МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра биомедицинской информатики

ИССЛЕДОВАНИЕ И РАЗРАБОТКА МЕТОДОВ ГЕНЕРАЦИИ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНЫХ НЕЙРОННЫХ СЕТЕЙ

Курсовая работа

Зеленковского Виктора Петровича
студента 3 курса,
специальность «информатика»

Научный руководитель:
доцент кафедры БМИ
Ковалёв В.А.

Минск, 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В последние несколько лет особую популярность в мире машинного обучения обрели генеративно-состязательные сети (GANs). С каждым годом они становятся более мощными, создавая данные, имеющие высокую степень схожести с исходными, на которых они были обучены.

Сегодня генеративно-состязательные сети используются во многих областях, в том числе и в медицине. В данной сфере наибольшую популярность сети обрели благодаря своей возможности создания медицинских изображений. Их количество, как правило, ограничено, из-за чего не представляется возможным обучить методы машинного обучения, которые в последствии используются для ранней диагностики заболеваний, до необходимого уровня. GANs могут использоваться в качестве инструмента увеличения таких данных путем преобразования изображения в изображение, генерации новых данных и увеличения разрешения исходных медицинских изображений.

Но в медицине качество сгенерированных данных особо важно, так как на этих данных в дальнейшем будут обучаться модели машинного обучения. А модели, плохо обученные из-за нерепрезентативных и плохих исходных данных, могут привести к серьёзным последствиям. Поэтому качество генерируемых медицинских изображений должно быть высоко.

Пытаясь решить проблемы, описанные выше, данная работа направлена на изучение методов генерации изображений, созданию новых медицинских изображений и оценке их качества.

ГЛАВА 1

ГЛУБОКИЕ СВЁРТОЧНЫЕ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНЫЕ СЕТИ

1.1 Общие идеи генеративно-состязательных сетей

Генеративно-состязательные сети (GANs) существуют уже несколько лет. Они были представлены в статье 2014 года [1] Яном Гудфеллоу и его коллегами из Монреальского университета и с тех пор являются популярной областью исследований. Проанализировав данную статью и обобщив её результаты, можно сказать, что GANs является типом генеративной модели, которая пытается синтезировать новые данные, неотличимые от обучающих. Представленная форма обучения является обучением без учителя.

Архитектура генеративно-состязательной сети представляет собой две нейронные сети, конкурирующие между собой: **генератор**, которому на начальном этапе подается вектор случайных чисел и выводятся сгенерированные данные, и **дискриминатор**, которому подается фрагмент данных и выводится значение, которое отражает уверенность дискриминатора в том, что входное изображение получено из обучающего набора данных. Другими словами, генератор создает изображения, а дискриминатор пытается отличить их от настоящих изображений.

Обе сети сначала довольно плохо справляются, но когда обучение проходит успешно, они совместно совершенствуются до тех пор, пока генератор не начнет выдавать убедительные изображения. Процесс обучения происходит таким образом, что улучшение одной из сетей соответствует ухудшению другой. Из-за этого значение функции потерь в любой момент времени не говорит нам о том, насколько хорошо обучена система в целом, а только о том, насколько хорошо работает генератор или дискриминатор относительно их предыдущих версий.

Однако использовать GAN на практике часто было довольно сложно. С самого начала исследователи заметили, что обучения данных сетей приводит к **коллапсу модели**. Коллапс модели может произойти, когда дискриминатор обучился настолько, что обучающие градиенты для генератора становятся все менее и менее полезными. Это может произойти относительно быстро во время обучения, в последствии чего генератор начинает создавать каждый раз одни и те же данные.

С тех пор исследовательское сообщество продолжает придумывать множество способов сделать обучение более надежным, а сгенерированные данные более похожими на исходные.

1.2 Свёрточные сети в генеративно-состязательных сетях

Значительное улучшение в генерации изображений произошло в 2016 году, когда Рэдфорд и др. опубликовали статью[2]. До этого все попытки улучшить GANs, используя свёрточные нейронные сети(CNNs) для моделирования изображений, не привели к существенным результатам. Это и побудило авторов глубоких свёрточных генеративно-состязательных сетей (DCGANs) разработать альтернативный подход к генерации изображений.

