CYCLE GAN ON CARTOON SERIES

Cycle GAN

It successfully solves the tasks of domain adaptation and pix2pix transformation.

Does not require paired correspondence of objects from the two domains.

It trains relatively slowly (6–18 hours).

Requires a lot of memory (2 RTX 3090s).

Input

Output

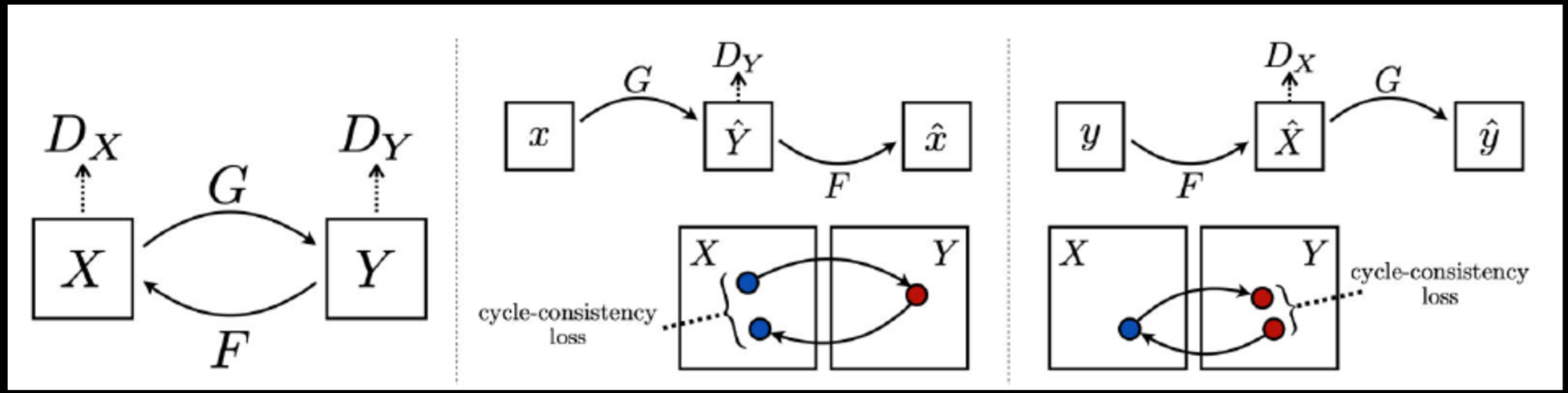
Baseline Architecture

- Original CycleGAN implementation with standard parameters.

- There are good examples, but the style transfer quality is generally low.

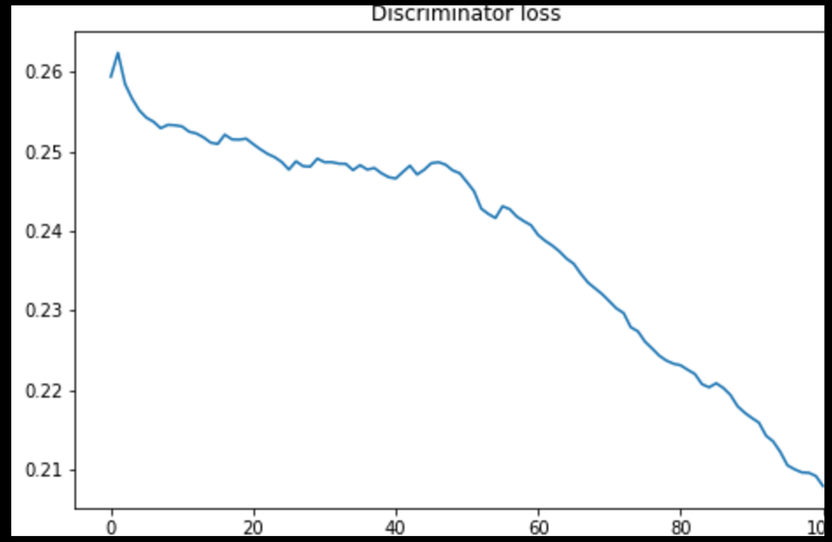
- Issues with discriminator overfitting.

