Аннотация.

1. **Цель работы.**

Цель данной работы: изучить Generative Adversarial Networks. Познакомиться с видами реализаций этих сетей. В качестве результат работы реализовать 2 вида сетей GAN для решения задачи генерации картинок с машинками и сравнить полученные результаты.

1. **Обзор публикаций и известных решений.**

Generative Adversarial Networks – это вид нейронный сетей, который состоит из двух отдельных нейронных сетей, которые называются генератор и дискриминатор. Основная идея таких сетей: генератор – создаёт решения и совершенствует, дискриминатор обучается на выборке подлинных решений. Между собой генератор и дискриминатор взаимодействуют так: генератор создаёт решение, которое вместе с реальным решение подаётся на вход дискриминатору. Дискриминатор, базируясь на уже полученных знаниях определяет где истинное и где ложное решение. Затем дискриминатор отправляет вектор ошибок – что нужно поправить генератору, чтобы его решение стало более реальным. Генератор создаёт новое решение учитывая ошибки и т.д. Хороший результат – когда дискриминатор с равной вероятностью выбирает ложное и подлинное решения. Такие сети часто используются для получения фотореалистичных изображений, например для элементов промышленного дизайна, дизайна интерьера, одежды, сумок, портфелей, сцен компьютерных игр и т. д. Сети GAN используются также в сети Facebook. В последнее время системы GANs стали использоваться для подготовки кадров фильмов или мультипликации. Также эти системы помогают воссоздать трёхмерную модель объекта с помощью фрагментарных изображений и улучшить изображения, полученные из астрономических наблюдений.

Одной из основных проблем генеративно-состязательных сетей является их нестабильность. Это затрудняет эксперименты с новыми вариантами или использование их в новых областях и резко ограничивает применимость таких сетей. Большинство статей, которые были изучены, описывают конкретные виды реализации GAN, которые уменьшают их нестабильность

* 1. **DCGAN - DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS [1].**

Исторические попытки увеличить масштабы использования GAN, использующих CNN для моделирования изображений, не увенчались успехом. Это побудило автор разработать альтернативный подход генерации изображений , которые могут быть смоделированы более надежно. Авторы также сталкивались с трудностями при попытке масштабирования GAN, использующих архитектуры CNN, обычно используемые в контролируемом обучении. Однако после обширного исследования моделей были определены семейства, которые привели к более стабильному обучению по целому ряду наборов данных и позволили обучаться более глубоким генеративным моделям. Основой описанного в статье подхода являются следующие пункты.

1. Все сверточные сети (Springenberg и др., 2014) , которые заменяют детерминированные пространственные функции объединения (например, maxpooling) со страйдами сверток, позволяя сети узнать свою собственную пространственную даунсамплинг. В статье данный подход используется в генераторе, позволяя ему изучать собственную пространственную дискретизацию и дискриминатор.
2. Тенденция к устранению полностью связанных слоев поверх сверточных функций.
3. Пакетная Нормализация (Иоффе & Szegedy, 2015) , которая стабилизирует обучения путем нормализации входа для каждого блока , чтобы иметь нулевое среднее и единичную дисперсию.
4. Активации РЕЛУ (Наир & Хинтон, 2010) используется в генераторе, за исключением выходного слоя , который использует функцию TANH.

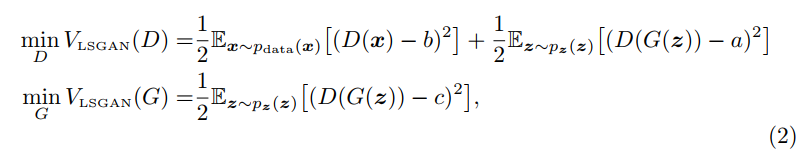
Также в статье были даны следующие рекомендации:

1. Заменить все пулы слоев чередующимися свертками (дискриминатором) и дробильными свертками (генератором).
2. Использовать batchnorm как в генераторе, так и в дискриминаторе.
3. Удалить полностью подключенные скрытые слои для более глубоких архитектур.
4. Использовать активацию ReLU в генераторе для всех слоев, за исключением вывода, в котором использовать Tanh. Использовать активацию LeakyReLU в дискриминаторе для всех слоев.

**2.2 Least Squares Generative Adversarial Networks [2].**

Неконтролируемое обучение с генеративными состязательными сетями (GAN) оказалось очень успешным. Регулярные ГАН выдвигают гипотезу о дискриминаторе, как классификаторе с функцией потери сигмовидной кросс-энтропии. Однако, эта функция потерь может привести к проблеме исчезающих градиентов в процессе обучения. Чтобы преодолеть такую проблему, в этой статье предлагаются порождающие адвериальные сети наименьших квадратов (LSGAN), которые используют функцию потерь наименьших квадратов для дискриминатора.

Чтобы исправить проблему нестабильности ГАН, в статье предлагаются генерирующие интеллектуальные сети наименьших квадратов (LSGAN). Предположим, используется схема кодирования a-b для дискриминатора, где a и b - метки для поддельных данных и реальных данных, соответственно. Тогда объективные функции для LSGAN могут быть определены следующим образом:



где c обозначает значение, которое отвечает за чило раз, когда Дискриминатор посчитал поддельные данные Генератора реальными.

