Cycle GAN on cartoon series

Михаил Захватаев

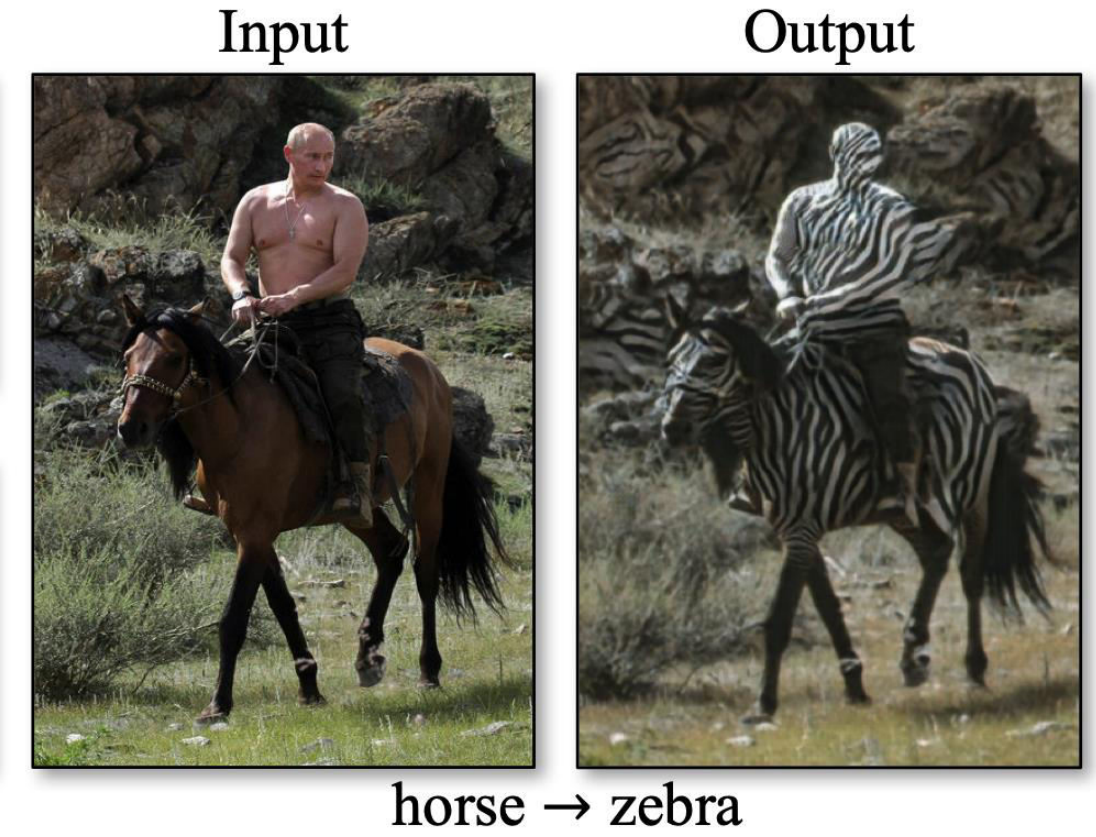
Михаил Коновалов

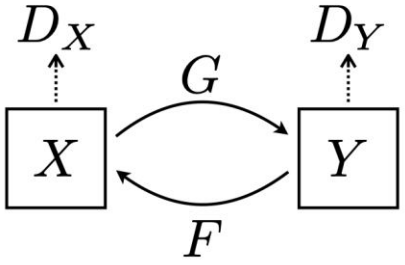
Михаил Смирнов

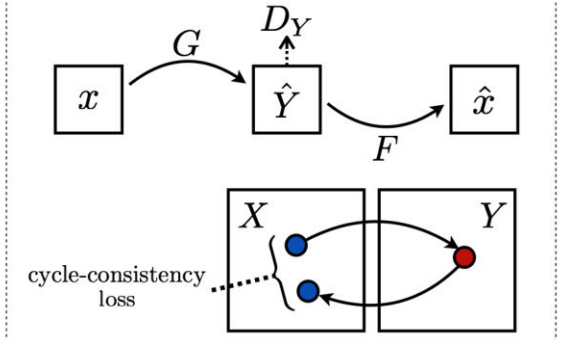
Cycle GAN

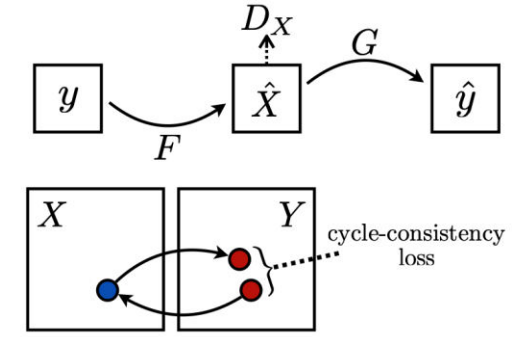
- Хорошо решает задачи доменной адаптации и pix2pix преобразования
- Не требует попарного совпадения объектов из двух доменов