1.2.1 Архитектура

Исходная архитектура GAN использовала только многослойный перцептрон (MLP), но CNN лучше обрабатывают изображения, чем MLP. Именно поэтому основой подхода, представленного авторами, является использование и изменения архитектуры CNN, представленные ниже:

- для генератора, показанного на рисунке ниже, свертки заменены транспонированными свертками, поэтому представление данных на каждом уровне генератора последовательно увеличивается, поскольку оно отображается из низкоразмерного скрытого вектора в изображение высокой размерности. Также сделана замена любых объединяющих слоев на пошаговые свертки (дискриминатор) и свертки с дробным шагом (генератор);

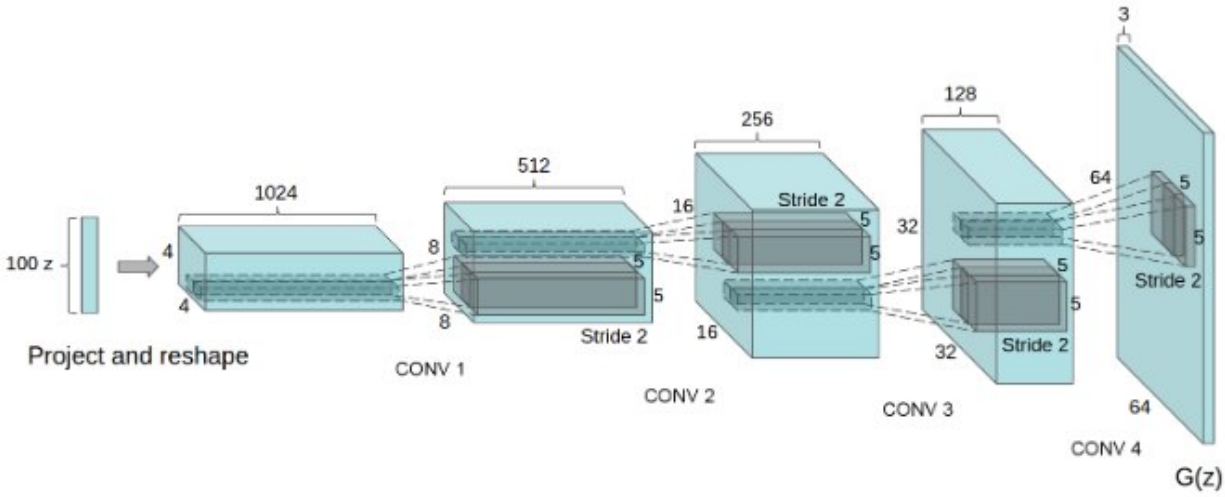

Рисунок 1.1 — Архитектура DCGAN

- использование пакетной нормализации как в генераторе, так и в дискриминаторе;
- использование ReLU активации в генераторе для всех слоев, за исключением выходного, который использует Tanh. Использование LeakyReLU активации в дискриминаторе для всех слоев;
- использование Adam оптимизатора вместо SGD с momentum.

1.2.2 Преимущества и недостатки

Вышеперечисленные модификации позволили DCGAN добиться стабильной тренировки.

Но существенным недостатком DCGANs является тот факт, что данная архитектрура состязательных сетей может создавать изображения только определенного размера. Чем выше разрешение изображения, тем легче дискриминатору отличить настоящие изображения от сгенерированных, что делает *коллапс модели* (поведение сети, характерезующееся генерацией только небольшого подмножества выборок) более вероятным.

Генерация изображений 32x32 или даже 128x128 стала возможным с помощью DCGANs, но создание изображений с разрешением выше 512x512 осталось сложной задачей.

ГЛАВА 2

ПРОГРЕССИВНЫЙ РОСТ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНЫХ СЕТЕЙ

2.1 Архитектура

В 2018 году Каррас и др. [3] представили методику обучения для создания изображений с высоким разрешением.

