


Правительство Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук
Образовательная программа бакалавриата 09.03.04 «Программная инженерия»

ОТЧЕТ
по производственной практике
в структурном подразделении факультета компьютерных наук

Выполнила студентка
группы БПИ184
Аверичкина, В. Н.


(подпись)

Руководитель практики

Приглашенный преподаватель Департамента больших данных и информационного поиска

Айбек Аланов

Дата 24.08.2021

(оценка)

(подпись)

Аннотация

Данный документ представляет собой отчет по производственной практике студентки 3 курса НИУ ВШЭ Факультета Компьютерных Наук образовательной программы «Программная инженерия» Аверичкиной Виктории в структурном подразделении факультета компьютерных наук под руководством приглашенного преподавателя Департамента больших данных и информационного поиска Аланова Айбека.

Содержание

| | | |
|-----------------------------|-------|----|
| Аннотация | | 2 |
| Содержание | | 3 |
| Цель и задачи | | 4 |
| Обзор изученных материалов | | 5 |
| Основные методы и алгоритмы | | 6 |
| Основные этапы разработки | | 8 |
| Заключение | | 9 |
| Список источников | | 15 |
| Рабочий план-график | | 16 |

Цель и задачи

В рамках своей производственной практики на 3 курсе в структурном подразделении факультета компьютерных наук под руководством приглашенного преподавателя Департамента больших данных и информационного поиска Аланова Айбека мне было предложено разработать генеративную модель для изображений на основе датасета «Best Artworks of All Time», который включает в себе самые известные работы 50 самых влиятельных художников всех времен.

Для достижения поставленной цели далее мне было необходимо выполнить две основные задачи своей практики: во-первых, я должна была освоить основы глубинного обучения и фреймворка Pytorch, а во-вторых, мне нужно было освоить и наконец разработать модели GAN для генерации изображений из фиксированного датасета «Best Artworks of All Time».

Таким образом, цель:

- Разработка генеративных моделей для изображений

Задачи:

- Освоение основ глубинного обучения
- Освоение основ фреймворка Pytorch
- Освоение основ и принципов работы генеративно-сопоставительных нейросетей GAN
- Разработка модели GAN для генерации изображений из фиксированного датасета «Best Artworks of All Time»

Обзор изученных материалов

Для того, чтобы достигнуть поставленной цели, мне было необходимо изучить все основные понятия глубинного обучения. Для этого я использовала статью «Глубокое обучение (Deep Learning): основы»¹, в которой подробно описывается: что такое нейросеть, два основных типа обучения нейросетей, в чем заключается отличие обычных нейросетей от нейросетей глубокого обучения.

После изучения базовых терминов и основ глубинного обучения я приступила к ознакомлению с библиотекой Pytorch, которая помогает создавать, обучать и использовать модели и нейросети глубокого обучения. Чтобы разобраться в базовых функциях и возможностях, я прочитала статью «PyTorch для начинающих: основы»², где описывались различия тензоров (которые используются для обучения в Pytorch) и обыкновенных многомерных матриц, основные модули библиотеки Pytorch (такие как `torch.nn`, `torch.optim`, `torch.utils` и `torch.autograd`) и их использование для построения, обучения и использования моделей. Также в статье упоминался смысл использования GPU и перевода всех тензорных вычислений на GPU.

В качестве ознакомления с основными принципами и методами работы генеративно-сопоставительных нейросетей я воспользовалась статьей «A Gentle Introduction to Generative Adversarial Networks (GANs)»³. В ней разбирались основные принципы работы генеративно-сопоставительных нейросетей, их архитектура и различные виды GAN.

Для генерации изображений я решила использовать модель генеративно-сопоставительных нейросетей на основе глубокой сверточной архитектуры (Deep Convolutional Generative Adversarial Networks). Чтобы лучше ознакомиться с ее устройством, я изучила материал статьи «DCGAN: Tutorial»⁴. В ней подробно описываются различные слои нейросети, их предназначение, а также гиперпараметры, которые можно настроить самостоятельно. Здесь же описывается основной алгоритм обучения, настройки и реализации модели GAN.