- Относительно долго учится (6-18 часов)
- Требует много памяти (2 RTX 3090)







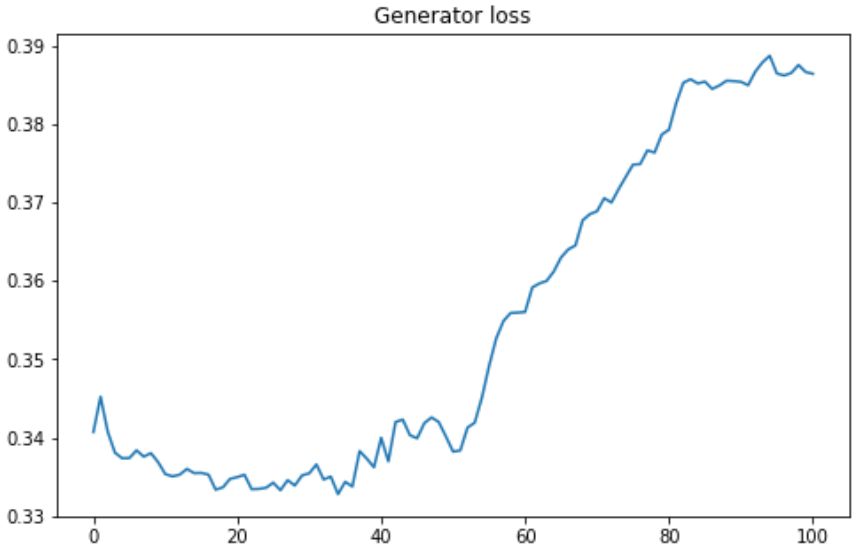


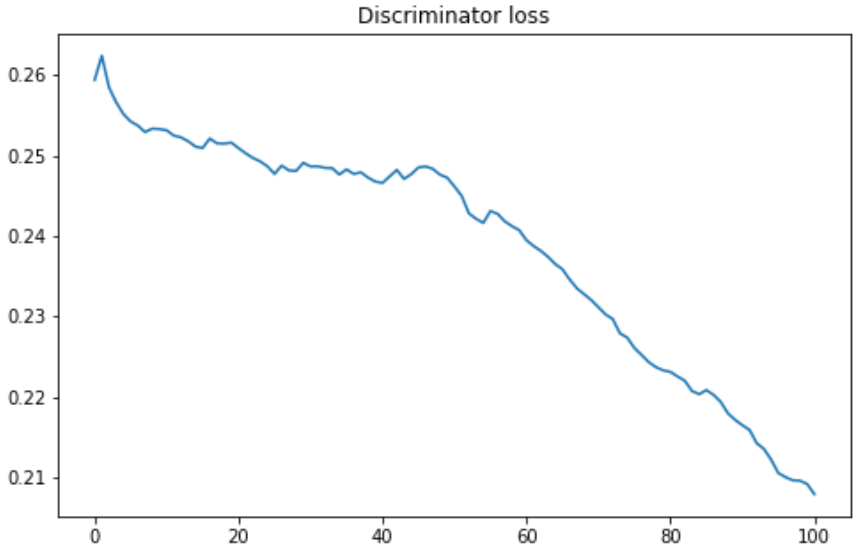
Базовая архитектура

- Оригинальная имплементация CycleGAN со стандартными параметрами
- Есть хорошие примеры, но в основном низкое качество переноса стиля
- Проблемы с переобучением дискриминатора
- Модель слабо изменяет оригинальную картинку