Преимущества LSGAN: он наказывает поддельные образцы, даже если они правильно классифицированы. Это помогает LSGAN генерировать образцы, которые ближе к реальным данным.

Во-вторых, наказание образцов, лежащих далеко от границы решения, может генерировать больше градиентов при обновлении генератора, что, в свою очередь, уменьшает проблему исчезновения градиентов. Это позволяет LSGAN выполнять более стабильную работу в процессе обучения.

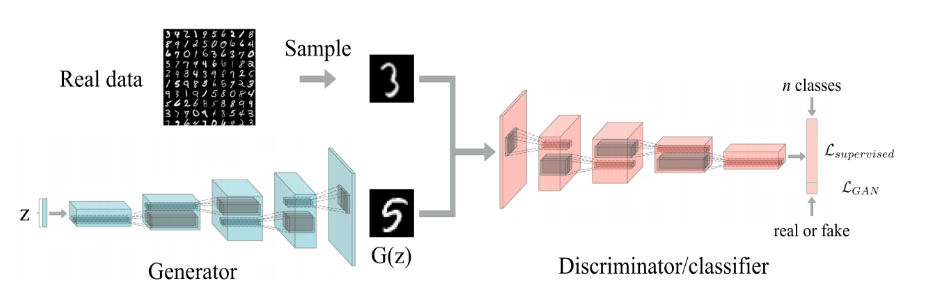
Экспериментальные результаты показывают, что LSGAN описанный в этой статье генерируют изображения более высокого качества и более реальные, чем обычные GAN. Также LSGAN более стабильные относительно обычных GAN.

* 1. **SSGAN**

Semi-supervised Learning Generative Adversarial Networks – модификация сети GAN, представляющая собой подход, в котором совмещается обучение без учителя с подходом с обучением учителем.

Основая идея данной реализации сети заключается в том, чтобы использовать образны, сгенерированные генератором GAN для повышения производительности задач классификации изображений путем улучшения обобщения.

В общем, основной идеей является обучение сети, играющей как роли классификатора, выполняющего задачу классификации изображений, так и дискриминатора, обученного различать сгенерированные сэмплы, созданные генератором из реальных данных. Чтобы быть более конкретным, дискриминатор / классификатор принимает изображение как входное и классифицирует его в n + 1 классы, где n - число классов задачи классификации. Истинные образцы классифицируются в первые n классов, а сгенерированные образцы классифицируются в n + 1-й класс, как показано на рисунке ниже.



Данная сеть была успешна протестирована на таких известных датасетах, как CIFAR10, MNIST, SVHN.

1. **Описание прототипа.**

В ходе работы былонаписано два прототипа задачей которых является генерация изображений с машинами. Первый прототип основан на сети DCGAN, второй на SSGAN.

* 1. **Описание первого прототипа, основанного на DCGAN.**

Опираясь на знания, полученные из статьей, было решено использовать следующую архитектуру сети:

Генератор состоит из пяти слоёв:

1 Слой: Свёртка с дробным числом -> Batch normalization -> ReLU активация

2 Слой: Свёртка с дробным числом -> Batch normalization -> ReLU активация

3 Слой: Свёртка с дробным числом -> Batch normalization -> ReLU активация

4 Слой: Свёртка с дробным числом -> Batch normalization -> ReLU активация

5 Слой: Свёртка с дробным числом -> Tanh функция

Дискриминатор тоже состоит из пяти слоёв:

1 Слой: Свёртка -> LeakyReLU активация

2 Слой: Свёртка -> Batch normalization -> LeakyReLU активация

3 Слой: Транспонированная свёртка -> Batch normalization -> LeakyReLU активация

4 Слой: Транспонированная свёртка -> Batch normalization -> LeakyReLU активация

5 Слой: Свёртка

В качестве функции потерь использовалась - binary cross-entropy

Код реализации доступен по ссылке <https://github.com/ustadenis/GAN/blob/master/code/DCGan_car_generator.ipynb>.

* 1. **Использование SSGAN для генерации изображений на исходном датасете**

Для выполнения данной задачи была взята реализация SSGAN на базе фреймворка TensorFlow (<https://github.com/gitlimlab/SSGAN-Tensorflow>). Основные задачи были следующие:

1. Протестировать производительность и время работы SSGAN на тестовом наборе данных (CIFAR10)
2. Реализовать алгоритм, представляющий текущий датасет (машины) в виде необходимого сети для работы h5py файла (data.hy), представляющий из себя набор данный, хранящий исходное изображение + его лейбл.
3. Протестировать SSGAN на полученном датасете.
4. **Используемые наборы данных**

Для нашей задачи мы использовали следующие данные <http://ai.stanford.edu/~jkrause/cars/car_dataset.html>. Этот датасет содержит 16185 изображений и 196 классов машин.

