МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра дискретной математики и алгоритмики

Бычков Алексей Вячеславович

Алгоритмы синтеза изображений в больших разрешениях на основе генеративно-состязательных нейронных сетей

Магистерская диссертация
специальность 1-31 81 09 «Алгоритмы и системы обработки больших объемов информации»

Научный руководитель
Краснопрошин Виктор Владимирович
доктор технических наук,
профессор

Допущен к защите
«____»_____ 2020 г.
Зав. кафедрой дискретной математики и алгоритмики
_____ В.М. Котов
доктор физико-математических наук, профессор

Минск 2020

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Магистерская диссертация 53 с., 25 рис., 35 источников

Ключевые слова: СИНТЕЗ ИЗОБРАЖЕНИЙ, ОПТИМИЗАЦИЯ, ВЫСОКОЕ РАЗРЕШЕНИЕ, ФУНКЦИЯ ОШИБКИ, НОРМАЛИЗАЦИЯ, ГЕНЕРАТОР, ДИСКРИМИНАТОР, ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНАЯ НЕЙРОННАЯ СЕТЬ.

Объект исследования – методы синтеза изображений в высоких разрешениях при помощи генеративно-состязательных сетей.

Цель работы – изучить современные алгоритмы синтеза изображений, разработать и реализовать на их основе свое решение, лишенное недостатков своих составных частей.

Область применения - генерация данных, восстановление данных.

Методы исследования – анализ, эксперимент, тестирование, сравнение.

Результаты исследования:

- изучены современные алгоритмы синтеза изображений в больших разрешениях
- на основании теоретического исследования было выбрано два типа генеративно-состязательных сетей в качестве основы для построения новых алгоритмов.
- построены и реализованы два алгоритма генеративно-состязательных сетей.
- реализованные алгоритмы обучены на нескольких различных наборах данных.
- продемонстрирована возможность с помощью данных алгоритмов синтезировать изображения в больших разрешениях.
- указаны их достоинства и недостатки

РЭФЕРАТ

Магістарская дысертацыя 53 ст., 25 мал., 35 крыніц.

Ключавыя словы: , АПТЫМІЗАЦЫЯ, ВЫСОКІ ДАЗВОЛ, ФУНКЦЫЯ ПАМЫЛКІ, НАРМАЛІЗАЦЫЯ, ГЕНЕРАТАР, ДЫСКРЫМІНАТАР, ГЕНЕРАТЫУНА-СПАБОРНЫЯ НЕЙРОННЫЯ СЕТКІ.

Аб'ект даследавання - метады сінтэзу малюнкаў у высокіх дазволах пры дапамозе генератыўна-спаборных сетак ..

Мэта работы - вывучыць сучасныя алгарытмы сінтэзу малюнкаў, распрацаваць і рэалізаваць на іх аснове сваё рашэнне, пазбаўленае недахопаў сваіх складовых частак.

Вобласць прымянення - генерацыя дадзеных, аднаўленне дадзеных.

Метады даследавання - аналіз, эксперымент, тэставанне, параўнанне.

Вынікі даследавання:

- вывучаны сучасныя алгарытмы сінтэзу малюнкаў у вялікіх дазволах
- На падставе тэарэтычнага даследавання было абрана два тыпу генератыўна-спаборных сетак у якасці асновы для пабудовы новых алгарытмаў.
- пабудаваны і рэалізаваны два алгарытму генератыўна-спаборных сетак.
- пабудаваныя алгарытмы навучаны на некалькіх розных датасетах.
- прадэманстравана іх магчымасць сінтэзаваць выявы ў вялікіх дазволах.
- пазначаны іх вартасці і недахопы

ABSTRACT

Master thesis 53 p., 35 figs.., 35 references.

Key words: IMAGE SYNTHESIS, OPTIMIZATION, HIGH RESOLUTION, LOSS FUNCTION, NORMALIZATION, GENERATOR, DISCRIMINATOR, GENERATIVE ADVERSARIAL NEURAL NETWORK.

Research field - high-resolution image synthesis methods employing generative adversarial networks (GAN).

The aim of the work is to study state of the art image synthesis algorithms. The rationale for this project was development and implementation of a new solution based on existing methods without known disadvantages of their components.

Applications - data generation, data recovery. Research methods - analysis, experiment, testing, comparison.

Main results are:

- review of modern algorithms for high-resolution image synthesis was performed
- Following a theoretical analysis, two types of generative-competitive networks were chosen as the basis for new algorithms.
- two generative adversarial networks were proposed and implemented.
- algorithms were trained on several different datasets.
- Experiments demonstrated their ability to synthesize images in high resolutions.
- advantages and disadvantages of the proposed methods were noticed

ВВЕДЕНИЕ

На протяжении уже долгого времени одной из основных проблем алгоритмов машинного обучения с учителем (англ. “supervised learning”) является проблема данных. В частности, для обучения нейросетей зачастую необходимы выборки из миллионов элементов. Если рассмотреть в качестве примера обучение распознаванию изображений, то миллионы изображений должны быть в достаточном качестве и должны быть размечены, что требует ручного труда большого количества людей. Тем самым, косвенные затраты на обучение нейросети по распознаванию изображений колоссальны.

Выходом мог бы стать алгоритм, способный без участия человека генерировать подобные выборки. Целью данной магистерской работы являются именно подобные алгоритмы. За последние несколько лет наметился очевидный прогресс в этой области благодаря появлению генеративно-состязательных сетей (GAN). Однако, подобные нейронные сети являются крайне сложными для обучения, и вопрос синтеза изображений в больших разрешениях с их помощью по-прежнему остаётся открытым.

Последние исследования в области GAN направлены на реализацию техник, которые бы позволяли привнести большую стабильность в процесс обучения подобного рода сетей: многочисленные виды нормализации данных, такие как спектральная нормализация, нормализация «pixel-wise» и многие другие; различные функции ошибки, такие как расстояние Вассерштейна, Кульбака-Лейблера и др.

Также не маловажным фактором является скорость обучения. К примеру, разработка ученых компании NVIDIA «Progressive growing GAN», по заявлению команды, обучается для синтеза изображений в разрешении 1024 пикселей на 8 максимальных видеокартах NVIDIA V100 (NVI1) в течение 2 дней (Pro).

В данной работе будет проведен анализ существующих подходов в реализации GAN. На основе наиболее успешных из них будет реализован алгоритм, целью которого является синтез изображений в достаточно больших разрешениях. При этом при выборе подходов, которые войдут в качестве составных частей в результирующий алгоритм, одним из основополагающих факторов будет скорость обучения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

1.1 Искусственные нейронные сети: основные понятия, классификация

Под искусственными нейронными сетями (в дальнейшем просто «нейронная сеть») подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Они представляют собой распределенные и параллельные системы, способные к адаптивному обучению путем анализа положительных и отрицательных воздействий (Круглов В.В, 2002).

Нейронные сети являются одним из множества типов алгоритмов машинного обучения. Однако, за последние годы именно они приобрели большую популярность.

Основной структурной единицей нейронной сети является искусственный нейрон. Множество таких нейронов взаимодействует между собой. На рисунке ниже представлена схема искусственного нейрона:

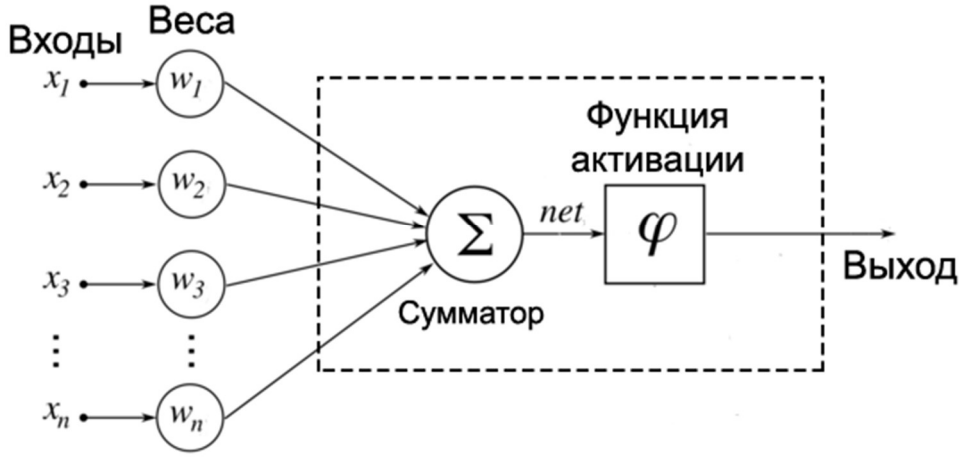

Рис. 1 Схема искусственного нейрона (Нейронные сети)

Как видно из схемы, у нейрона есть несколько входов, на которых он принимает различные сигналы, преобразует их посредством нелинейной активационной функции и передает другим нейронам. Другими словами, искусственный нейрон — это такая функция $R^n \rightarrow R$, которая преобразует несколько входных параметров в один выходной.

Своей эффективностью нейросетевые технологии обязаны двум составляющим:

1) распараллеливанию обработки больших объемов информации.

2) способности обучаться, т. е. создавать обобщение или, иными словами, способности получать обоснованный результат на основании данных, которые не присутствовали в процессе обучения (Хайкин, 2016)

Как и в случае более общей задачи машинного обучения, задача нейросети – найти достаточно хорошую функцию, аппроксимирующую обучающую выборку, при условии, чтобы эта функция обобщалась и на все остальные еще не встреченные входные данные. Отсюда возникают два ключевых понятия: функция ошибки и регуляризация.

Задача функции ошибки предельно проста: оптимизируя эту функцию, а именно, минимизируя ее значение, будет улучшаться точность ответов нейросети. В качестве алгоритма оптимизации используется метод градиентного спуска (и его многочисленные варианты улучшения), суть которого сводится к поэтапному изменению значений параметров функции на значение, пропорциональное градиенту в точке.

$\omega_i = \omega_i - \alpha * \frac{\partial E}{\partial w_i}$, где α – это скорость обучения (learning rate), E – функция ошибки.

В качестве функции ошибки часто используется средняя квадратичная ошибка, в которой минимизируется среднее квадратов отклонений предсказанных значений от истинных:

$$RSS(\omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i w)^2$$

Подобная функция ошибки используется в задачах регрессии. Так же в задачах регрессии могут использоваться различные её модификации, такие как средняя квадратичная логарифмическая ошибка (MSLE) : $\frac{1}{N}\sum_{i=1}^{N}(log\ (y_i + 1) - log\ (x_i w + 1))^2$, которую можно использовать в случае, когда целевое значение обучающей выборки имеет большой разброс, и при обучении подобные случаи нет необходимости налагать слишком высокие штрафы.

В задачах же классификации используется функция кросс-энтропии, которая бывает как бинарной, в случае выбора между двумя возможными классами, так и мульти-классовой:

$J = -\frac{1}{N}\sum_i L_i * \log(S_i)$, где L_i- элемент вектора значения в one-hot (wikipedia) кодировке, в которой длина вектора равна количеству классов; для правильного класса проставлена 1, для всех остальных – 0. S_i – результат работы алгоритма, а именно, вероятность , с которой алгоритм относит результат обработки входящих данных к i – ому классу. Вектор S_i - в большинстве случае – это результат работы функции softmax, которая используется в паре с кросс-энтропией в качестве функции ошибки.