¹ <https://neurohive.io/ru/osnovy-data-science/glubokoe-obuchenie-deep-learning-kratkij-tutorial/>

² https://ai-news.ru/2019/07/pytorch_dlya_nachinaushih_osnovy.html

³ <https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>

⁴ https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

Основные методы и алгоритмы

В основе реализованного мною решения задачи генерации изображений по датасету «Best Artworks of All Time» лежит алгоритм генеративно-состязательных нейросетей на основе глубокой сверточной архитектуры. Разберем основные понятия, методы и алгоритмы, которые были задействованы мною при решении поставленной задачи.

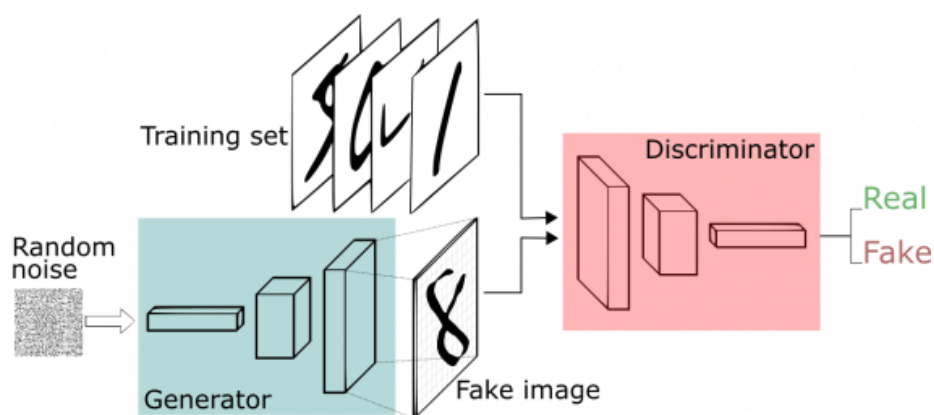
Для начала стоит отметить, что такое GAN. Генеративно-состязательная нейросеть (GAN) – это алгоритм машинного обучения, который основан на обучении без учителя (unsupervised learning). Он сочетает в себе две нейросетевые модели: дискриминатор и генератор.

Дискриминатор обучается на датасете и учится определять образы, представляемые на картинах. Его основная задача в рамках генеративно-состязательной нейросети – определить, является входное изображение настоящим или же сгенерированным.

Генератор учится создавать те самые изображения. Его основная задача – сгенерировать такое изображение, которое дискриминатор примет за настоящее.

На этом принципе и строится генеративно-состязательная нейросеть: генератор пытается «обмануть» созданием ненастоящего произведения искусства дискриминатор. Его цель – повысить ошибку дискриминатора. А дискриминатор в свою очередь учится различать настоящие и ненастоящие изображения. В этом и заключен состязательный момент данной нейросети.

Алгоритм работы следующий:



Изначально генератору на вход подается шум, который впоследствии тот и преобразует в итоговое изображение, с помощью настраивания весов. Сгенерированная картинка передается на вход дискриминатору вместе с тренировочным датасетом реальных картинок. Дискриминатор определяет реальное изображение или же сгенерированное.

Алгоритм обучения генератора и дискриминатора имеет следующий вид:

1. Настраиваем число эпох (итераций) обучения

Для дискриминатора:

2. Генерируем картинки из шума при помощи генератора

3. Просматриваем ошибку дискриминатора на сгенерированных картинках, которые она должна идентифицировать как ненастоящие, и на реальных картинках из датасета, которые она должна идентифицировать как настоящие.

4. Получаем общую ошибку дискриминатора

5. Обновляем параметры дискриминатора с помощью градиента

Для генератора:

6. Генерируем картинки из шума при помощи генератора

7. Считаем ошибку дискриминатора на сгенерированных картинках

8. Обновляем параметры генератора с помощью градиента

После чего генератор с уже настроенными параметрами способен выдавать сгенерированные изображения, которые сложно отличить от реальных.

Основные этапы разработки

Теперь, зная основные алгоритмы и принципы работы генеративно-сопоставительных нейросетей, можно описать из чего состоял процесс создания таковой в рамках моей производственной практики.