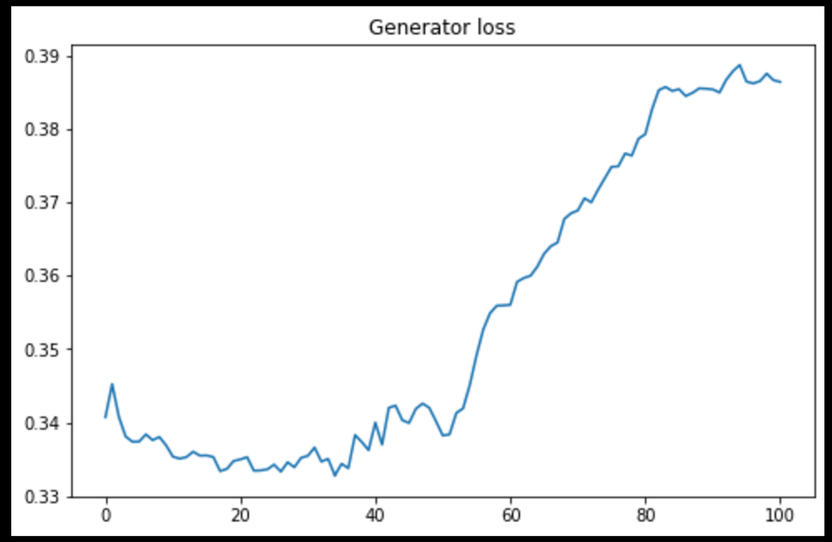
- The model weakly alters the original image


D_X
G
D_Y
X
Y
F
D_Y
G
x
$\hat{Y}$
$\hat{x}$
F
X
Y
cycle-consistency loss
D_X
G
y
$\hat{X}$
$\hat{y}$
F
X
Y
cycle-consistency loss

Discriminator Overfitting

When training the standard architecture, the generator's loss increases while the discriminator's loss decreases.





Solution Options :

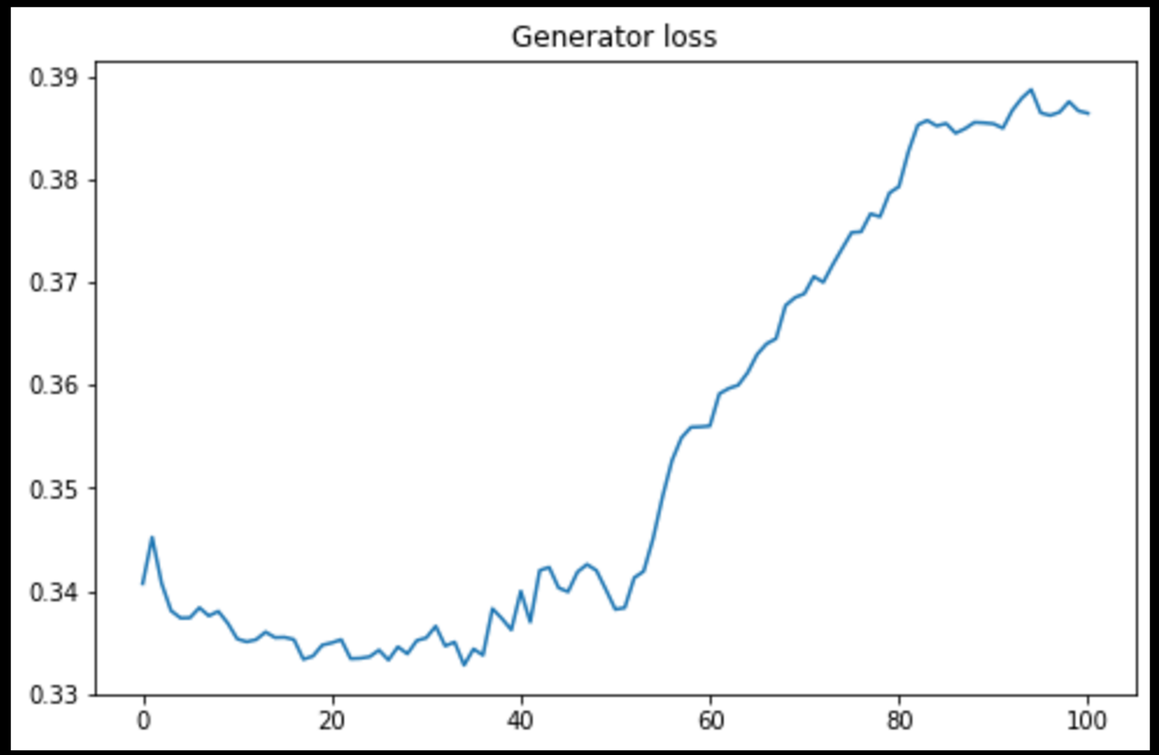
Adding noise to the discriminator's input to make its task harder.