Бинарная классификация является частным случаем кросс-энтропии и имеет вид:

$$J = -\frac{1}{N} * \sum_{n=1}^{N}[y_n log y_n^* + (1 - y_n)\log(1 - y_n^*)]$$

Альтернативой функции кросс-энтропии в случае бинарной классификации может быть так называемая hinge loss: $\max(0, 1 - \hat{y}^* * y\ )$. Для ее использования целевые значения должны принадлежать множеству {-1,1}. Исследования относительно данной функции ошибок на данный момент противоречивы: в некоторых случаях данная функция приводит к увеличению

производительности в сравнении с функцией кросс-энтропии, в некоторых – наоборот.

Еще одним важным понятием нейронных сетей является функция активации – нелинейная функция, которая применяется нейроном для получения выходного значения. Существуют несколько функций, которые на данный момент используются в большей части решений на основе нейронных сетей: sigmoid: $f(x) = \frac{1}{1+ e^{-x}}$ (и его аналог для задачи мульти-классовой классификации softmax: $f(x) = \frac{e^{x_i}}{\sum_i e^{x_j}}$), tanh: $f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ и relu : $f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$ (со своими производными, такими как leakyRelu (Wikipedia)). Графики этих функций отображены ниже на Рис. 2.

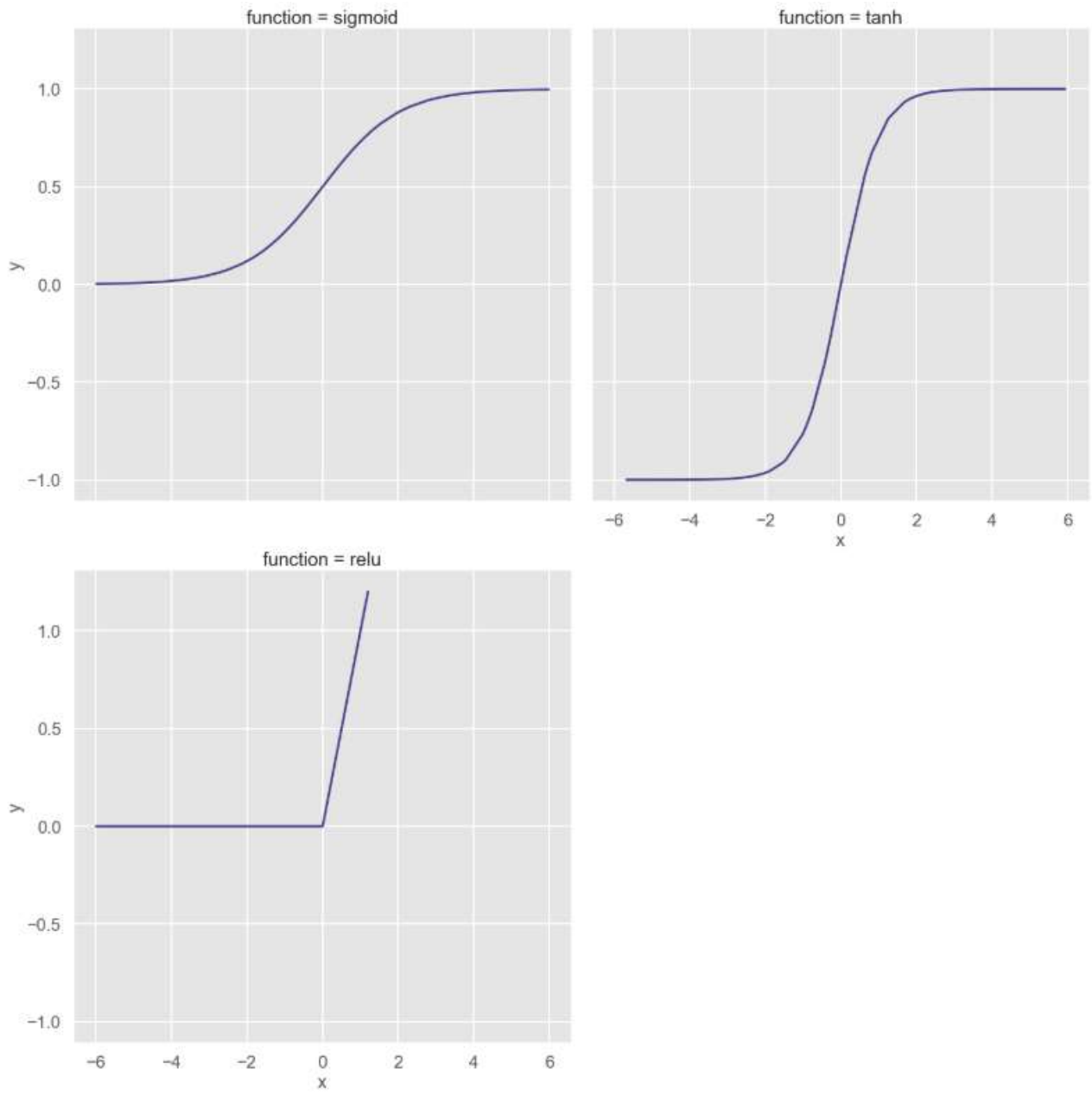

Рис.2 Графики основных функций активации

Зачастую выбор функции определен конкретной задачей. Так, например, в генеративно-состязательной сети в качестве функции последнего слоя генератора часто используется Tanh функция, а в задачах классификации – softmax. Тем не менее, есть ряд особенностей, работающих в пользу лишь некоторых функций из списка:

1) Сигмоидная функция практически не используется в последнее время из-за двух основных проблем: насыщения (vanishing gradient problem) и отсутствия

центрированности относительно нуля. Первая проблема возникает в результате того, что при очень многих значениях градиент функции стремится к нулю, и, в случае обратного распространения ошибки при умножении текущего градиента на общий, значение еще быстрее стремиться к нулю. Вторая же проблема ведет к тому, что в случае положительности входных значений, все градиенты весов будут либо положительны, либо отрицательны. Это может привести к нежелательной зигзагообразной динамике обновления весов.

2) Tanh функция, в отличие от сигмоидной, центрирована относительно нуля, однако при этом так же склонна к насыщению.
3) Relu – не центрирована, однако, существуют ее модификации, такие, как Leaky ReLU = $f(x) = \begin{cases} x, & x > 0 \\ 0.01 * x, & x \leq 0 \end{cases}$

Ещё одна крайне важная деталь внутренней реализации нейронных сетей – это алгоритмы регуляризации. В случае больших сетей, состоящих из большого числа входных параметров, слоев и нейронов в каждом из них, вероятность переобучения крайне высока.

Для борьбы с переобучением в нейронных сетях существует ряд подходов.

1) $L_2 -$, $L_1 -$ регуляризация. Ее основным принципом является наложение штрафа за слишком большие веса при определенных входных параметрах. $L_2 = \lambda \sum_\omega \omega^2$, $L_1 = \lambda \sum_\omega | \omega |$
2) Алгоритм «Dropout». Его суть состоит в том, что в процессе обучения с заданной вероятностью каждый нейрон может быть выключен, т.е. его выходной сигнал будет равен 0. Данная техника препятствует тому, чтобы определенные узлы нейронной сети приобретали излишнюю значимость, т.к. будет существовать вероятность их отключения.

3) Метод ранней остановки. Данный метод представляет собой интуитивный способ борьбы с переобучением. Его суть состоит в том, чтобы отложить часть тренировочной выборки. При этом все обучение будет проходить на основной выборке, а отложенная часть будет использоваться для валидации. Таким образом, валидационный набор данных не принимает непосредственного участия в обучении, поэтому предполагается, что ошибка на данной выборке будет хорошо оценивать ошибку и на новых входных данных – из тестового множества. Останавливать обучение в таком случае следует в тот момент, когда валидационная ошибка перестанет уменьшаться.
4) Кросс-валидация. Есть несколько модификаций данного подхода, но суть сводится к следующему:
   - Обучающая выборка разбивается на k непересекающихся множеств одинаковой мощности.
   - Далее выполняются k следующих операций: одно множество оставляют на валидацию, и модель обучается на остальных k-1 множествах.

Для данного класса задач существует принципиальная классификация нейронных сетей: по характеру обучения. Согласно данной классификации, нейросети делятся на два подмножества:

1) Обучаемые с учителем. Данный класс сетей предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой ожидаемый выход. Эти данные принято называть обучающей парой. Как правило, нейронная сеть обучается на некотором множестве подобных обучающих пар. Процесс обучения заключается в том, что для каждого входного значения считается ошибка между полученным и ожидаемым результатом. Далее, изменяются веса согласно выбранному алгоритму оптимизации, минимизирующему данную ошибку.
2) Обучаемые без учителя. Сети, решающие данный тип задач, наиболее приближены к своей биологической модели. Обучающее множество

состоит лишь из входных векторов; процесс обучения выделяет статистические свойства обучающего множества и классифицирует вектора.

Первый класс задач на данный момент наиболее распространен в прикладных задачах. Однако, этот класс имеет один существенный недостаток: как было описано выше, он предполагает наличие обучающих пар, т.е., входящих данных, для которых заранее известен верный ответ. Обучающие пары размечаются людьми, а учитывая огромные объемы тренировочных данных, необходимых для обучения сети, задача получения размеченных данных становится зачастую основной проблемой в области построения нейросетей.

1.2 История развития нейронных сетей

Впервые такая модель, как искусственный нейрон, была предложена в научной работе Уоррена Мак-Каллока и Уолтера Питца в 1943 году. В этой классической работе была описана логика вычислений в нейронных сетях на основе нейрофизиологии и математической логики. Ученые показали, что сеть, составленная из множества единичных нейронов и синапсических связей, теоретически способна выполнять любые вычисления. Их работа пробудила большой интерес исследовательского сообщества к данной области.

Следующей важной вехой был 1949 год, который ознаменовался выходом в свет книги Д. Хебба «Организация поведения», в которой дается толкование процессу обучения с точки зрения физиологии. Он предположил, что по мере обучения различным задачам, связи в мозге постоянно меняются. Знаменитый постулат обучения Хебба гласит, что эффективность переменного синапса между двумя нейронами повышается при многократной активации этих нейронов через данный синапс. Т.е., он первым предположил, что обучение сводится к

изменению силы синаптических связей, что эквивалентно увеличению веса связи искусственного нейрона.

Середина 50-х годов известна появлением перцептрона Розенблатта – первой модели однослойной нейронной сети, состоящей из датчиков, сигналы которых поступают в ассоциативные элементы, преобразующих данную информацию и передающих ее реагирующим элементам.

Далее последовал период спада в исследованиях нейронных сетей. Исследования нейронных сетей практически не развивались до тех пор, пока компьютеры не достигли больших вычислительных мощностей. Одним из важных шагов, стимулировавших дальнейшие исследования, стала разработка в 1975 году Вербосом метода обратного распространения ошибки, который позволил эффективно решать задачу обучения многослойных сетей

21-ый век – это время постоянных открытий и изменений в данной сфере. Появление достаточных мощностей и возможность распараллеливать вычисления на видеокартах позволила вдохнуть жизнь в некоторые забытые парадигмы и подходы. К примеру, в такие, как сверточные сети. Сейчас область нейросетевых технологий – одна из самых динамично развивающихся областей машинного обучения.