Переобучение дискриминатора

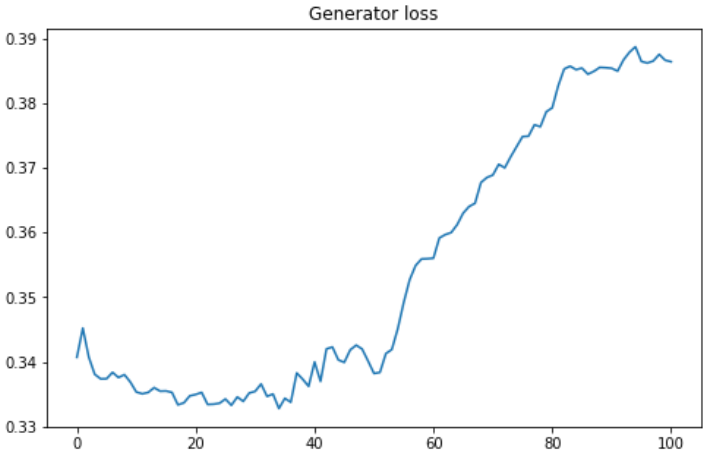
- При обучении стандартной архитектуры лосс **генератора** растет, лосс **дискриминатора** падает
- Варианты решения:
  - Зашумление входа дискриминатора, чтобы усложнить его задачу
  - Wasserstein loss
  - Cross entropy loss