Wasserstein loss (often used in the form of WGANs).

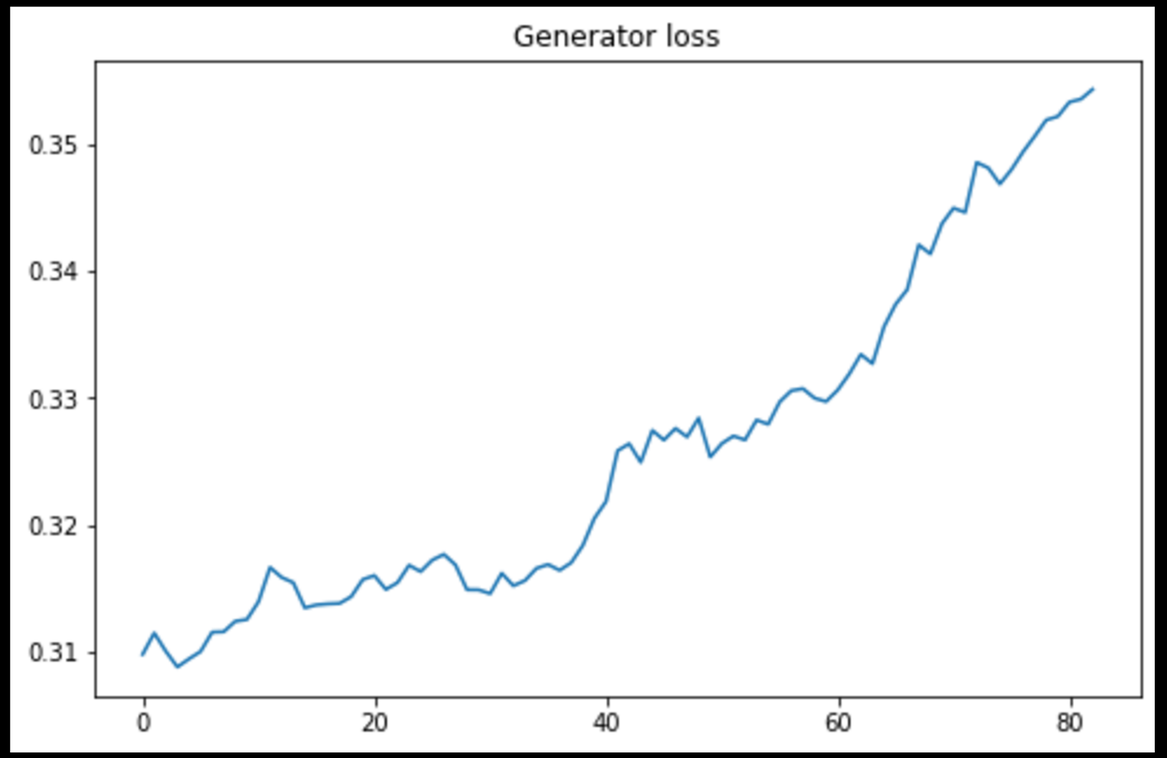
Cross-entropy loss.