В 2014 г. Йеном Гудфеллоу была представлена работа под названием Generative Adversarial Networks(GAN), в которой он описывает совершенно новую архитектуру сетей. Ключевая же идея, заложенная в генеративные сети, описанные Гудфеллоу, была описана чуть раньше, в 2014 году, Ярославом Ганином в соавторстве с Виктором Лемпицким в статье "Unsupervised Domain Adaptation by Backpropagation". Генеративно-состязательные сети - яркий пример алгоритмов обучения без учителя; с их помощью можно имитировать любое распределение данных, в частности изображения. Генеративные сети являются основным инструментом решения поставленной в данной магистерской

диссертации задачи. С 2014 года это направление является одним из самых актуальных и динамично развивающихся направлений машинного обучения.

1.3 Задача генерации изображений.

Уже на протяжении долгого времени в области обучения искусственных нейросетей «с учителем» существует основная, пока не разрешимая проблема – это проблема данных. Чем больше обучающая выборка, тем лучше «предсказывающая» способность нейросети. В частности, в случае изображений, задача поиска миллионов репрезентативных изображений не всегда реализуема. В связи с этим, значеие такого алгоритма, который имел бы возможность автоматически генерировать изображения, сложно переоценить.

Основным инструментом решения подобных задач на данный момент являются генеративно-состязательные сети. Основной их идеей является наличие двух нейросетей, работающий в паре: генератор и дискриминатор. Дискриминатор на вход получает данные из двух источников: заранее подготовленной выборки с размеченными данными, а также вывод генератора. Основной задачей дискриминатора является определение того, является ли входящее сообщение реальным, либо оно поступило из генератора. Т.е., выходом дискриминатора будет являться бит. Соответственно, задача оптимизации дискриминатора – это задача минимизации его ошибки. Генератор же в свою очередь из случайного вектора генерирует данные, которые должны быть неправильно классифицированы дискриминатором. Данная схема изображена на рисунке 8.

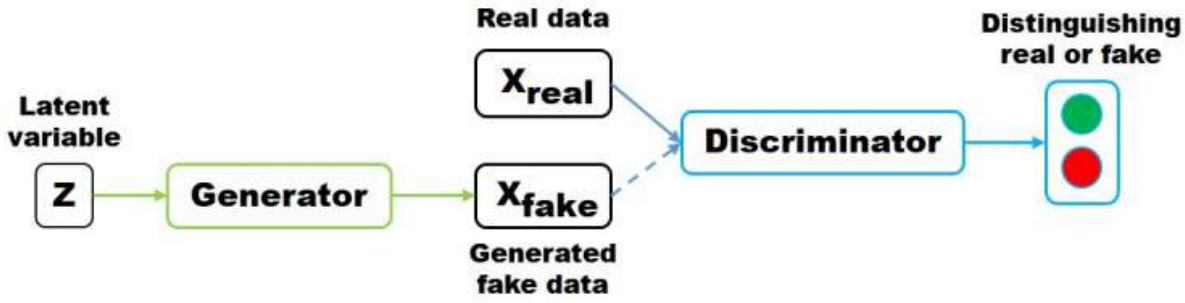

Рис. 8 - Схема работа GAN (Yongjun Hong)

Чтобы определить распределение данных, выдаваемых генератором $p_G(x)$, сперва вводится вектор шумов $p_z(z)$, затем вводится дифференцируемая функция отображения G(z, Θ_g), представляющая собой многоуровневый перцептрон с параметров Θ_g. Так же определяется второй перцептрон D(x, Θ_d), областью значений которого является скаляр. D(x) представляет собой вероятность того, что x принадлежит реальным данным, а не порожденными генератором p_G. Другими словами задача является классической минимакс игрой (Ian J. Goodfellow, 2014):

$$\min_G \max_D V(D,G) \tag{1}$$
$$= E_{x\sim p_{data}(x)}[\log \quad D(x)]$$
$$+ \; E_{z\sim p_z(z)}[\log \quad (1-D(G(z)))]$$

Где $E_{x\sim p_{data}(x)}[\log \quad D(x)]$ - кросс-энтропия ответов дискриминатора в случае реальных данных, а $E_{z\sim p_z(z)}[\log \quad (1-D(G(z)))]$ - в случае сгенерированных.

Таким образом, задача дискриминатора – это максимизировать данную функцию ошибки, генератора – минимизировать.

Можно заметить, что в случае минимизации выражения по функции генератора, первое слагаемое никак от генератора не зависит; таким образом оно

может быть опущено. Однако, как показывает практика, минимизации $E_{z\sim p_z(z)}[\ \log\ (1-D(G(z)))]$ не достаточно для успешного обучения генератора в виду того, что на ранних этапах обучения, когда дискриминатор с довольно большой уверенностью может отличить изображения от произвольного шума, $\log\ (1-D(G(z)))$ будет насыщаться. Решением будет обучение генератора путем максимизации $\log\ (D(G(z)))$ вместо минимизации $\log\ (1-D(G(z)))$. Данная функция позволяет получить гораздо больший градиент на начальных этапах обучения.

Суммируя все вышесказанное, итоговый алгоритм обучения генеративно-состязательной сети по мини батчам будем иметь следующий вид:

- На каждой итерации выбирается k элементов выборки $(x_1, x_2, \dots, x_k)$, а также генерируется k произвольных векторов определенной длины $(z_1, z_2, \dots, z_k)$, которые будут использованы в качестве входных параметры сети генератора.
- Для всех x_i, а так же z_i вычисляются $D(x_i)$, а так же $D(G(z_i))$
- Обновляем D, учитывая полученное значение градиента:

$$\nabla_{Q_d}\frac{1}{k}\sum_{i=1}^{k}(\log D(x_i)+\log(1-D(G(x_i))))$$

- Данные операции, обновляющие дискриминатор, в большинстве случаев производятся несколько раз, прежде чем обновить генератор.
- Обновляем генератор по вновь сгенерированным k векторам шумов.

$$\nabla_{Q_G}\frac{1}{k}\sum_{i=1}^{k}\log(1-D(G(x_i)))$$

Либо, как уже описано выше можно максимизировать $\log\ (D(G(z)))$

Несмотря на относительную простоту алгоритма, обучение генеративных сетей – крайне сложный и нестабильный процесс. Задача нахождения точки равновесия в данной минимакс игре усложняется с усложнением конкретного

случая применения GAN. Ян Гудфелоу писал, что ему на самом деле крайне повезло, что первые запуски генеративной сети, написанной им, принесли довольно осмысленные результаты. Потому что, учитывая всю нестабильность обучения, выбор иных гиперпараметров или структуры сети привел бы к невозможности получить достаточные для продолжения исследований результаты. Таким образом, успешный синтез изображений в высоком разрешении по-прежнему остается неуловимой целью. В рамках данной магистерской работы была поставлена задача реализации генеративных состязательной сети, позволяющей синтезировать достоверные изображения в достаточно больших разрешениях: 64x64 и 128x128 пикселей.

Будет проведен анализ ряда существующих успешных подходов и рекомендаций, на основе всего этого будет построен агрегированный алгоритм и проанализированы полученные результаты. Решение должно представлять собой разумный компромисс между качеством полученных изображений и скоростью обучения, однако первостепенной целью оптимизации будет являться качество изображений, получаемых на выходе работы сети.

1.4 Проблемы реализации GAN.

В задачах синтеза изображений генеративные сети испытали вторую волну успеха после внедрения GAN, основу которых составляли глубокие сверточные сети. (Alec Radford, 2016). Однако, использование сверточных сетей в рамках генеративных моделей привело к появлению специфических проблем.

1.3.1. Удаленные структурные зависимости.

Например, наиболее продвинутый ImageNet GAN (Miyato & Koyama, 2018) может достаточно реалистично генерировать изображения, на которых не

присутствуют сложные структуры: горы, океан, небо, при этом сгенерированная собака будет без ног, но с крайне реалистичной шерстью. Суть проблемы состоит в том, что в операция свертки учитывается лишь ограниченное количество стоящих рядом пикселей, а пиксели, которые находятся на достаточно большом расстоянии, практически не будут иметь никакой связи. Именно поэтому шерсть будет выглядеть достоверно, но понять то, что у собаки должно быть 4 лапы, алгоритму будет практически невозможно. Конечно, локальность связей можно регулировать размером ядра свертки и количеством конволюционных слоев, но это значительно увеличивает общее число параметров модели, что приводит к переобучению и в целом значительно усложняет работу оптимизационных алгоритмов.

Очевидно, что чем больше изображение, и чем дальше могут находиться друг от друга пиксели – тем сложнее генеративной сети синтезировать правдоподобное изображение. Типичные проблемы с геометрией на больших разрешениях легко заметить на изображениях, отраженных на Рисунке 9.

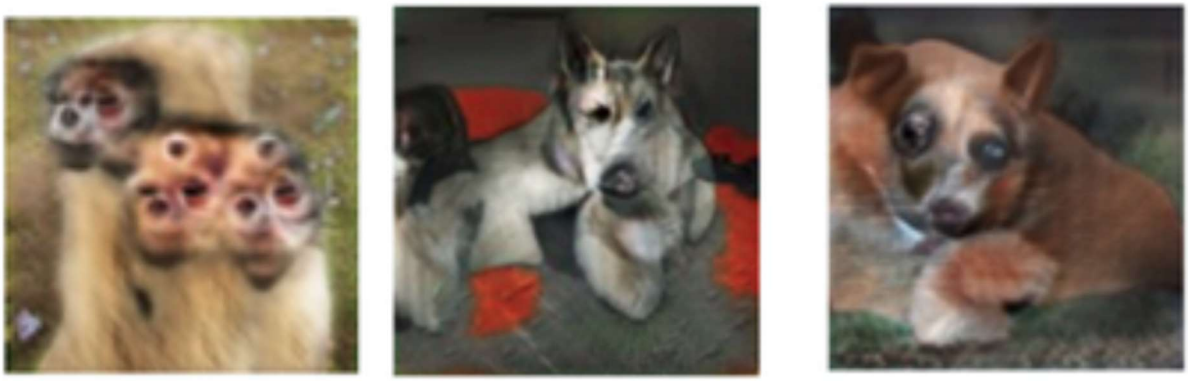

Рис. 9 – Сгенерированные изображения животных *(AIo)*

В 2014 году белорусским исследователем Дмитрием Богдановым был разработан «attention»-механизм (Dzmitry Bahdanau, 2015), который стал крайне популярным при реализации систем машинного перевода. В общих чертах, «attention»-механизм – это компонент архитектуры нейросети, который позволяет количественно определить зависимости в данных. Данное направление

набирало популярность, в 2016 году была предложена концепция self-attention (Cheng, 2016). Опять-таки первоначально данное понятие появилось в контексте задачи машинного перевода, однако в 2018 идея была перенесена и на задачу реализации генеративных сетей. (Han Zhang, 2018). Подобные сети получили название SAGAN, и модуль self-attention позволяет определить зависимости в разных частях изображения, далеко за пределами размера фильтра свертки. Схема модуля self-attention в генеративно-состязательной сети представлена на рисунке:

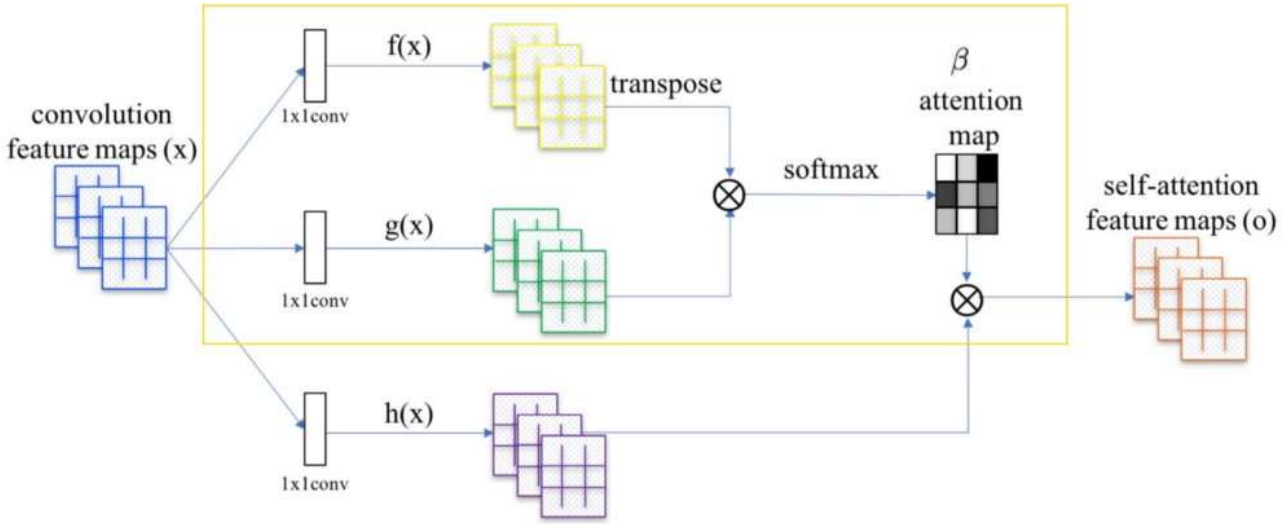

Рис. 11 – Модуль self-attention *(Han Zhang, 2018)*

Обработка модуля состоит из нескольких этапов. Сначала входной вектор предыдущего скрытого слоя $x \in R^{C \times N}$, где $N -$ размерность данных, $C -$ количество конволюционных фильтров предыдущего слоя,
трансформируется в два множества признаков f(x) = $W_f x$ и g(x) =$W_g x$ посредством применения 1x1 конволюций. Использование 1x1 конволюций обусловлено экономией памяти, выделяемой на хранение весов. В итоге, после применения softmax функции получаем «attention map» – матрицу, где каждый элемент представляет собой следующее значение:

$$\beta_{i,j} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}, \text{где } s_{ij} = f(x_i)^T g(x_j)$$

W_f и W_g - параметры модели, которые необходимо обучить. Таким образом, результирующая матрица показывает силу воздействия i-ого участка изображения на генерацию j-ого региона; тем самым принимаются во внимание пространственно-разделенные признаки. Матрицы весов W_f и W_g имеют размерности $R^{C^* \times N}$. Авторы метода (Han Zhang, 2018) ссылаются на то, что не было замечено какого-то существенного снижения качества результирующих изображения в случае уменьшения количества фильтров с C до C/k, где k = 1,2,4,8, поэтому для эффективности использования памяти предлагается выбирать k = 8.

В оригинальной работе, полученный с применением модуля, тензор умножается на некоторый обучаемый скаляр и сдвигается на значение входного тензора в модуле self-attention: $res = \gamma * y_{self-attentio} + x$. В своей реализации модуля self-attention я буду придерживаться оригинала.

В статье (Han Zhang, 2018) ничего не сказано про количество модулей и их размещение. В литературе были найдены реализации, в которых self-attention модуль использовался в самом конце слоев, после итогового слоя свертки. В реализации авторов статьи SAGAN используется только один модуль, но он располагается среди сверточных сетей. В данной работе будет исследована важность вопроса расположения self-attention модуля, а также будут проведены испытания работы сети, состоящей из нескольких таких модулей.

1.3.2 Проблема сходимости.

Одной из наиболее изучаемых тем в области GAN является тема влияния функции ошибки на обучение сети. Существует достаточное большое количество различных подходов, и на данный момент нет однозначного варианта при выборе

функции ошибки, однако, с уверенностью можно заявить, что ее неверный выбор порождает целый ряд проблем в обучении генеративно-состязательной сети.

В своей работе, описывающей такой тип сетей, как GAN, Ян Гудфеллоу продемонстрировал, что для фиксированного генератора существует оптимальный дискриминатор и он имеет вид (Ian J. Goodfellow, 2014):

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2}$$

В таком случае, подставляя это значение в функцию ошибки (1) можно получить:

$$\mathrm{C(G)} = \max_D V(G,D) = E_{x\sim p_{data}}\left[\log \frac{p_{data}(x)}{p_{data}(x)+ p_g(x)}\right] + E_{x\sim p_g}\left[\log \frac{p_g(x)}{p_{data}(x)+ p_g(x)}\right] \tag{3}$$

Это и есть критерий оценки генератора. Таким образом задача состоит в том, чтобы данную функцию минимизировать. В (Ian J. Goodfellow, 2014) доказывается теорема о глобальном минимуме (3), который достигается тогда и только тогда, когда $p_g = p_{data}$; при этом значение критерия в точке глобального минимума равно $-\log 4$. В процессе доказательства возникает важное следствие:

$$C(G) = -\log 4 + 2 * JSD(p_{data}||p_g) \tag{4}$$

Таким образом, задача обучения генератора сводится к задаче минимизации расстояния Дженсена-Шэннона между распределением реальных и сгенерированных данных. Этот факт открывает большие возможности для оптимизации обучения GAN, так как расстояние Дженсена-Шэннона можно обобщить до более широкого класса дивергенций (Sebastian Nowozin, 2016)

М. Аржовский провел полномасштабные исследования различных функций расстояния, таких как расстояние Кульбака-Лейблера, Дженсена-Шэннона, Вассерштейна, и показал на примере, что все они, за исключением расстояния Вассерштейна, в определенных случаях не приведут к сходимости

распределения генератора к распределению данных (Martin Arjovsky, 2017), что приведет к невозможности обучения генератора. На основе выводов, сделанных из данной работы, был предложен новый тип генеративной сети, в основе которого лежала минимизация расстояния Вассерштейна:

$$\mathrm{W}(P_G, P_D) = \min_{\gamma \in \prod(P_G, P_D)} E_{(x,y)\sim\gamma}\big[||x - y||\big]. \quad (5)$$

Данный тип сетей получил название WGAN. В своей реализации я так же буду использовать принципы WGAN, однако в уже обновленной реализации – согласно принципу спектральной нормализации. (Takeru Miyato, 2018). Связь спектральной нормализации и расстояния Вассерштейна вытекает из следующих утверждений. Согласно теореме двойственности Канторовича-Рубинштейна, (5) можно переписать в виде: $\mathrm{W}(P_G, P_D) = \max_{||f||_L \leq 1} E_{x\sim P_G}[f(x)] - E_{x\sim P_D}[f(x)]$, т.е. максимуму по всем 1-Липшицевым функциям (Han Zhang, 2018). В выкладках (Takeru Miyato, 2018) показано, что нормализация спектральной нормы матрицы весов σ(W) = 1 приведет к тому, что и спектральная норма функции f будет ограничена единицей, т.е. все функции, по которым будет оптимизироваться сеть, будут 1-Липшицевыми. Таким образом, процесс спектральной нормализации весов сводится к тому, чтобы для каждого слоя выполнять:

$$W_{SN} := \frac{W}{\sigma(W)} \quad (6)$$

Нахождение спектральной нормы требует разложения матрицы весов по сингулярным числам, что является вычислительно трудной задачей. Поэтому в оригинальной статье предлагается степенной итерационный метод (power iteration).

Таким образом, использование спектральной нормализации поможет GAN обучаться стабильнее.

1.4 Постановка задачи

Учитывая все вышесказанное, очевидно, что успешный синтез изображений в высоком разрешении по-прежнему остается неуловимой целью. Существует ряд алгоритмов, которые довольно успешно себя показали в подобного рода задачах. Однако при этом, обучения нейросетей согласно данных алгоритмов даже на самом производительном оборудовании является крайне трудоемкой задачей. Ввиду чего, в рамках данной магистерской работы была поставлена задача на базе представленных алгоритмов реализовать свою агрегированную генеративно-состязательную сеть для синтеза достоверных изображений в достаточно больших разрешениях. Из-за большой трудоемкости поставленной задачи, генерация изображений будет ограничена разрешением 64 пикселей. Решение должно представлять собой разумный компромисс между качеством полученных изображений и скоростью обучения, однако первостепенной целью оптимизации будет являться качество изображений, получаемых на выходе работы сети.

2. РАЗРАБОТКА АРХИТЕКТУРЫ СЕТИ ДЛЯ ГЕНЕРАЦИИ ИЗОБРАЖЕНИЙ

2.1 Разработка подхода на основе SAGAN

2.1.1 Решение «Mode collapse» проблемы.

Одна из самых распространенных проблем при обучении – это «mode collapse». Результатом подобной проблемы является то, что генератор всегда синтезирует одинаковые, или практически одинаковые изображения. Такое происходит, в частности, когда дискриминатор запаздывает с обучением. В таком случае генератор находит некое оптимальное изображение, которое всегда обманывает дискриминатор. В результате, вне зависимости от входного вектора шумов z, генератор будет синтезировать одно и то же изображение. В связи с этим, все последние исследования сходятся к тому, что дискриминатор необходимо обучать быстрее, чем генератор. Это интуитивно понятно, так как сеть распознавания сначала необходимо обучить каким-то шаблонам, прежде чем предлагать ей распознавать сгенерированные изображения. Данные рассуждения привели к введению правила TTUR (two time-scale update). В статье (Martin Heusel, 2018) приведено доказательно влияния подобного подхода на сходимость к точке равновесия по Нэшу минимакс игры дискриминатора и генератора.

2.1.2 Архитектура сети SAGAN

В основе сети лежат три основные модуля – модуль «self-attention», модуль генератора и модуль дискриминатора. «Self-attention» входит в качестве дополнительного слоя, как в генератор, так и в дискриминатор. Сами же модули дискриминатора и генератора построены на основе глубоких сверточных сетей с

разницей в том, что в генераторе использованы обратные сверточные слои. Также для повышения стабильности обучения, после каждого сверточного слоя был использован слой нормализации по батчам.