1. **Описание сценариев для демонстрации работы.**

В качестве демонстрации нашей работы можно предоставить код с комментариями в Jupyter Notebook, рассказать про нейронные сети, которые использовались в работе и показать результаты: изображения с машинами.

1. **Описание эксперементов и полученные результаты.**

Результаты нашей работы приведены ниже.

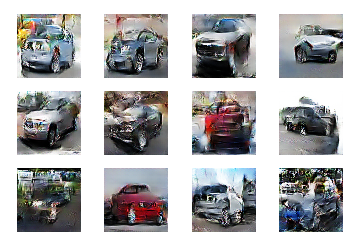
DCGAN:

Генерация машин после 125 эпох обучения





Генерация машин после 250 эпох





Как видно с увеличением эпох качество сгенерированных картинок улучшается.

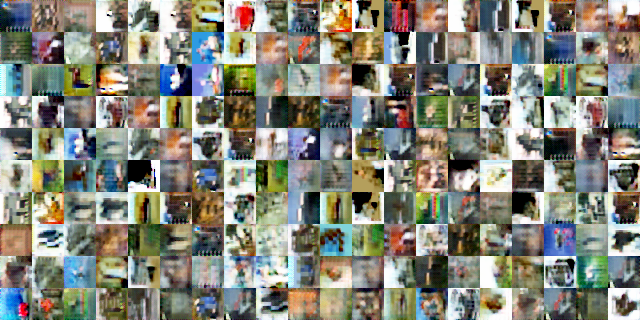
SSGAN:

Основной проблемой в работе данной реализации сети стали требования к высокой производительности машины, на которой происходит обучение. Сеть способна показывать хорошие результаты при длительном обучении (100+ epoch). Одной эпохой является 1000 итераций, в то время когда 1 итерация в среднем длится около 20 секунд (при использовании CPU).

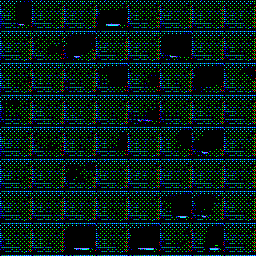
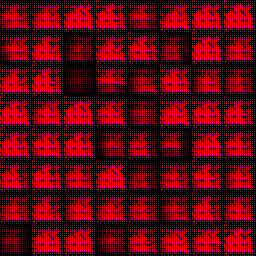
В конечном итоге для обучения данной сети на среднестатистическом компьютере без возможности использования GPU (или без наличия производительного GPU) требуется очень большое количество времени.

Сеть, обученная на CIFAR10 в 1000 эпох (1 000 000 итераций)

Способна сгенерировать изображения подобного вида:



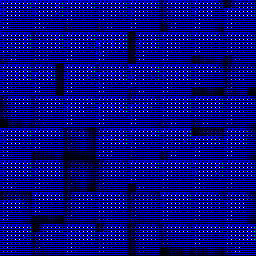
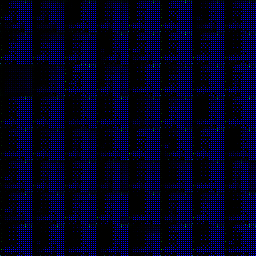
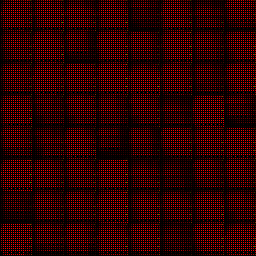
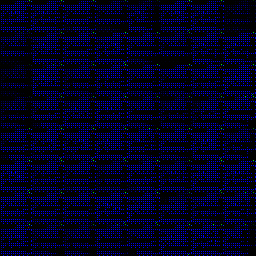
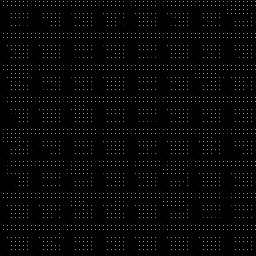
В то время как сеть, обученная на 30 эпохах (30 000 итераций), не способна предоставить удовлетворительного результата:



(1 epoch) (27 epoch)

Время выполнения 30 эпох на процессоре Intel Core i7 2600 – 15 часов.

Результаты обучения на сгенерированном датасете (cars dataset):



(30 000 итераций)

Из полученных результатов можно сделать лишь один вывод: производительности среднестатистического компьютера с CPU недостаточно для того, чтобы судить о реальных возможностях данной сети.

1. **Что делать дальше.**

В качестве улучшений можно использовать следующее:

1. Увеличить число эпох для обучения и сделать обучение на GPU
2. Реализовать другие варианты GAN.
3. Существуют разные наборы данных с машинками, есть предположение, что GAN не будет стабилен на разнных данных, поэтому в качестве улучшения можно создать более стабильную сеть GAN, используя рекомендации из статей.
4. **Список литературы:**

1 – https://arxiv.org/abs/1511.06434.

2 - https://arxiv.org/abs/1611.04076.

3 - <https://arxiv.org/abs/1701.04862>.

4 - <https://arxiv.org/abs/1606.03498>