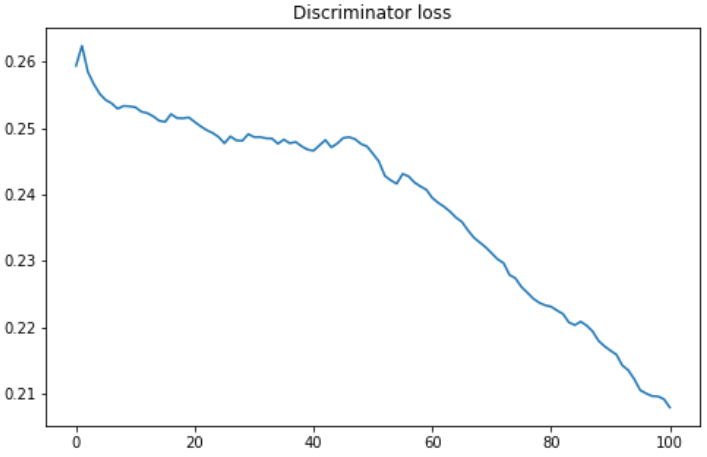




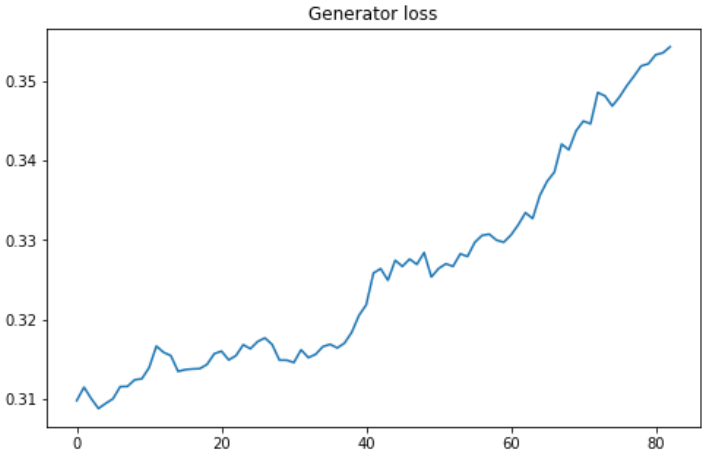
Переобучение дискриминатора

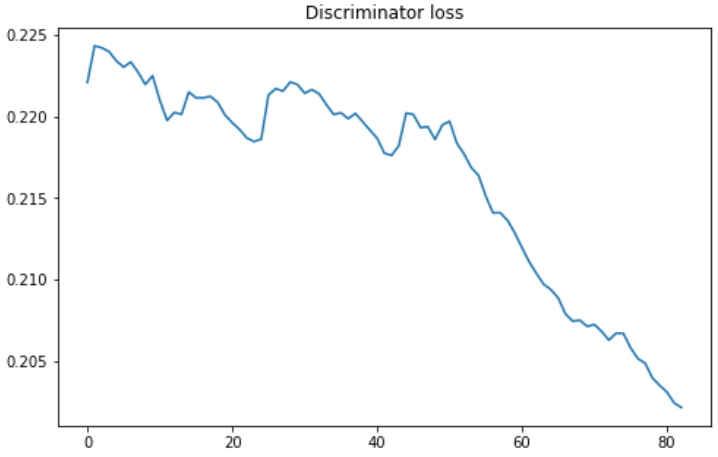
Base



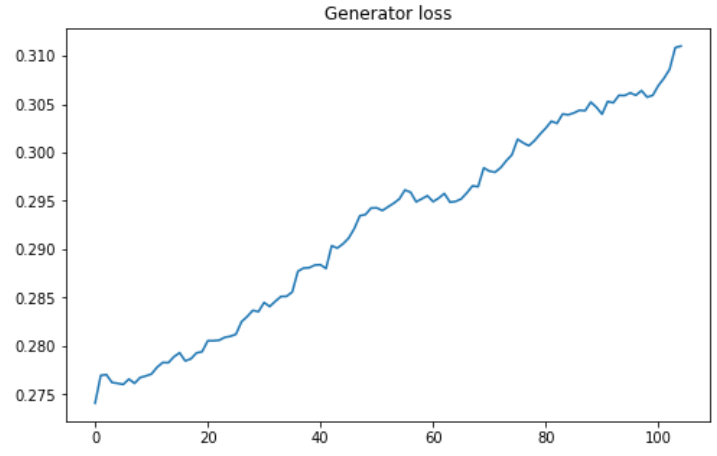


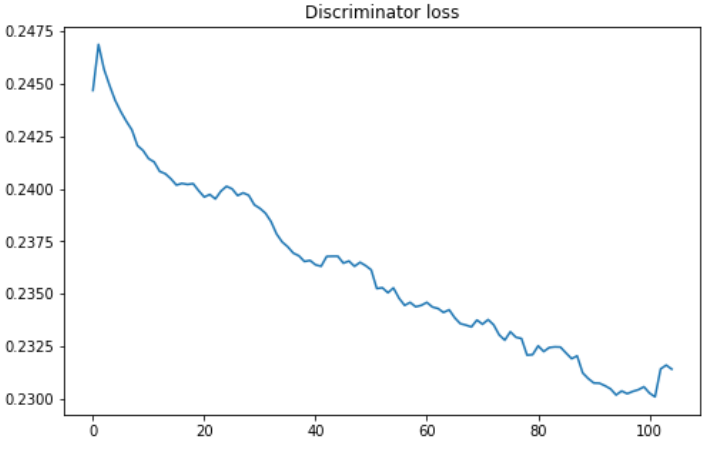
Gaussian noise std=0.1





Gaussian noise std=1





Переобучение дискриминатора

- При обучении стандартной архитектуры лосс **генератора** растет, лосс **дискриминатора** падает
- Варианты решения:
  - Зашумление входа дискриминатора, чтобы усложнить его задачу – лосс генератора уменьшился, но проблема осталась
  - Wasserstein loss – не удалось настроить для данной задачи
  - Cross entropy loss – применяли для следующих моделей