Архитектура сети изображена ниже на рис. 12:

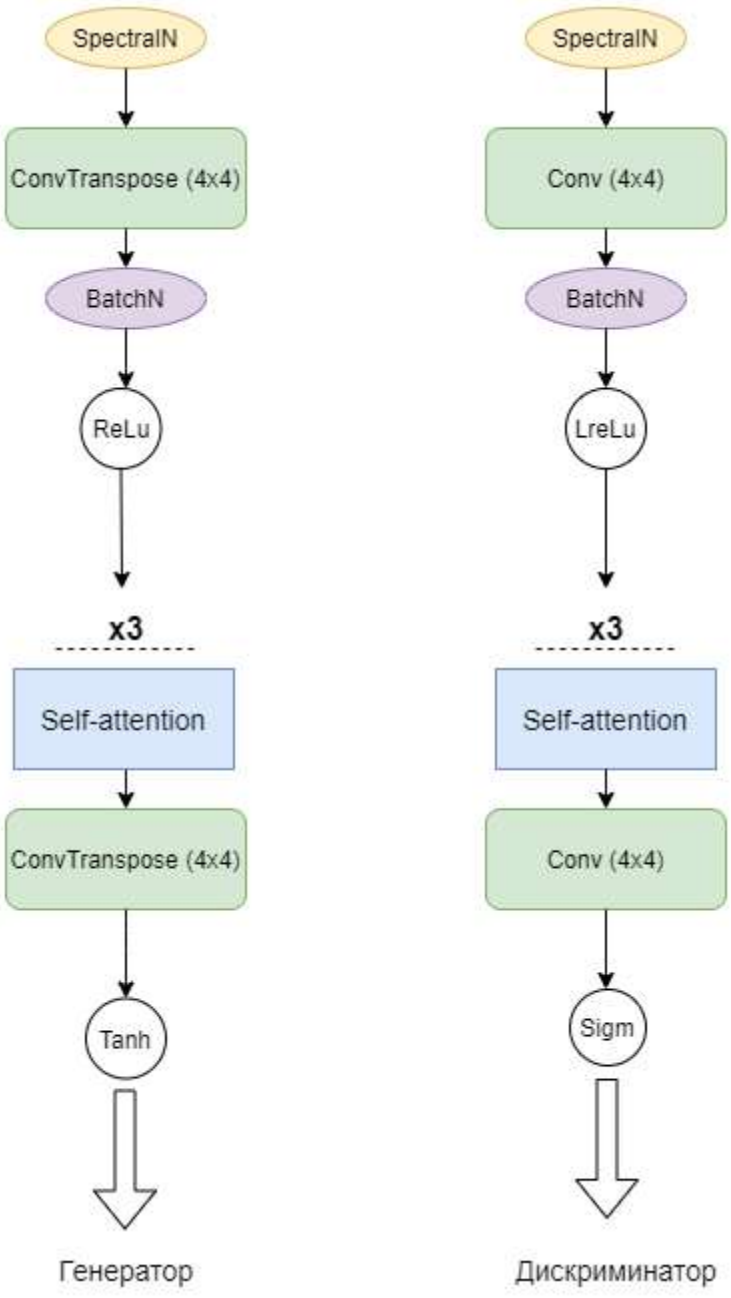

Рис. 12 – Схема сетей в реализации SAGAN

Как видно из рисунка, сети практически симметричны.

2.1.3 Дополнительные механизмы оптимизации

1) Сглаживание меток при подсчете значения ошибки обучения (например, вместо 1 будет использоваться случайное число из отрезка [0.8, 1]), что минимизирует обнуление градиента для генератора, т.е. стабилизирует процесс обучения. Во многих работах советуют использовать сглаживание с одной стороны.
2) Вместо «pooling»-слоев были использованы конволюции со сдвигом (strided convolutions), (Alec Radford, 2016)
3) Использована нормализация по батчам для генератора (данный подход характерен для SAGAN сети.), а также активационная функция Relu для генератора и LeakyRelu для дискриминатора.
4) Использование оптимизатора Adam.

2.3 Разработка подхода на основе PGGAN

В результате обучения SAGAN на разрешении 64 x 64 возникли определенные сложности, которые будут описаны в подробностях в следующей главе. Это показывает, что классический GAN, даже при условии определенных улучшений крайне, сложно обучить синтезу изображений в достаточно больших разрешениях. Таким образом, необходимо некое изменение в самом процессе обучения. В 2018 году ряд исследователей из NVIDIA представили новый тип генеративных сетей под названием Progressive Growing GAN (Tero Karras, 2018). В магистерской работе была так же поставлена задача реализации алгоритма на базе данного алгоритма. Суть это алгоритма состоит в изменении самой методологии обучения генеративной сети, в которой сначала обучение проводится для изображений в небольших разрешениях, начиная с 4-х пикселей. Затем, когда сеть достаточно обучилась на низком разрешении, происходит

динамическое добавление новый слоев, что позволяет увеличить размерность входящего изображения; таким образом, новый этап обучения происходит для изображений удвоенной размерности. Процесс поэтапного удваивания разрешения происходит до того момента, когда будет достигнуто требуемое разрешение. Такой подход позволяет сети сперва изучить сложные структурные формы на маленьких разрешениях, а затем постепенно их детализировать. Это также позволяет значительно уменьшить время обучения, т.к. сеть растет поэтапно. Схема работы сети PGGAN представлена на Рис.13.

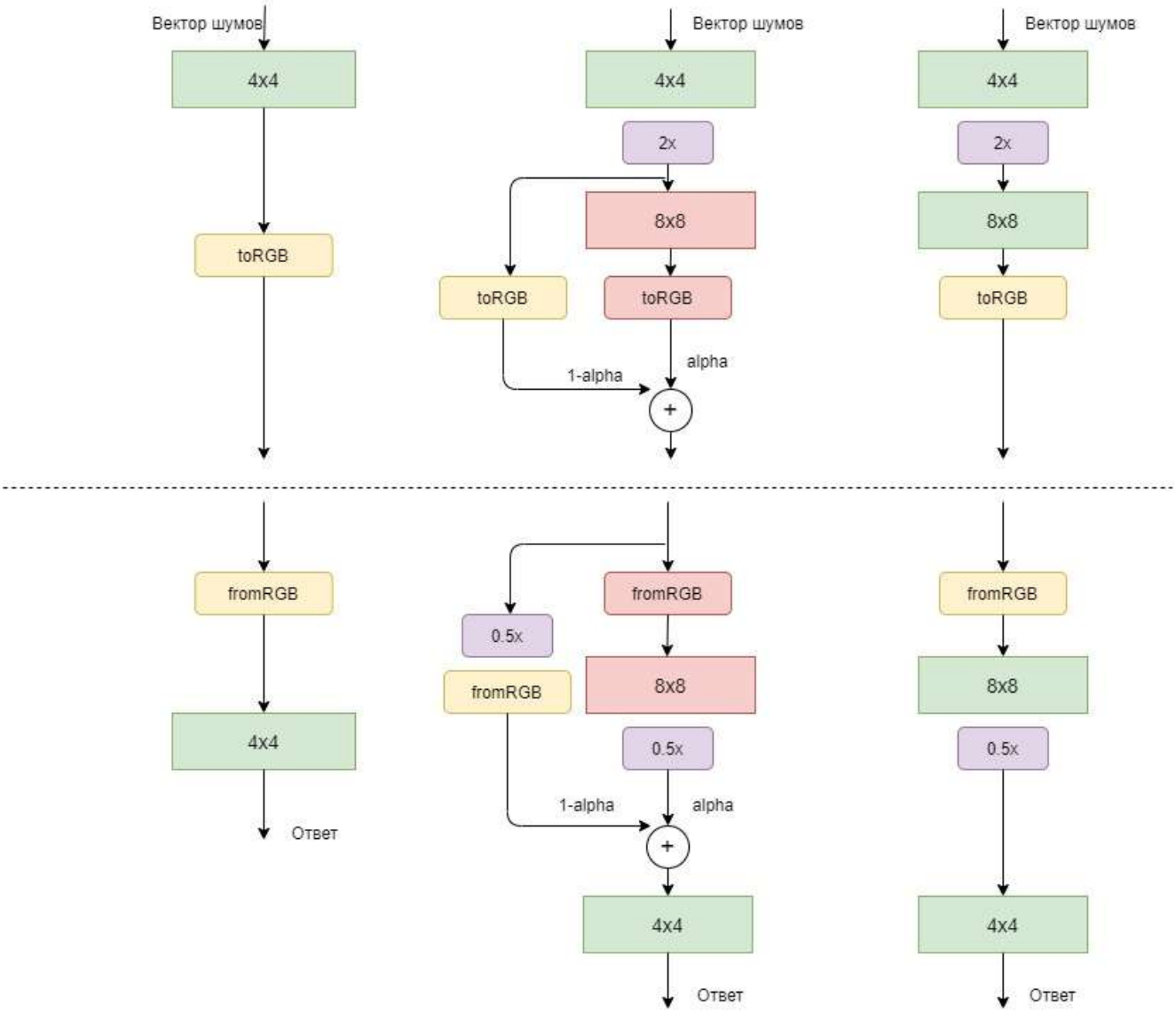

Рис. 13 – Схема работы PGGAN

В процессе обучения можно выделить две основные части: этап адаптации роста и этап стабилизации.

Процесс обучения начинается с этапа стабилизации –на этом этапе сеть обучается на изображениях с минимальных разрешением. Авторы оригинальной статьи на каждый этап выделяли 800k итераций. Далее сеть динамически расширяется добавлением нового блока. В качестве блока авторы используют два Conv 3 x 3 слоя. Таким образом, происходит добавление нового блока в обе сети: генератора и дискриминатора. При этом, с уже работающими блоками они связываются через upsample/downsample слой для генератора и дискриминатора соответственно. Однако, перестройка сети происходит постепенно, благодаря этапу адаптации роста. Реализуется это благодаря введению параметра alpha, который определяет, какой объем данных добавленного блока будет использован в сети. Для вывода «старой» части сети, этот параметр будет равен 1-alpha. В самом начале фазы адаптации alpha равен 0, т.е. для сети ничего не изменилось с момента добавления нового блока. С каждой итерацией параметр alpha равномерно увеличивается. После этапа адаптации, когда весь поток данных идет через новый слой, сеть «склеивается», т.е. удаляются уже ненужные ветви, поддерживающие старую структуру.

Данная сеть также использует «minibatch discrimination» для увеличения вариабельности синтезируемых изображений, а также спектральную нормализацию и нормализацию по батчам. Авторы статьи предлагают использовать нормализацию по пикселям, однако я отдал предпочтение спектральной.

2.3 Оценка качества результатов.

Важный момент, на котором необходимо сделать акцент – это правдоподобность изображений. Мнение конкретного индивидуума нельзя назвать корректной метрикой, поэтому в данной работе я буду пользоваться общепринятыми показателями. В настоящий момент наиболее популярны такие метрики как Inception score (IS) и Frechet Inception distance (FID). В своей работе я буду использовать FID для оценки правдоподобности изображений.

Для вычисления FID вновь используется заранее обученная сеть для извлечения признаков из промежуточный уровней. Затем моделируется распределение данных для этих признаков с использованием многомерного распределения Гаусса со средним значением μ и ковариацией Σ. FID между реальными изображениями *x* и сгенерированными изображениями *g* вычисляется по слудующей формуле:

$$\mathrm{F}ID(x,g) = ||\mu_x - \mu_g||_2^2 + Tr\left(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}}\right), \text{где } Tr$$

$$- \text{след матрицы, т. е. сумма всех диагональных элементов}$$

Чем ниже FID, тем выше качество сгенерированных изображений.

Также стоит отметить, что официальное значение оценки качества генератора возможно лишь при использовании официальных программных пакетов, в частности, это реализация tensorflow для получения оценки FID. В данной работе была использована реализация на pytorch (https://github.com/mseitzer/pytorch-fid). Ее значения могут иметь некоторую погрешность порядка сотых или тысячных.

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ЭКСПЕРИМЕНТЫ.

3.1 Описание аппаратного обеспечения.