Для начала мне было необходимо все полученные из датасета данные привести к единому размеру (для удобства работы далее) и к формату тензоров, потому что именно тензоры в библиотеке `pytorch` осуществляют эффективные и быстрые математические операции, которые могли бы потребоваться для работы с ними.

После обработки входных данных настал черед конструирования архитектуры сетей для дискриминатора и генератора. Для генератора я выбрала 5 слоев, с пакетной нормализацией и функцией активации `ReLU` (возвращает 0 для отрицательных значений, само значение для положительных). Итоговая функция активации гиперболический тангенс позволяет на выход подавать значения от -1 до 1. Дискриминатор же состоит из 5 слоев с пакетной нормализацией, функцией активации `LeakyReLU` и итоговой функцией активации сигмоиды (значения от 0 до 1).

Для обучения использовалась функция потерь бинарная кросс-энтропия. Оптимизатор `Adam` с параметрами `learning rate = 1e-3`, `betas = 0.5` и `0.999`.

При обучении датасет делился на батчи размером 128. Поэтому за одну эпоху количество шагов оптимизатора = 64. Количество эпох = 220.

Для более эффективных и быстрых вычислений все необходимые тензоры с данными и нейросети были переведены на GPU. Процесс обучения каждой из представленных ниже вариаций моделей занял приблизительно 4 часа 50 минут.

Разработка велась на языке программирования `Python 3` с использованием сервиса `Yandex DataSphere` и предоставленными мощностями GPU.

Заключение

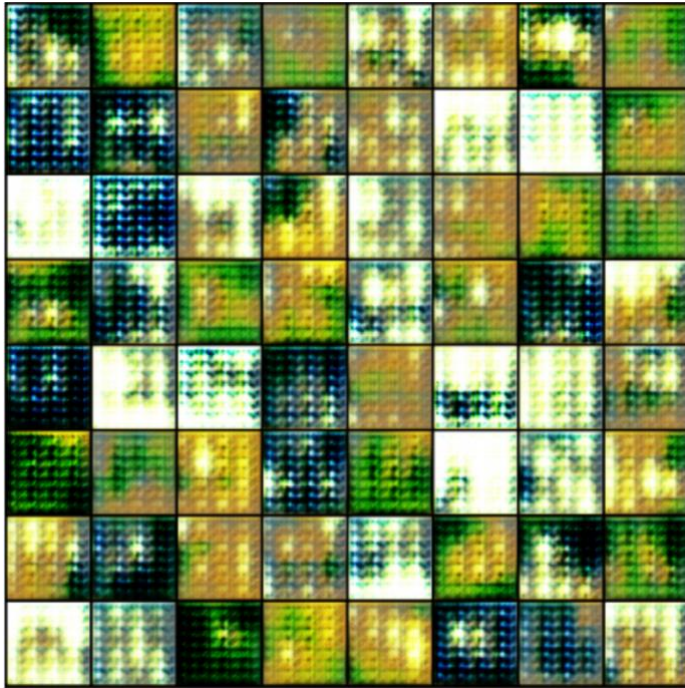
Далее я представлю результаты работы GAN для 3 моделей с некоторыми модификациями, о которых будет рассказано дальше.

Первая модель (базовая):

Параметры: $lr = 1e-3$, $\text{betas} = (0.5, 0.999)$

Результаты работы:

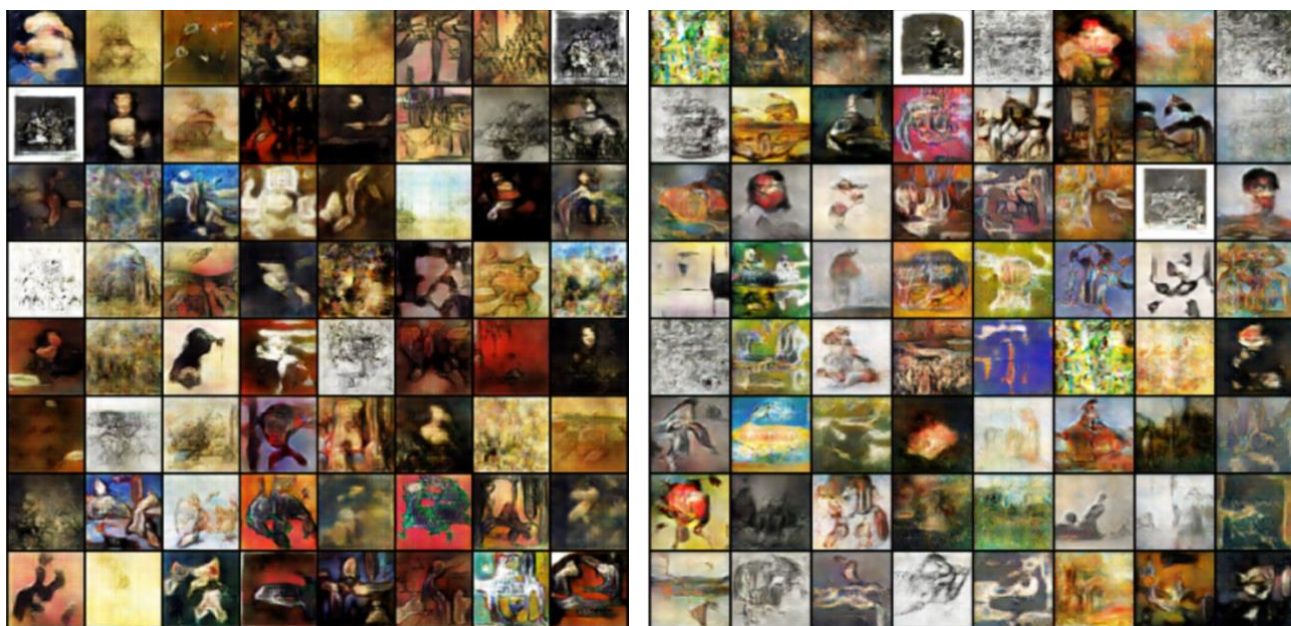
После первых 10 эпох:



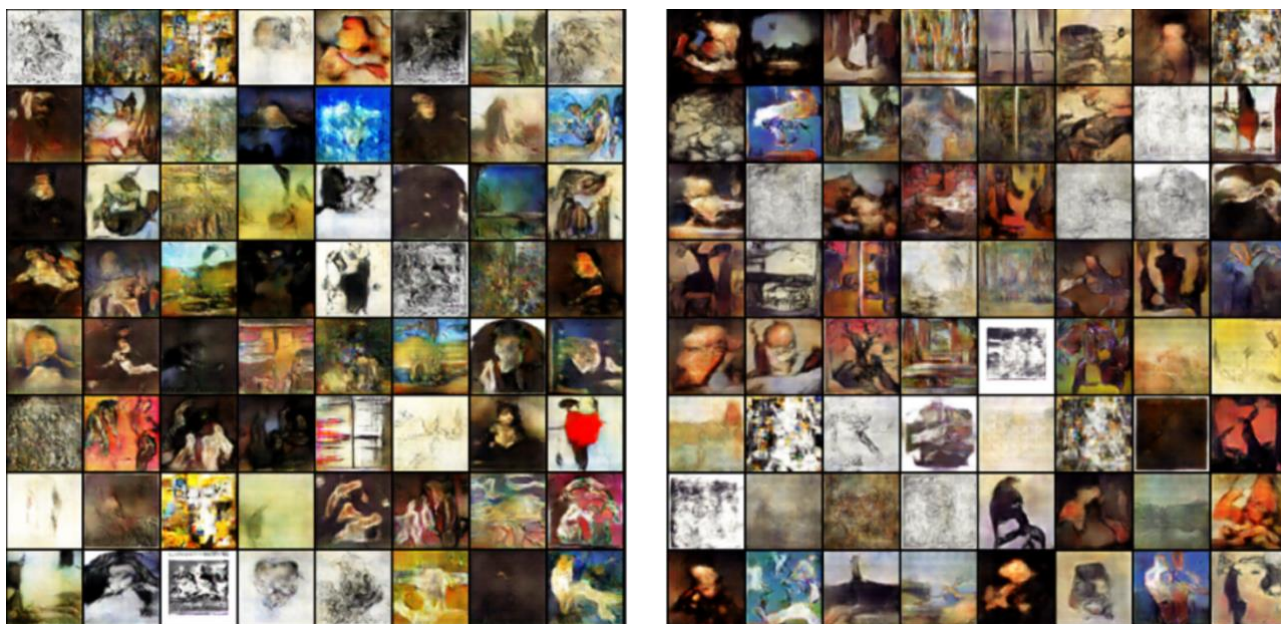
После первых 50 эпох (изображения цветные, но все еще присутствуют пятна):