VGG identity loss

- Идея из статьи CartoonGAN
- Использовали VGG с тремя свертками
- Применяли VGG loss к новой и старой архитектуре

```
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()
          )
```

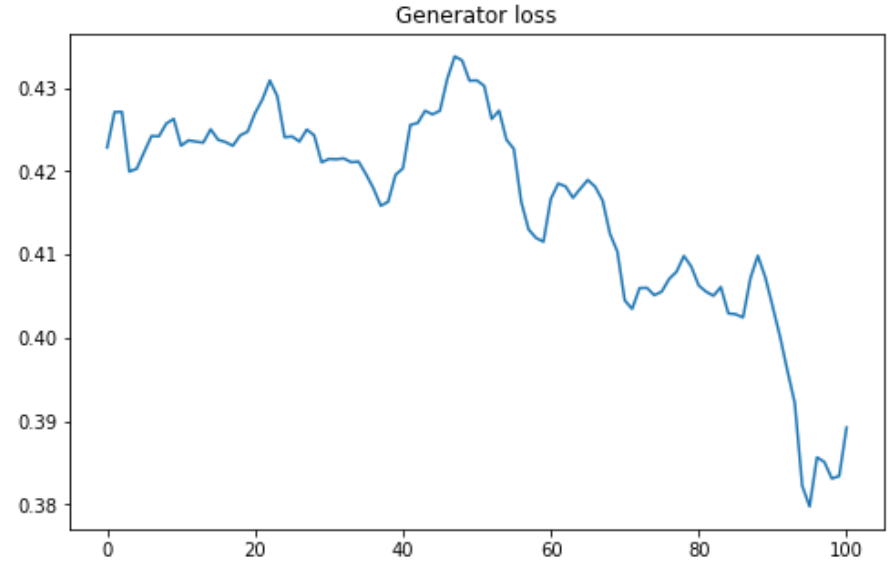
$$\mathcal{L}_{con}(G, D) =$$
$$\mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]$$