Поставленная в магистерской работе задача решалась при помощи программной платформы разработки Pytorch, котораяй является реализацией известной программной платформы Torch для языка программирования Python. Учитывая, что обучение генеративно-состязательных сетей является вычислительно сложной задачей, обучение сети на процессоре превратило бы проблему в практически нерешаемую. Таким образом, обучение всех сетей происходило на видеокарте используя CUDA SDK (CUD). В рамках данной магистерской работы была использована одна из наиболее мощных на данный момент видеокарт NVIDIA TESLA K80 (NVI) с 12Гб памяти. Упомянутая инфраструктура была развернута на виртуальном сервере Google Cloud. В качестве интегрированной среды разработки был использован jupyter notebook (jup), установленный на виртуальный сервер. Аналитика данных с различными графиками была произведена в пакете Tensorboard (Ten)

3.2 Используемые данные и их подготовка.

В качестве набора данных, на которых производилось обучение были использованы следующие два варианта: фотографии лиц знаменитостей в разрешении 64 пикселей и LSUN датасет (LSU) из категории церквей в разрешении также 64 пикселей.

Для подготовки данных был реализован класс Loader, являющийся оберткой для стандартного torch.utils.data.DataLoader. Помимо предоставления интерфейса torch.utils.data.DataLoader, Loader класс реализует ряд необходимых трансформаций над изображениями:

1) Масштабирование изображения (в случае PGGAN). Данная трансформация необходима для того, чтобы обучать на различных разрешениях: от 4 до 64 пикселей. В качестве типа интерполяции использовался алгоритм поиска «ближайшего соседа»

2) Перевод изображения в тензор – объект, которым оперирует программная платформа pytorch.

3) Нормализация. Т.к. в сети генератора мной часто использовалась активационная функция тангенса в качестве функции последнего слоя, выход сети генератора в таком случае был в диапазоне [-1,1], а соответственно и поступающие на вход дискриминатора реальные изображения должны быть из этого диапазона.

Также стоит упомянуть, что я не использовал такую полезную операцию, как перемешивание данных при получении («shuffle»), так как эта операция крайне сильно увеличивало время работы алгоритма.

3.3 Примитивная модель генеративно-состязательной сети.

В качестве подтверждения сложности поставленной задачи была реализована классическая модель GAN для известной задачи генерации лиц знаменитостей в разрешении 32x32, что ниже желаемого разрешения. Сеть построена по классическому шаблону Deep convolutional GAN(DCGAN).
Однако, даже при небольшом разрешении человеческий глаз легко распознает синтезированное изображение:

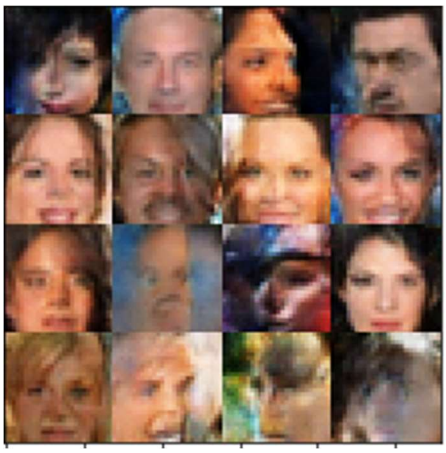

Рис. 14 – Сгенерированные изображения лиц

Метрика FID для оценки качества сгенерированных изображений в данном случае равна 102.35.

3.4 Особенности программной реализации.

В структуру сетей входят 4 основных модуля: загрузчик данных, которой уже был описан выше, класс генератора, класс дискриминатора, а также тренировочный модуль, в котором происходят все итерации и обновление весов сетей.

Классы генератора и дискриминатора наследуют стандартный класс torch.nn.Module. Учитывая, что в структуре сетей чередуются повторяющиеся модули, при реализации удалось вынести данный функционал в отдельную функцию. Пример представлен ниже на рисунке:

```
def conv_block(self, first):
    block = []
    first_cov, second_cov = g_sizes[self.current_s]

    if first:
        block.append(nn.ConvTranspose2d(self.latent_size, first_cov, 4))
        block.append(nn.BatchNorm2d(first_cov))
        block.append(nn.LeakyReLU(0.2))
    else:
        block.append(nn.Conv2d(first_cov, first_cov, 3,padding = 1))
        block.append(nn.BatchNorm2d(first_cov))
        block.append(nn.LeakyReLU(0.2))

    block.append(nn.Conv2d(first_cov, second_cov, 3,padding = 1))
    block.append(nn.BatchNorm2d(second_cov))
    block.append(nn.LeakyReLU(0.2))
    return nn.Sequential(*block)
```

Рис. 15 – Пример выноса блока сети PGGAN в функцию.

Замечание относительно тренировочного модуля: его задача - получение значения ошибки сети генератора и сети дискриминатора (на реальных и сгенерированных данных) и выполнение обновления весов согласно полученным значениям градиентов. Также данный модуль собирает статистику обучения в Tensorboard (Ten).

3.4.1 Реализация сети SAGAN.

В сети SAGAN есть ряд уникальных деталей, в частности «self-attention» модуль, реализация которого наследует torch.nn.Module и содержит несколько 1x1 сверточных слоев со своими весами, а также выполняет ряд матричных умножений. Код данного модуля представлен ниже на рисунке.

```
class attn(nn.Module):
    def __init__(self, insize):
        super(attn,self).__init__()
        self.insize = insize
        self.g_conv1to1 = nn.Conv2d(insize, insize//8, 1, 1, 0)
        self.f_conv1to1 = nn.Conv2d(insize, insize//8, 1, 1, 0)
        self.h_conv1to1 = nn.Conv2d(insize, insize, 1, 1, 0)
        self.gamma = nn.Parameter(torch.zeros(1))

    def forward(self, input):
        batchsize,channels,width ,height = input.size()

        f = self.f_conv1to1(input)
        g = self.g_conv1to1(input)
        h = self.h_conv1to1(input)

        f_processed = f.view(batchsize,-1, width*height).permute(0,2,1)
        g_processed = g.view(batchsize,-1, width*height)

        attention_matrix = \
        torch.nn.functional.softmax(torch.bmm(f_processed, g_processed \
                                              ),dim =-1)
        h_res = torch.bmm(h.view(batchsize,-1, width*height), attention_matrix.permute(0,2,1))
        h_res = h_res.view(batchsize, channels, width, height)
        return self.gamma* h_res + input
```

Рис.16 – Модуль «self-attention»

3.4.2. Реализация сети PGGAN.

Процесс обучения сети PGGAN коренным образом отличается от обучения классических генеративно-состязательных сетей тем, что он состоит из нескольких фаз. Таким образом тренировочный модуль выглядит уже иным образом:

```
d_optimizer = optim.Adam(D.parameters(), lr = lr, betas=(0.5, 0.999))
g_optimizer = optim.Adam(G.parameters(), lr = lr, betas=(0.5, 0.999))

while current_ext <= max_ext:
    step(g_optimizer, d_optimizer, False, D, G, loader)

    if current_ext < max_ext:
        reset_optimizer(g_optimizer, G)
        reset_optimizer(d_optimizer, D)

        D.extend()
        D = D.to(device)
        G.extend()
        G = G.to(device)
        current_ext *= 2

        loader = Data_Loader('celebs', current_ext, batch_size).loader()
        step(g_optimizer, d_optimizer, True, D, G, loader)

        D.stabilize()
        D = D.to(device)
        G.stabilize()
        G = G.to(device)
        reset_optimizer(g_optimizer, G)
        reset_optimizer(d_optimizer, D)
    else:
        break

step(g_optimizer, d_optimizer, False, D, G, loader)

torch.cuda.empty_cache()
```

Рис. 17 – Тренировочный модуль сети PGGAN

Как видно из кода, представленного на рис. 13, сначала отрабатывается фаза стабилизации, после чего происходит рост сети и наступает обучение в режиме адаптации и т.д. При этом, после каждого этапа очищаются градиенты оптимизаторов. В самом конце, как предлагается в оригинальной статье, выполняется заключительный повторный этап стабилизации. Еще одним ключевым моментом является тот факт, что для сети PGGAN плохо работает правило TTUR, т.е. скорость обучения необходимо использовать одинаковую, как для генератора, так и для дискриминатора.

Так же для сетей PGAN появилось два метода, реализующие адаптивный и стабилизационный проход:

```
def forward_adaptive(self, x, alpha):
    current_output = self.current_blocks(x)
    new_block_output = self.new_block(current_output)
    x = alpha * (self.new_to_rgb(new_block_output)) + (1.0-alpha) * self.current_to_rgb(current_output)
    return x

def forward(self,x):
    return self.current_to_rgb(self.current_blocks(x))
```

Рис. 18 – Два этапа в обучении сети.

Также стоит отметить, что выбор количества итераций зависит от разрешения. Был введен коэффициент, позволяющий дольше обучать большие разрешения.

3.5 Результаты экспериментов

На первом этапе мной была обучена сеть на наборе данных, состоящем из изображений лиц знаменитостей в разрешении 32 на 32 пикселя. Выше уже был приведен пример лиц, сгенерированных тривиальной DCGAN без каких-либо оптимизаций. В случае оптимизированной сети, результат выглядит более качественно даже на небольших периодах дискретизации (эпохах). Значение FID для сгенерированных изображений равно 80.91, в случае же обычной сети это значение было выше 100 (изображение на рисунке ниже кажется размытым, потому что каждое изображение масштабировано для разрешения 64 x 64):