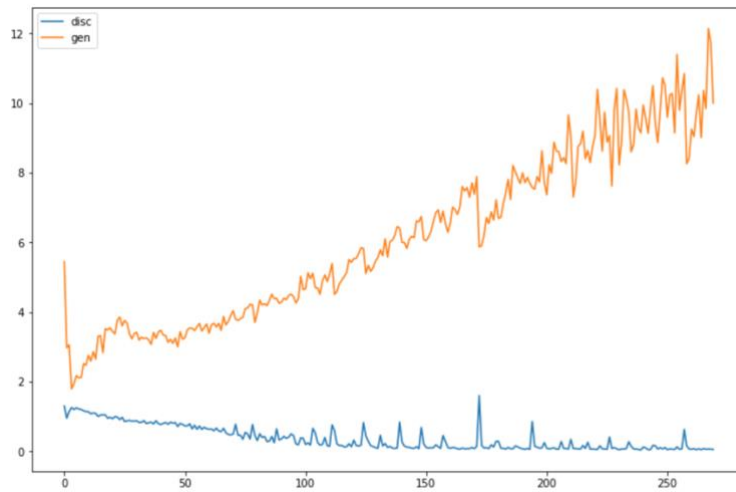
После первых 150 эпох (в выборке уже появляются картины, сгенерированные подражанием карандашу, а также светлые пятна – подражания портретам в выборке):



После 220 эпох:



Несмотря на достаточно хорошее качество сгенерированных изображений, график функции потерь на каждой итерации показывает, что к концу обучения ошибка генератора стремительно возрастает.



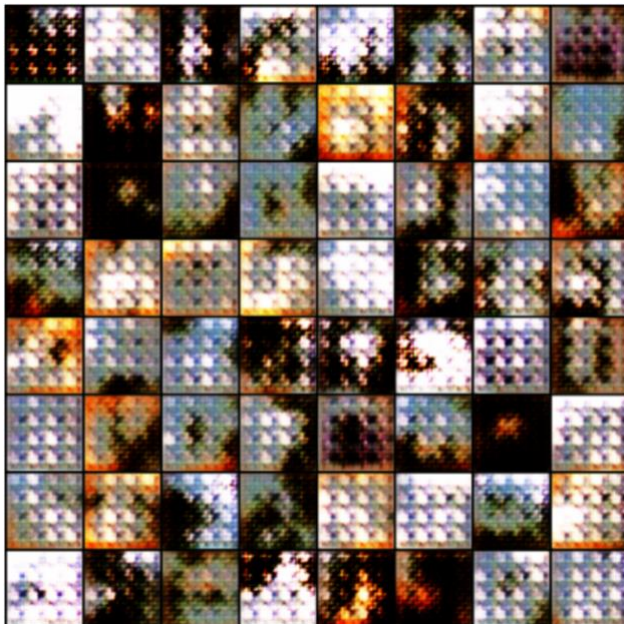
Дискриминатор слишком быстро учится распознавать реальные и сгенерированные изображения. Для этого можно добавить в оптимизатор дискриминатора параметр L2 регуляризации, который будет сильнее штрафовать дискриминатор за каждую ошибку.