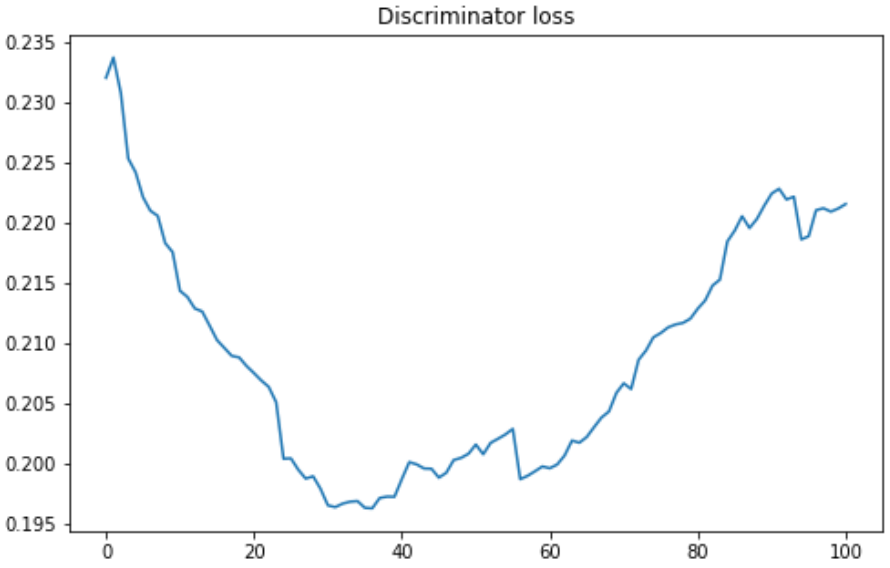
Исходное изображение

Оригинальная архитектура

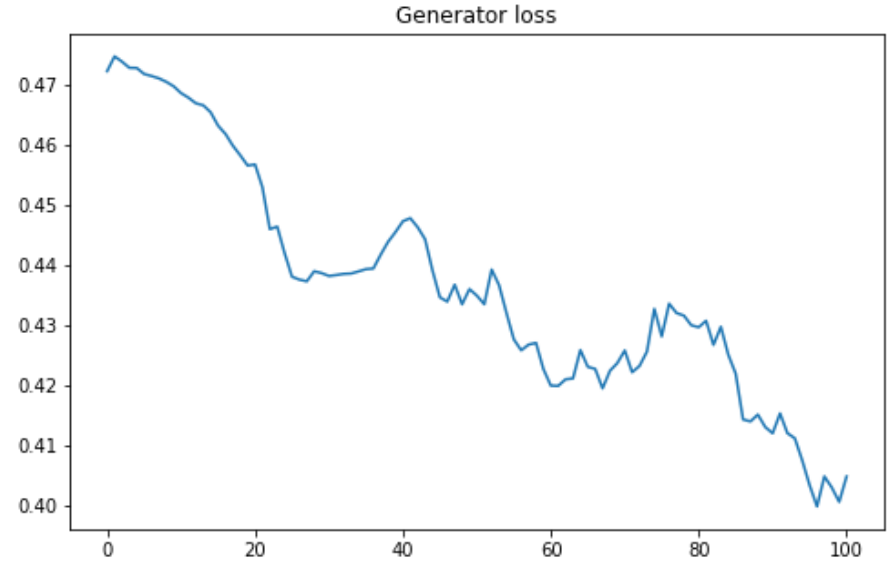
Новая архитектура

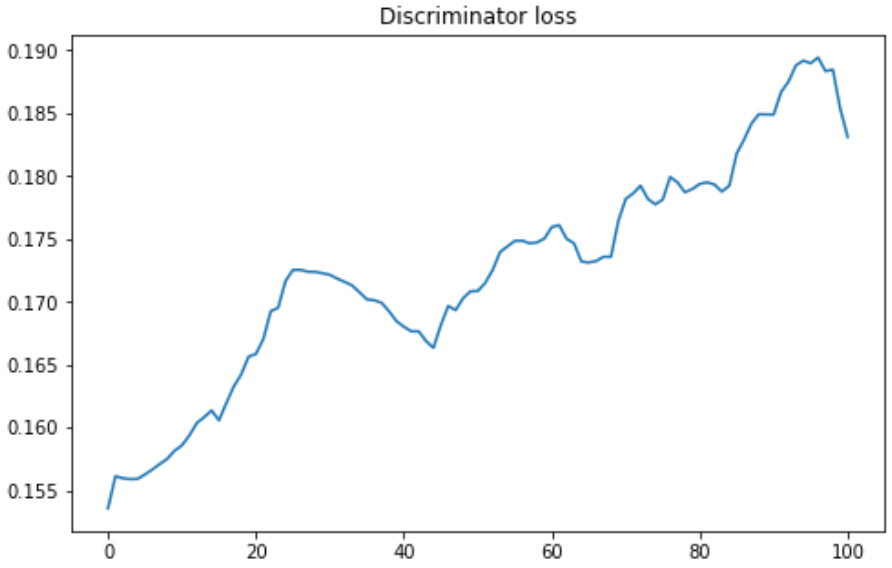
Оригинальная архитектура





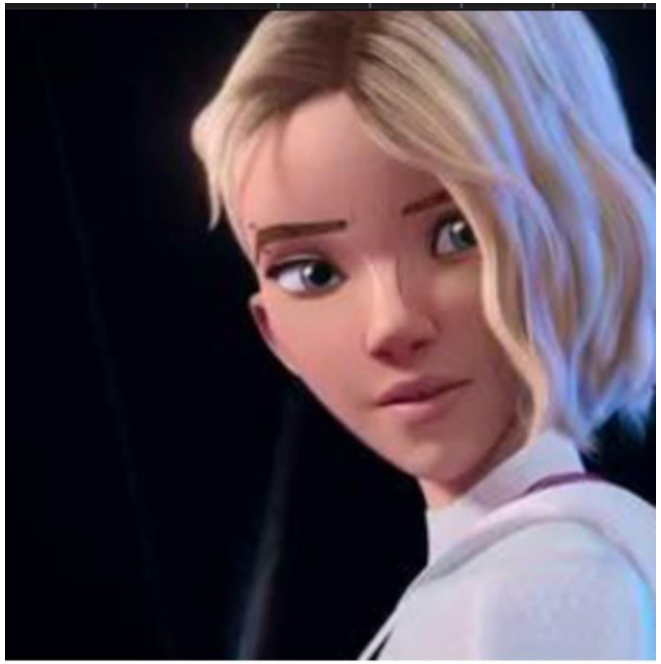
Новая архитектура

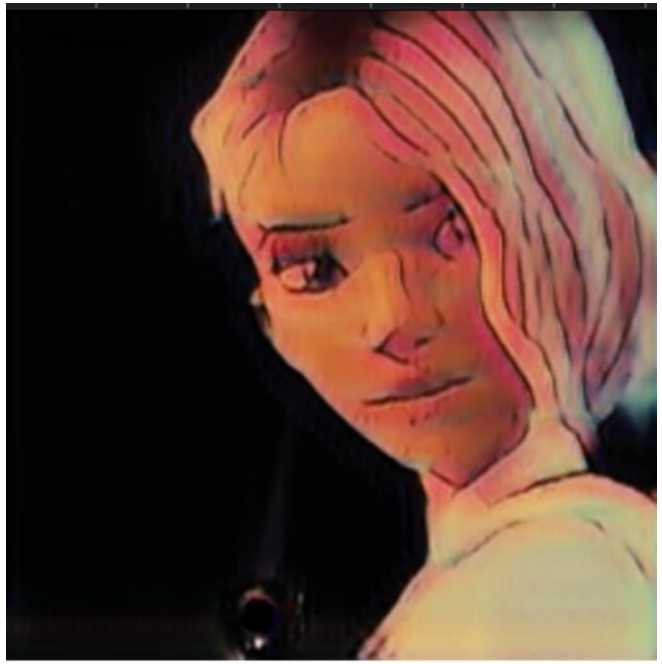