Discriminator Overfitting

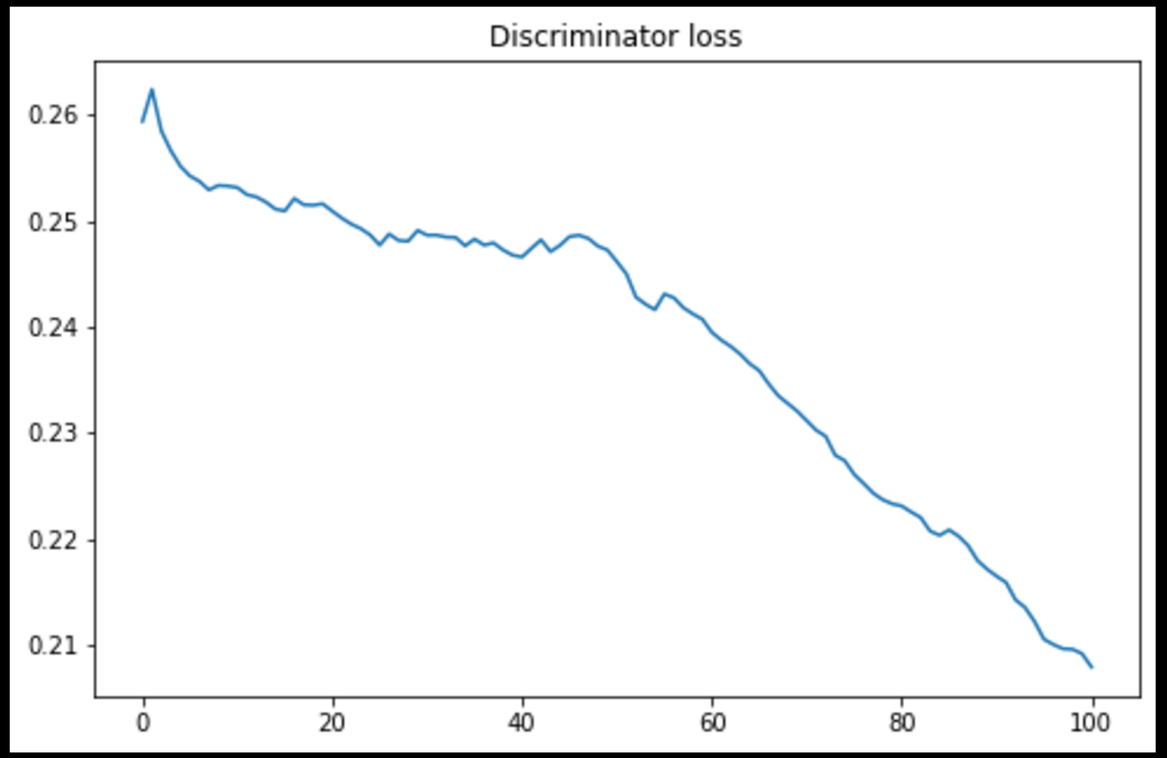
Base



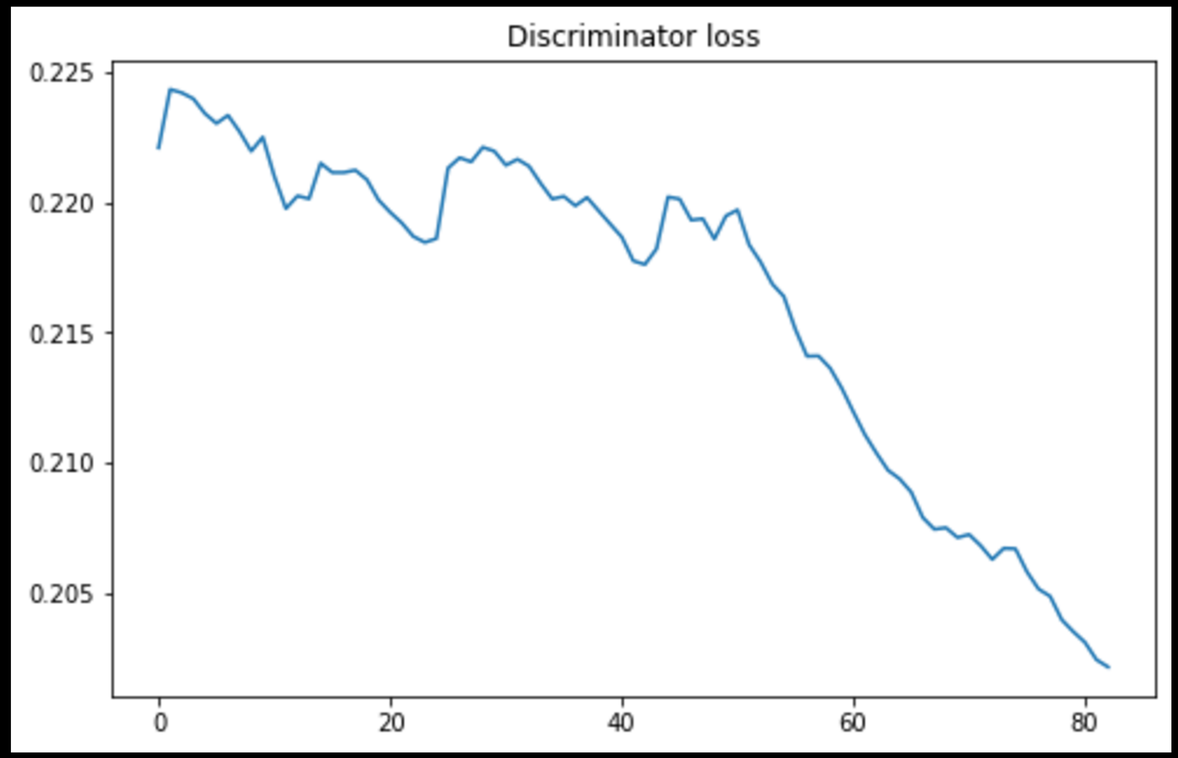
Gaussian noise std=0.1



Base



Gaussian noise std=0.1



Discriminator Overfitting

When training the standard architecture, the generator's loss increases while the discriminator's loss decreases.

Solution Options (or Potential Fixes):

Adding noise to the discriminator's input to make its task harder – the generator's loss decreased, but the core problem (overfitting) remained.

Wasserstein loss – could not be successfully tuned (or implemented) for this specific task.

Cross-entropy loss – applied to subsequent models.

VGG identity loss

- Idea taken from the CartoonGAN paper.

- Used a VGG with three convolutional layers.

- Applied VGG loss (perceptual loss) to both the new and old architectures.

