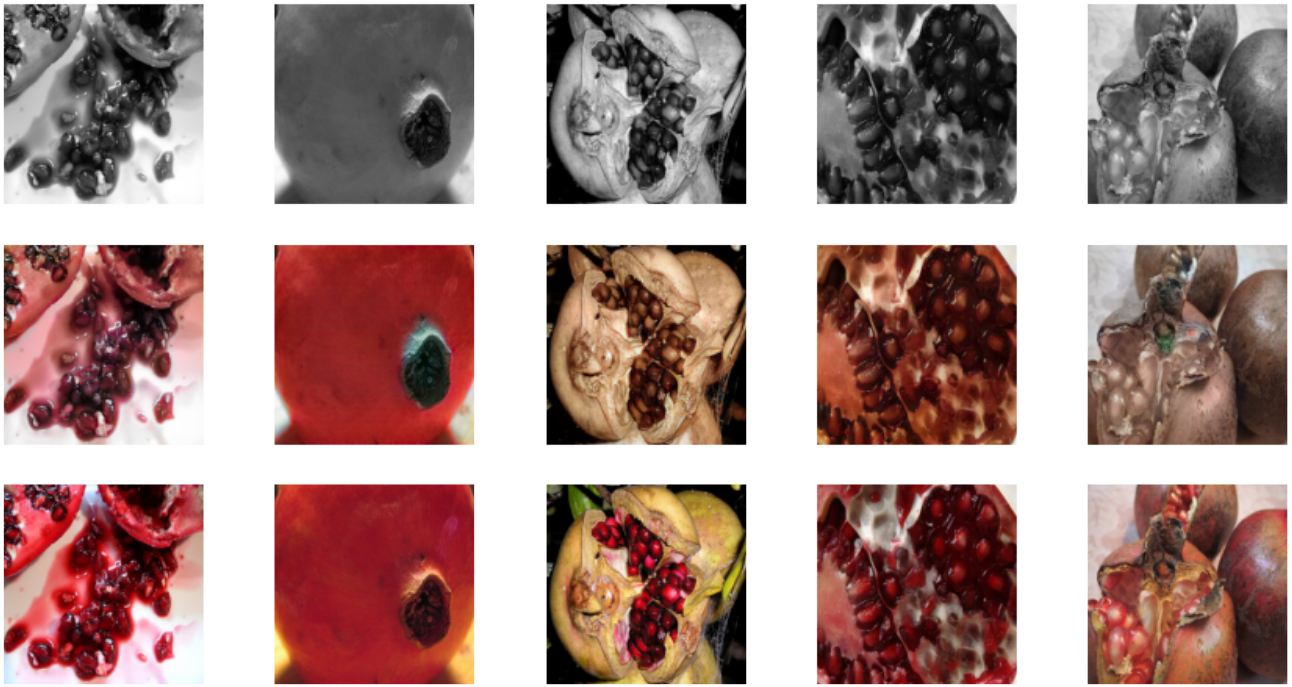


Generated Images:end of Training.

Test Images

Test Images

Test Images



Test Images