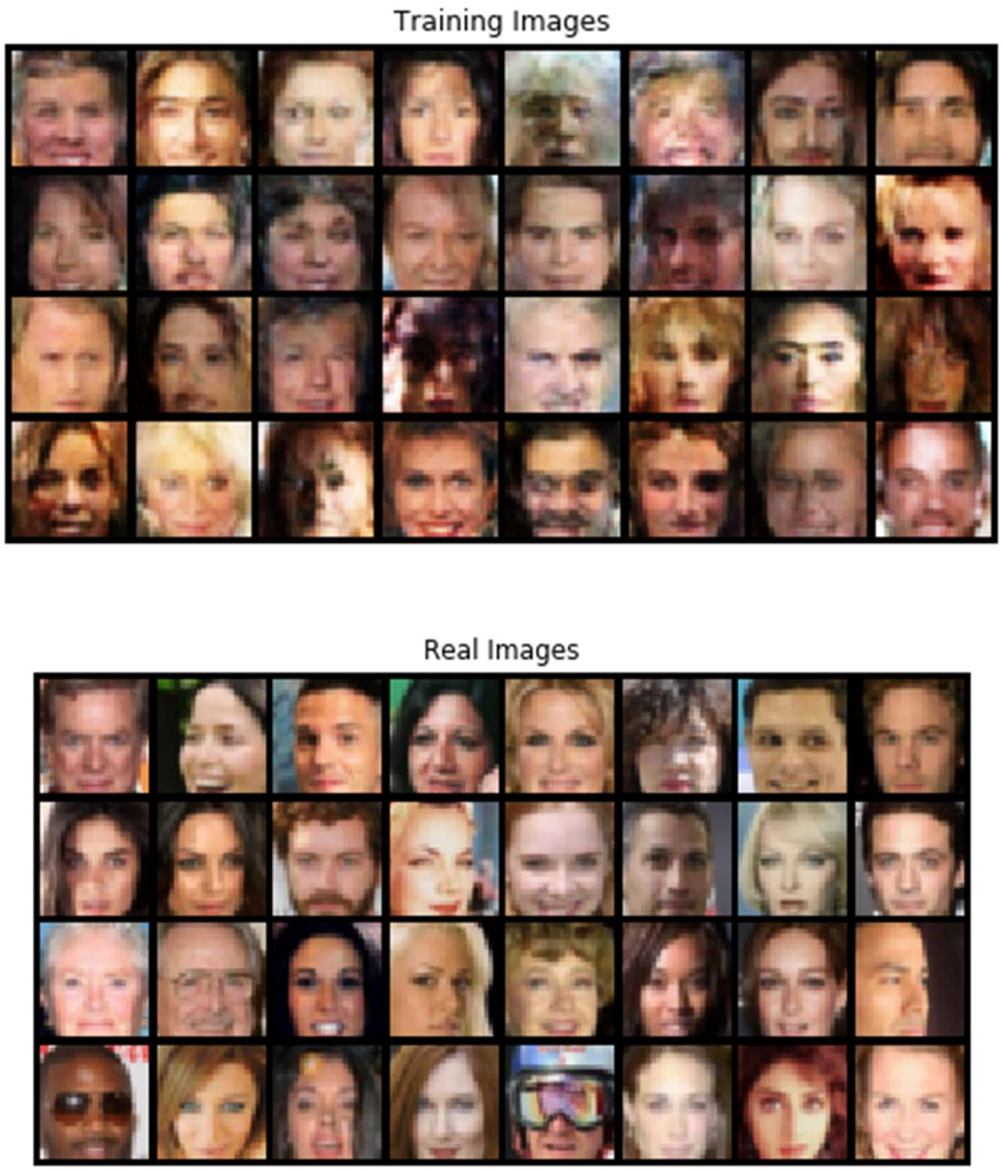

Рис 19 – Изображения лиц, полученные в результате работы сети SAGAN и примеры реальных изображений лиц.

Как уже описывалось выше, в большинстве работ, посвящённых генеративным сетям, используется скорость обучения (lr, «learning rate») равная 0.0001. В моих экспериментах были использованы различные значения, и значение, равное 0.0001 приводило к лучшим показателям работы сети. На

графике ниже изображено изменение ошибки генератора для двух вариантов learning rate: 0.0001, 0.0002.

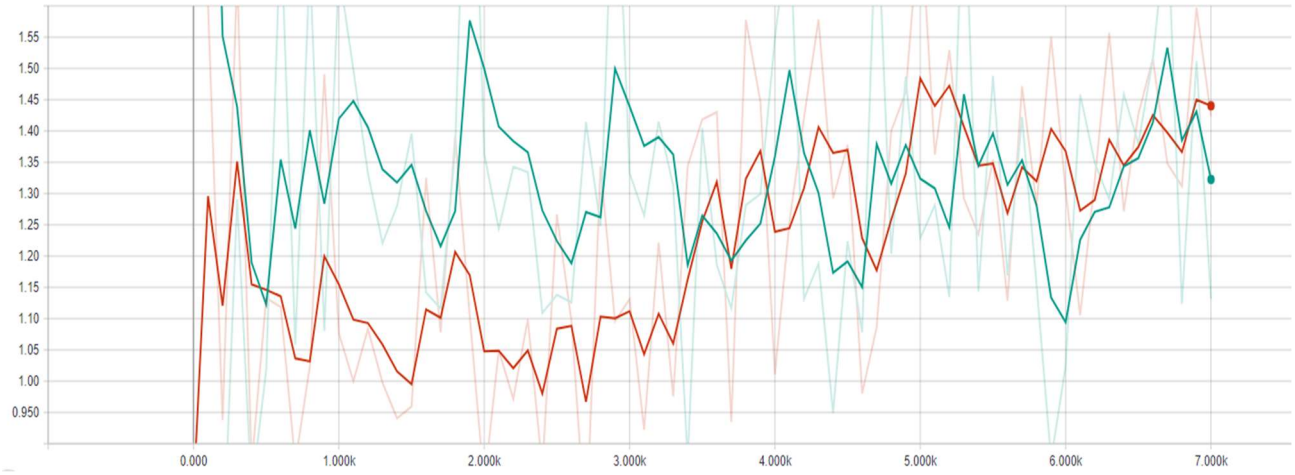

Рис. 20 – Зеленым цветом изображена ошибка для lr = 0.0001, красным – 0.0002

Из графиков на рисунке 12 может показаться, что график, соответствующий lr = 0.0001 - более гладкий, соответственно и обучение происходит предсказуемо, однако наличие всплесков – это просто признак так называемой смены моды. Т.е., это признак вариабельности синтезированных изображений. При скорости обучения lr = 0.0002 крайне легко достигнуть mode collapse. Далее представлен график изменения ошибок сети генератора и дискриминатора для lr = 0.0001:

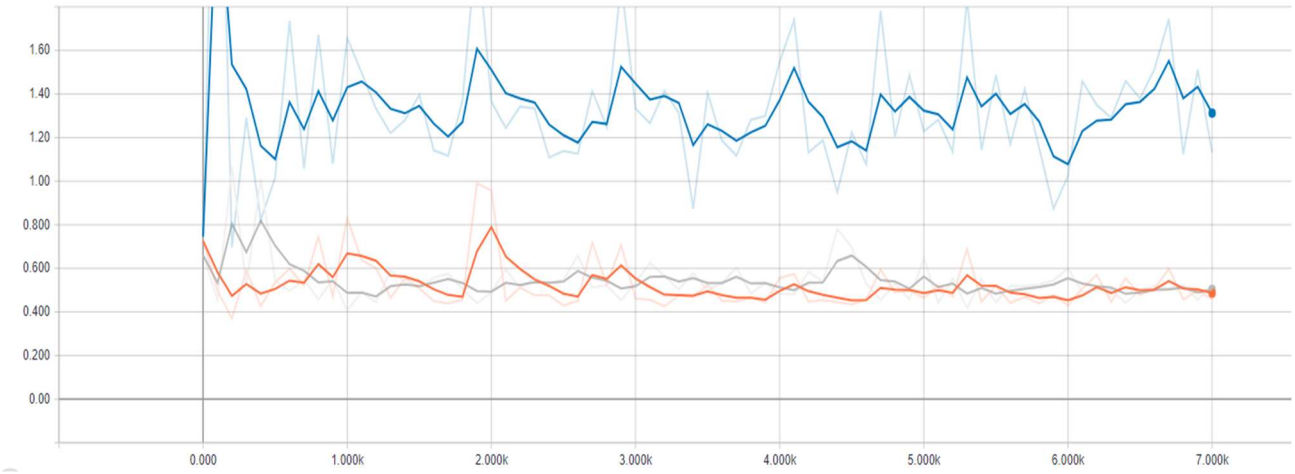

Рис. 21 – График изменения ошибок генератора и дискриминатора при фиксированном lr.Синий график – генератор.

Как видно из графика, оба параметра, ошибка дискриминатора на реальных данных и ошибка дискриминатора на данных генератора, не демонстрируют значительных колебаний и не колеблются вблизи нулевых значений, что говорит о стабильном процессе обучения.

Учитывая все проведенные мной эксперименты, лучше всего себя показала скорость обучения lr=0.00009. На рисунке ниже представлены аналогичные графики ошибок для различных значений lr [0.00008, 0.00009,0.0001]:

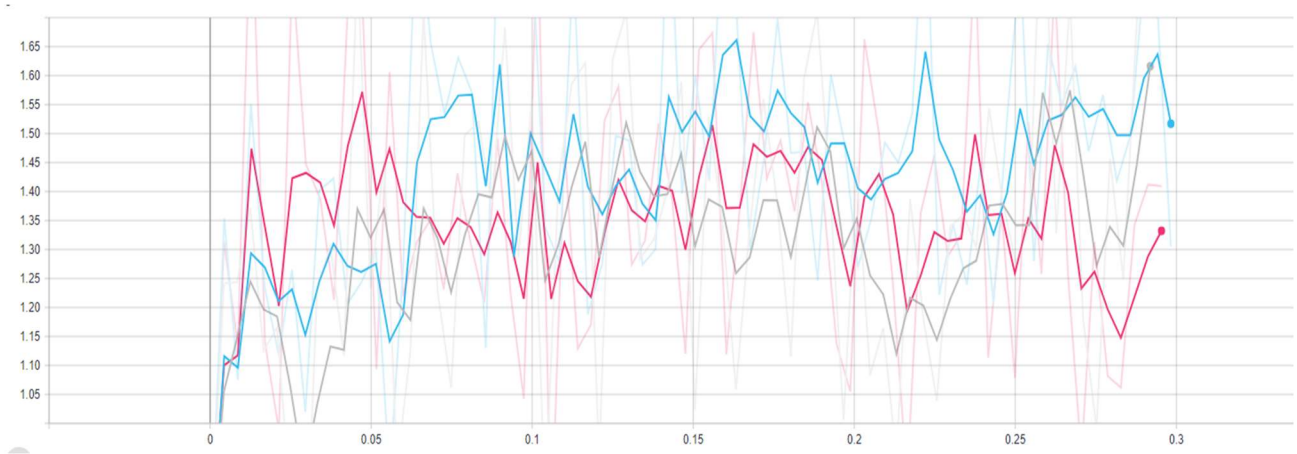

Рис. 22 – Красным цветом представлены значения ошибки при LR=0.00009, синим – 0.0001

Стоит обратить внимание, что на таких разрешениях влияние слоя self-attention минимально в виду того, что разрешение не велико. При этом, наличие спектральной нормализации серьёзно влияет на качество обучения сети. Ниже представлен график изменения ошибки генератора при наличии и отсутствии спектральной нормализации. Как видно, в случае ее отсутствия ошибка генератора довольно быстро растёт:

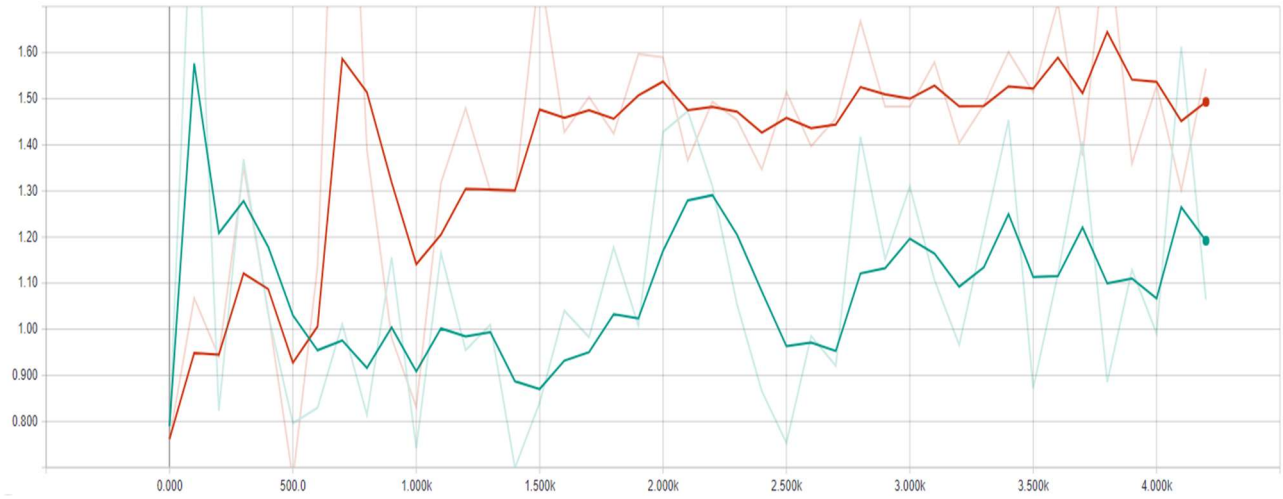

Рис. 23 – Ошибка генератора в случае наличия (зеленый) и отсутствия (красный) спектральной нормализации.