```
In [245]: model

Out[245]: VGG(
            (features): Sequential(
              (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (2): ReLU(inplace=True)
              (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (5): ReLU(inplace=True)
              (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
              (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (9): ReLU(inplace=True)
            )
            (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
            (classifier): Sequential()
          )
```

$$\mathcal{L}_{con}(G, D) = \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]$$

Original Image
(or Source Image)

New Architecture

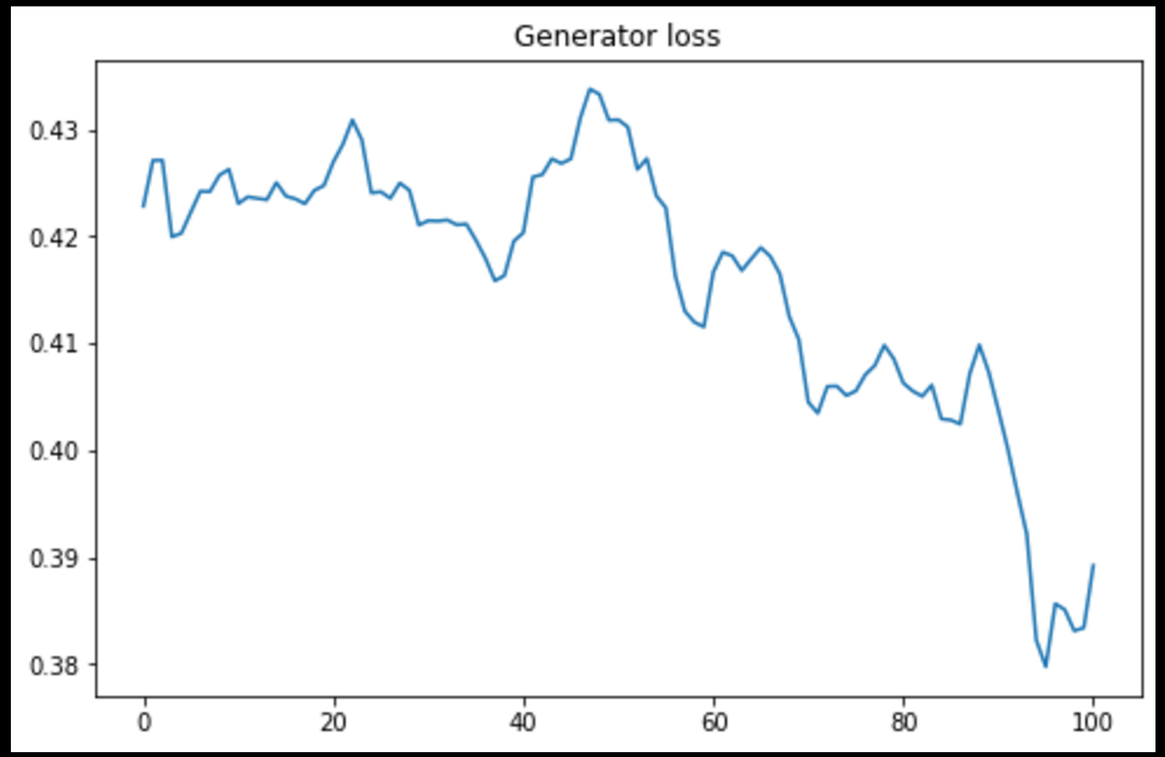
Original Architecture
(or Baseline Architecture)