Вторая модель (базовая с weight_decay):

Параметры: lr = 1e-3, betas = (0.5, 0.999), weight_decay = 5e-3

Результаты работы:

После 10 и 100 эпох соответственно:



После 150 и 220 эпох соответственно:

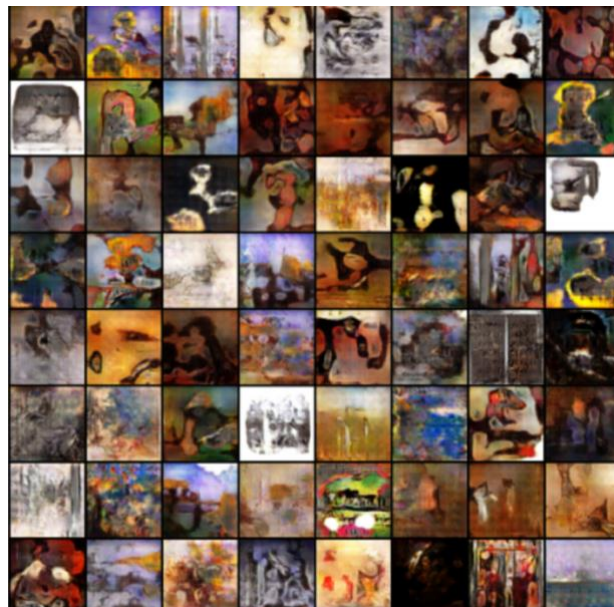
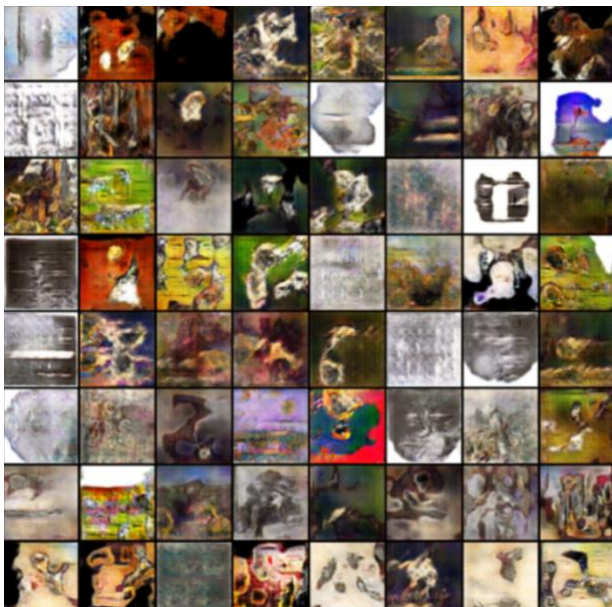
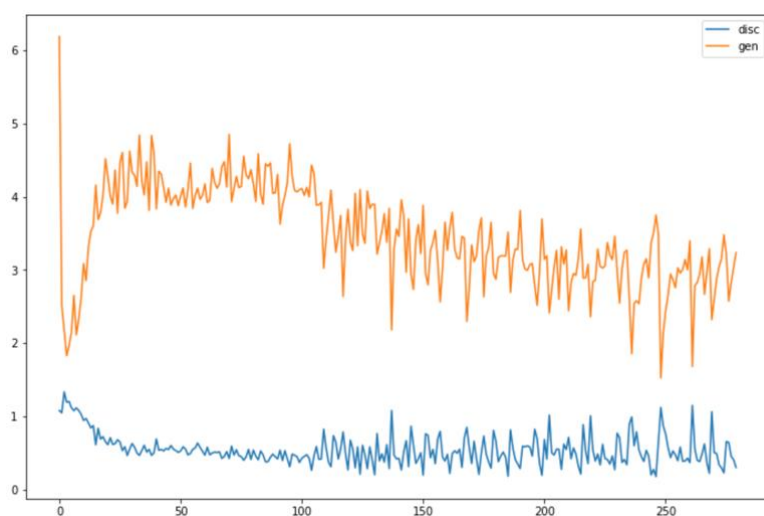


График функции потерь говорит нам о том, что штрафы помогли стабилизировать генератор, однако тот стал обучаться дольше обычного и к концу 220 эпохи все еще не способен выдавать нужные изображения.