3.5.1. Проблемы метода на базе SAGAN

При переходе к разрешению 64 x 64 возникли проблемы в работе сети. Основной сложностью была ситуация «mode collapse». Стандартные пути решения данной проблемы – такие, как уменьшение значения lr, оказались неэффективны.

Рис.24 – «Mode collapse» в случае синтеза изображений лиц

Существует техника, которая в некоторых случаях, позволяет справиться с подобной проблемой: дискриминация по мини батчам (minibatch discrimination). Суть метода сводится к тому, что в момент прохождения данных по сети, учитываются не только один конкретный элемент, а все элементы минибатча. Также подсчитывается новая статистика: мера похожести изображений, поступающих на вход. Таким образом сеть учится “штрафовать” случаи, когда на вход поступает множество похожих изображений, а именно это и происходит в случае mode collapse. Данная техника улучшила качество генерируемых изображений в разрешении 32 x 32, однако не смогла существенно повлиять на способность данной сети синтезировать изображение в разрешении 64 x 64.

В качестве эксперимента были так же синтезированы изображения для известного набора данных LSUN категории церквей. В данном примере проблемы «mode collapse» не наблюдалось, однако, при этом качество полученных результатов оказалось крайне низким. Метрика FID не была использована, потому что даже невооруженным взглядом легко отличить синтезированное изображение от реального:

Рис. 25 – Примеры изображений церквей, полученных SAGAN на наборе данных LSUN

3.5.2. Обучение PGGAN

Тренировка сети в данном случае занимает значительное время: в случае 300k итерация на каждую стадию примерное время выполнения 4 часа, учитывая то, что все расчеты велись на производительной видеокарте (NVI). Соответственно, мной не проводились эксперименты на заявленных авторами 800k итерациях. В результате в первых экспериментах мной были получены следующие изображения:

Рис. 26 – Результат работы PGGAN

Сгенерированные изображения все еще довольно легко отличить от реальных изображений, однако их качество заметно лучше, чем качество изображений, синтезированных с помощью SAGAN.

График процесса обучения представлен ниже на картинке:

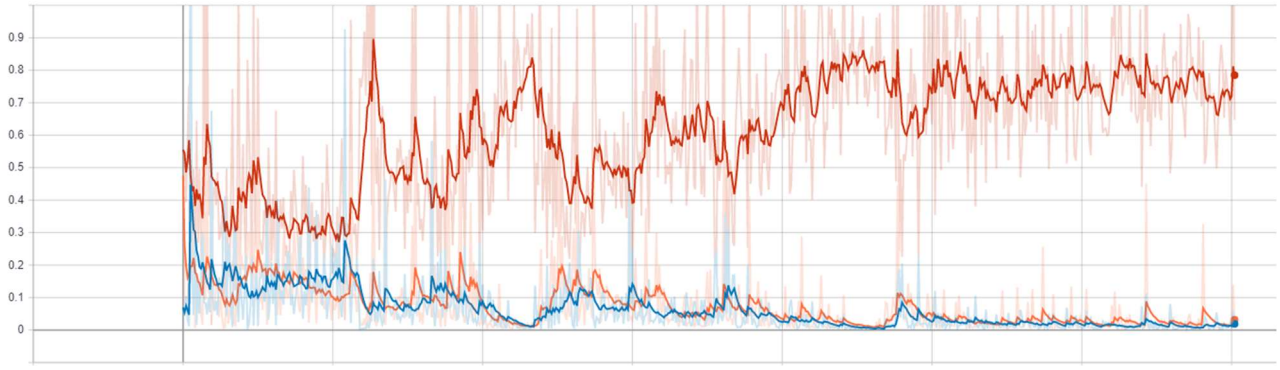

Рис. 27 – Изменение ошибки генератора (красная линия), а также дискриминатора при обучении PGGAN

Как видно из графика, при переходе к новому разрешению, ошибка генератора увеличивается. Тут важно «удержать» уменьшение ошибки дискриминатора и увеличение ошибки генератора, пока не будет достигнуто требуемое разрешение.

Данный тип сетей (PGGAN) требует более досконального погружения и исследования. Возможно, при правильном подборе гиперпараметров, функций ошибки, а также алгоритмов нормализации и иных оптимизаций, сгенерированные с помощью PGGAN изображения будет крайне сложно отличить от настоящих. В рамках данной работы подобные исследования проведены не были ввиду крайне серьезных временных и аппаратных затрат на проведение экспериментов.

3.5.3 Функция ошибки

В рамках данной магистерской работы не было проведено доскональное исследование влияния функции ошибки. В большинстве случаев использовалась стандартная «adversarial loss» из torch.nn.BCELoss (в случае SAGAN), либо среднеквадратичная ошибка (в случае PGGAN). При этом, для PGGAN была так же вычислена ошибка Вассерштейна, однако она не показала себя должным

образом: на достаточно близких эпохах ошибка дискриминатора начинала стремительно уменьшаться, при этом ошибка генератора росла. Стоит также упомянуть, что в обоих случаях я использовал одностороннее сглаживание целевых значений дискриминатора.

ЗАКЛЮЧЕНИЕ

В рамках данной магистерской работы были проанализированы новейшие методы для синтеза изображений в больших разрешениях. Было изучено большое число оптимизационных техник дня построения нейросетей синтеза изображений. На основе последних исследований SAGAN и BigGAN был реализован алгоритм генеративной состязательной сети. Экспериментальным путем были получены гиперпараметры, позволяющие наиболее стабильно обучать подобную сеть. В качестве результата работы данной сети были получены качественные изображения в разрешении 32 пикселей и вычислена соответствующая метрика FID, показавшая лучшие результаты при использовании классической DCGAN. Однако, также было показано, что при генерации изображений более высоких разрешений у сети на основе алгоритма SAGAN возникают проблемы с качеством результирующих изображений. В качестве контрмеры был проанализирован новый алгоритм генеративных сетей, реализующий иную методологию обучения - PGGAN. На основе данного подхода была реализована сеть, значительно превосходящая SAGAN по качеству генерируемых изображений в больших разрешениях.

Также был предложен ряд направлений для дальнейшего изучения, которые потенциально могли бы значительно улучшить качество и эффективность работы GAN.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. AI of the future: ‘Generative adversarial networks’ (GANs) [Online] // Infolob. - https://www.infolob.com/gans-future-ai-generative-adversarial-networks/.

2. Alec Radford Luke Metz, Soumith Chintala Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [Journal]. - 2016.

3 .Andrew Brock Jeff Donahue, Karen Simonyan LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS [Journal]. - 2019.

4. Castrounis Alex Artificial Intelligence, Deep Learning, and Neural Networks, Explained [Online]. - https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html.

5. Cheng J., Dong, L, Lapata, M Long short-term memory-networks for machine reading. [Journal]. - 2016.

6. CUDA [Online] // wikipedia. - https://ru.wikipedia.org/wiki/CUDA.

7. Dzmitry Bahdanau KyungHyun Cho, Yoshua Bengio NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE [Journal]. - 2015.

8. Han Zhang Ian Goodfellow, Dimitris Metaxas, Augustus Odena Self-Attention Generative Adversarial Networks [Journal]. - 2018.

9. Ian J. Goodfellow Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio Generative Adversarial Networks [Journal]. - 2014.

10. jupyter [Online] // jupyter. - https://jupyter.org/.

11. LSUN: Construction of a Large-scale Image Dataset [Online] // LSUN. - https://www.yf.io/p/lsun.

12. Martin Arjovsky Leon Bottou TOWARDS PRINCIPLED METHODS FOR TRAINING GENERATIVE ADVERSARIAL NETWORKS [Journal]. - 2017.

13. Martin Arjovsky Soumith Chintala, L´eon Bottou Wasserstein GAN. - 2017.

14. Martin Heusel Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium [Journal]. - 2018.

16. Miyato T. and Koyama, M. cGANs with projection discriminator [Journal]. - 2018.

17. Nielsen Michael [Online] // Neural Networks and Deep Learning. - http://neuralnetworksanddeeplearning.com/.

18. NVIDIA TESLA K80 [Online] // NVIDIA. - https://www.nvidia.com/ru-ru/data-center/tesla-k80/.

19. NVIDIA V100 TENSOR CORE GPU [Online] // NVIDIA. - https://www.nvidia.com/en-us/data-center/v100/.

20. O. Sbai M. Elhoseiny, A. Border, Y. Lecun Desing Inspiration from Generative Networks.

21. Progressive Growing of GANs for Improved Quality, Stability, and Variation – Official TensorFlow implementation of the ICLR 2018 paper [Online] // https://github.com/. - https://github.com/tkarras/progressive_growing_of_gans.

22. Sebastian Nowozin Botond Cseke, Ryota Tomioka f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization [Journal]. - 2016.

23. Takeru Miyato Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida SPECTRAL NORMALIZATION FOR GENERATIVE ADVERSARIAL NETWORKS [Journal]. - 2018.

24. TensorBoard: TensorFlow's visualization toolkit [Online] // tensorflow. - https://www.tensorflow.org/tensorboard.

25. Tero Karras Timo Aila, Samuli Laine, Jaakko Lehtinen Progressive Growing of GANs for Improved Quality, Stability, and Variation [Journal]. - 2018.

26. Wenzhe Shi Wenzhe Shi, Ferenc Huszar, Johannes Totz, Andrew P. Aitken Real-Time Single Image and Video Super-Resolution Using an Efficient [Journal]. - 2016.

27. wikipedia One-hot [Online] // Wikipedia. - https://en.wikipedia.org/wiki/One-hot.

28. Wikipedia Rectifier (neural networks) [Online] // Wikipedia. - https://en.wikipedia.org/wiki/Rectifier_(neural_networks).

29. Yongjun Hong Uiwon Hwang, Jaeyoon Yoo, Sungroh Yoon How Generative Adversarial Networks and Their Variants Work: An Overview [Online] // https://arxiv.org/.

30. Головко В.А Краснопрошин В.В. Нейросетевые технологии обработки данных [Book]. - Минск : БГУ, 2017.

31. Иошуа Бенджио Ян Гудфеллоу Глубокое обучение [Book]. - [s.l.] : ДМК, 2017.

32. Круглов В.В Борисов В.В Искусственные нейронные сети. Теория и практика [Book]. - [s.l.] : Горяча линия-Телеком, 2002.

33. Нейронные сети перцептрон Нейронные сети, перцептрон [Online] // ИТМО Викиконспекты. -

http://neerc.ifmo.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8,_%D0%BF%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD.

34. Николенко С.И Кадурин А.А., Архангельская Е.О Глубокое обучние [Book]. - Санкт-Петербург : Питер, 2018.

35. Хайкин Саймон Нейронные сети: полный курс [Book]. - [s.l.] : Вильямс, 2016.