Original Image
(or Source Image)

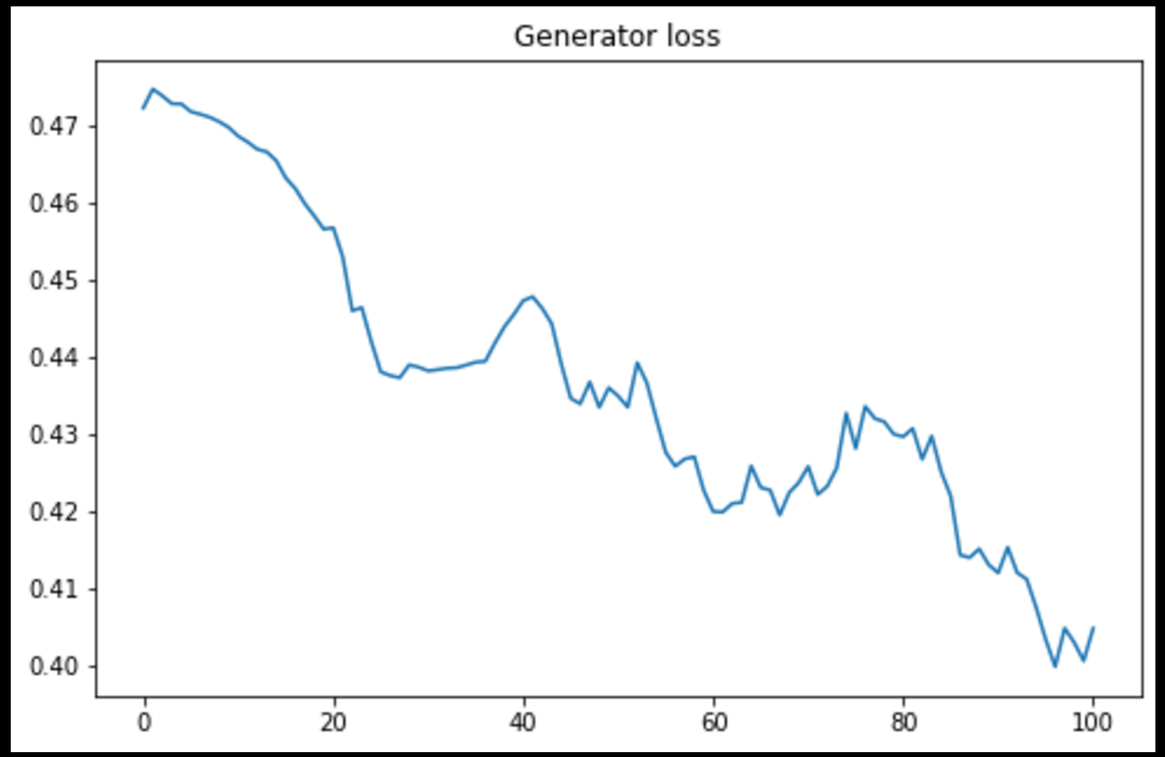
New Architecture

Original Architecture
(or Baseline Architecture)