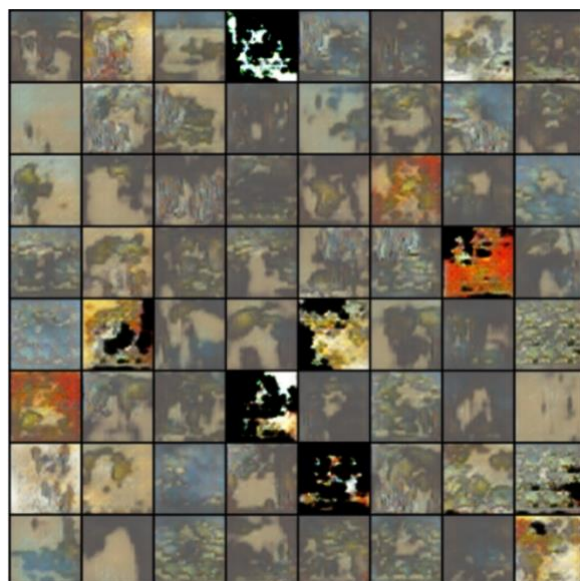
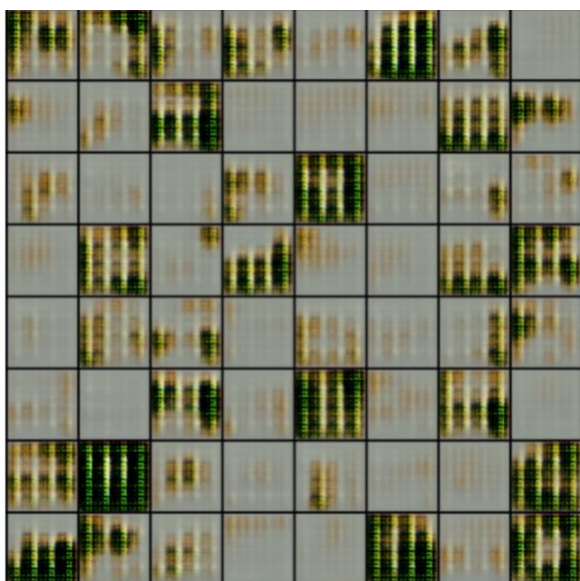
Воспользуемся спектральной нормализацией для дискриминатора. Однако в используемом источнике было отмечено, что скорость обучения заметно снижается, для чего имеет смысл выставить learning rate дискриминатора в 4 раза выше learning rate генератора.

Третья модель (базовая с spectral_norm):

Параметры: lr (discriminator) = $4e-3$, lr (generator) = $1e-3$, betas = (0.5, 0.999)

Результаты работы:

После 10 и 100 эпох соответственно:



После 150 и 220 эпох соответственно:

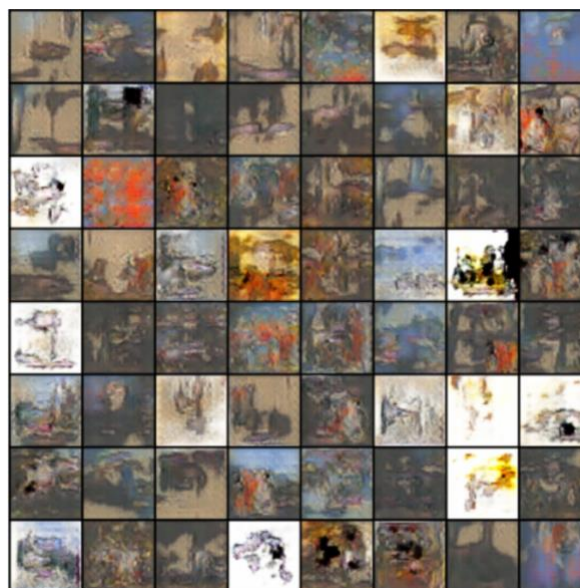
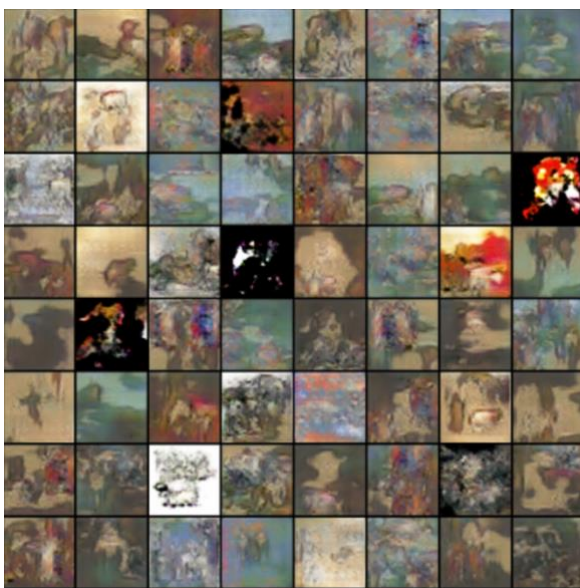
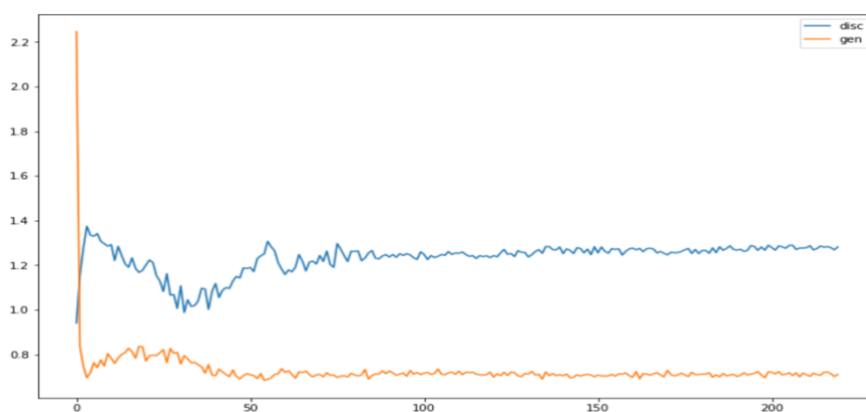