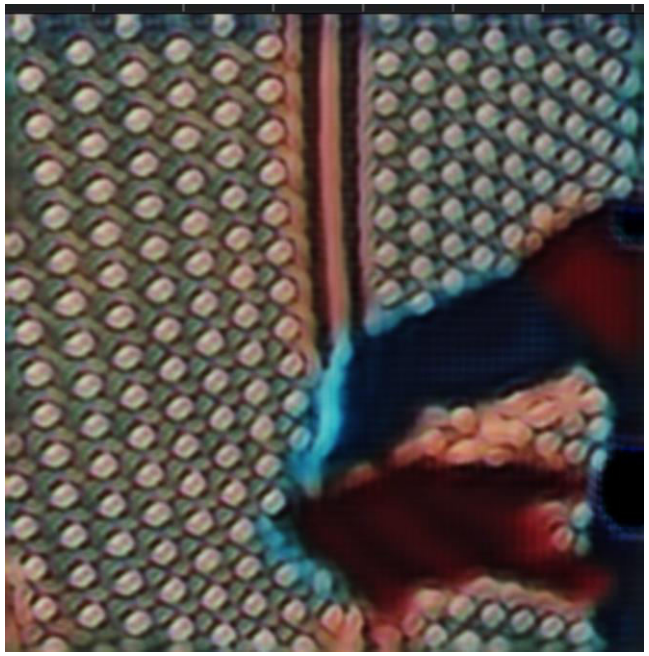


VGG identity loss

- MSE loss ограничивает перенос стиля, поэтому использовали Cross Entropy loss
- Обучение менее стабильное, иногда лосс дискриминатора падал до нуля
- Потом доучивали сеть с MSE, в итоге ситуация с лоссами улучшилась







Оригинальная архитектура

Новая архитектура