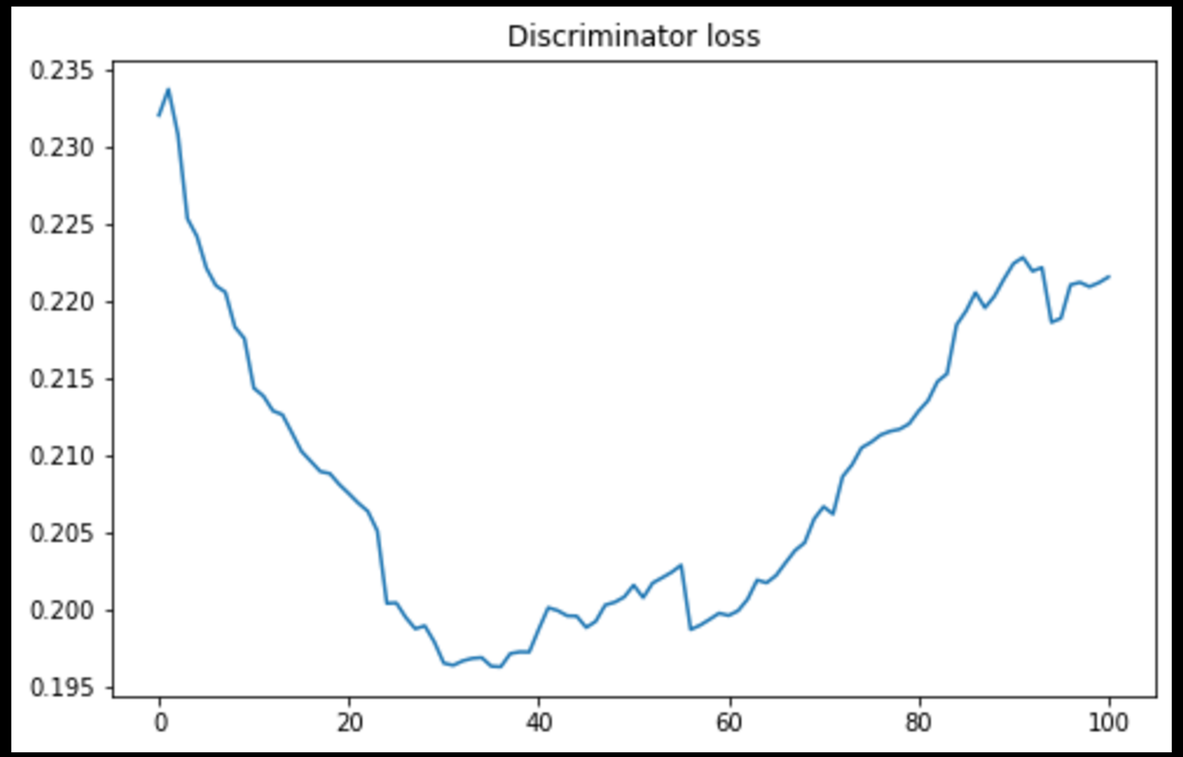
Original Architecture
(or Baseline Architecture)