График функции потерь показывает, что нам удалось добиться цели генеративно-сопоставительных сетей, ведь ошибка дискриминатора теперь превышает ошибку генератора, однако качество генерируемых изображений сильно упало. Модели не хватает 220 эпох.



Таким образом, на ограниченном количестве в 220 эпох лучше всего себя повела базовая модель с параметрами $lr = 1e-3$, $betas = (0.5, 0.999)$. Ей удалось научиться генерировать картины абстрактного стиля с элементами и очертаниями человека. Генерация картин классического стиля ей недоступна, поскольку все элементы датасета содержат картины из самых различных стилей, которые изображают разные предметы, что не позволяет в представленное число эпох научиться генерировать все стили.

Итоговая обученная модель GAN доступна по ссылке:
https://github.com/AverichkinaVictoria/GAN_art

Список источников

Alec Radford, Luke Metz, Soumith Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016, [arXiv:1511.06434](https://arxiv.org/abs/1511.06434)

Goodfellow, Ian J.; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron & Bengio, Yoshua, Generative Adversarial Networks, 2014, [arXiv:1406.2661](https://arxiv.org/abs/1406.2661)

Minhyeok Lee, Junhee Seok, Regularization Methods for Generative Adversarial Networks: An Overview of Recent Studies, 2020, [arXiv:2005.09165](https://arxiv.org/abs/2005.09165)

ai-news.ru/2019/07/pytorch_dlya_nachinaushih_osnovy.html (дата обращения: 02.07.2021)

machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (дата обращения: 10.07.2021)

neurohive.io/ru/osnovy-data-science/glubokoe-obuchenie-deep-learning-kratkij-tutorial/

(дата обращения: 02.07.2021)

pytorch.org/docs/stable/generated/torch.nn.utils.spectral_norm.html (дата обращения: 14.07.2021)

pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html (дата обращения: 10.07.2021)

sthalles.github.io/advanced_gans/ (дата обращения: 14.07.2021)

towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b (дата обращения: 14.07.2021)

Рабочий план-график

| № п/п | Сроки проведения | Планируемые работы | Выполнено |
|----------|---------------------|---|-----------|
| 1 | 01.07.2021 | Инструктаж по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка | да |
| 2 | 02-07.07.2021 | Прослушивание лекций и семинаров | да |
| 3 | 08-13.07.2021 | Работа над проектом | да |
| 4 | 14.07.2021 | Подготовка отчёта по практике | да |