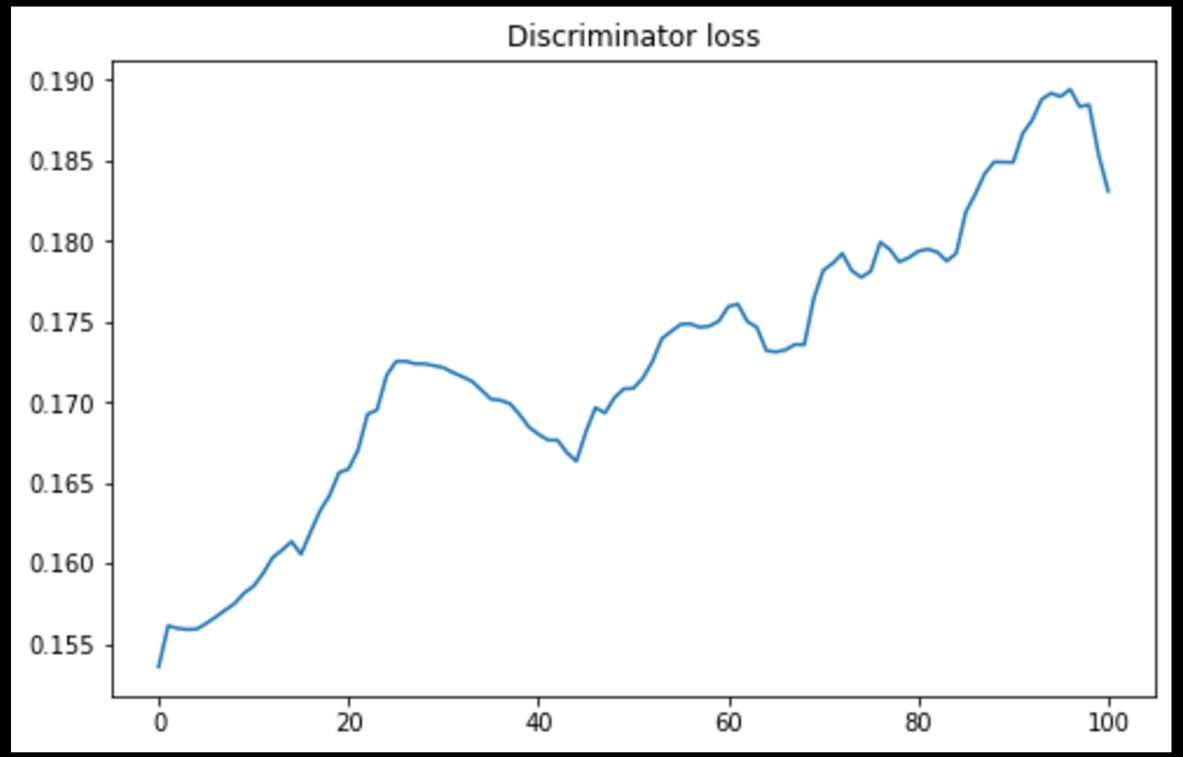
New Architecture



Original Architecture
(or Baseline Architecture)



New Architecture



VGG identity loss

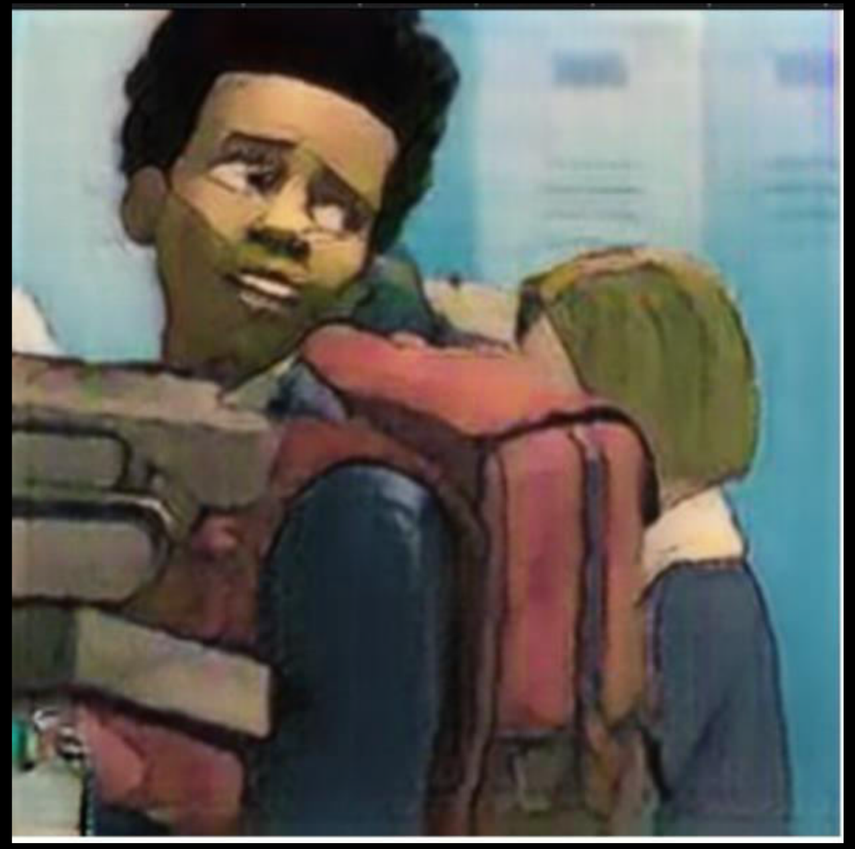
- MSE loss (Mean Squared Error) limited the style transfer, so we switched to Cross-Entropy loss.

- Training was less stable; the discriminator loss sometimes dropped to zero.

- Then we continued training the network with MSE; ultimately, the loss situation improved.

Original Architecture
(or Baseline Architecture)

New Architecture



Original Architecture
(or Baseline Architecture)

New Architecture

Original Architecture
(or Baseline Architecture)

New Architecture