2.1.1 Увеличение количества слоев

Идея метода заключается в том, чтобы иметь возможность синтезировать изображения высокого разрешения и высокого качества с помощью постепенного наращивание сетей дискриминатора и генератора в процессе обучения. Данный подход облегчает создание изображений с высоким разрешением, обучая сеть начиная с изображений с более низким разрешением, постепенно переходя к изображениям с более высоким разрешением.

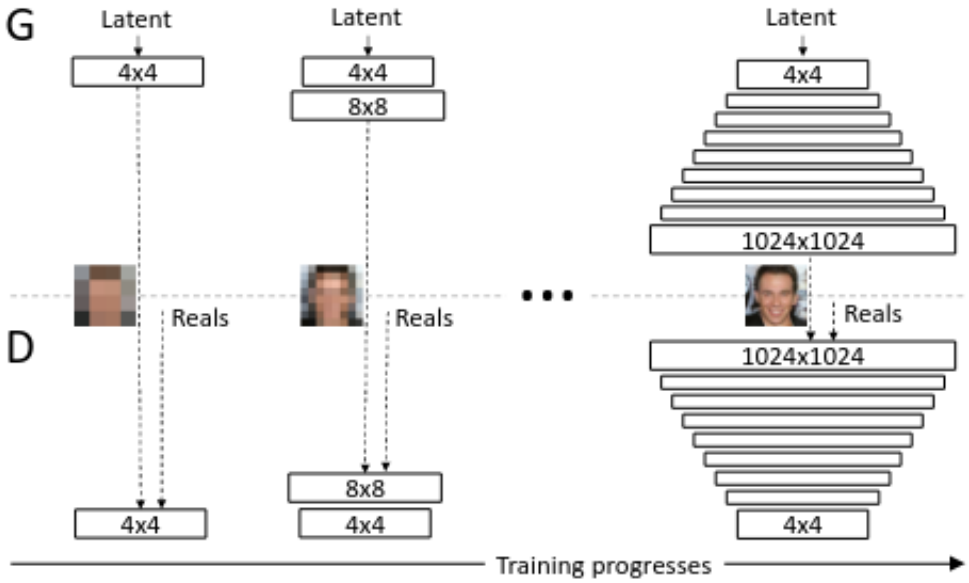

Рисунок 2.1 — Идея обучения

Постепенное увеличение количества слоев можно обобщить в следующие этапы обучения:

- искусственно сократить обучающие изображения до очень маленького начального разрешения (4x4 пикселя);
- создать генератор всего с несколькими слоями для синтеза изображений с таким низким разрешением и соответствующий дискриминатор зеркальной

архитектуры (поскольку данные сети являются небольшими, скорость их обучения относительно высокая и изучают они только крупномасштабные структуры, видимые на сильно размытых изображениях);

- после обчения добавить еще один слой к генератору и дискриминатору, удваивая выходное разрешение до размера 8x8 (обученные веса в ранних слоях сохраняются и не блокируются, а новый слой обеспечивает их постепенное затухание, чтобы помочь стабилизировать переход);
- продолжать добавлять слои, удваивая разрешение и тренируясь до тех пор, пока не будет достигнут желаемый размер и результат выходного изображения.

2.1.2 Изменение представления изображения

В дополнение к постепенному расширению сетей авторы статьи внесли несколько других архитектурных изменений, чтобы облегчить стабильное и эффективное обучение:

- каждый набор слоев генератора удваивает размер представления и вдвое уменьшает количество каналов, пока выходной слой не создаст изображение всего с тремя каналами, соответствующие RGB;
- дискриминатор уменьшает вдвое размер представления и удваивает количество каналов с каждым набором слоев;
- в обеих сетях схема удвоения канала прерывается ограничением количества фильтров разумным значением, например 512, чтобы общее количество параметров не стало слишком большим.

Таким образом архитектура прогрессивных нейронных сетей выглядит следующим образом:

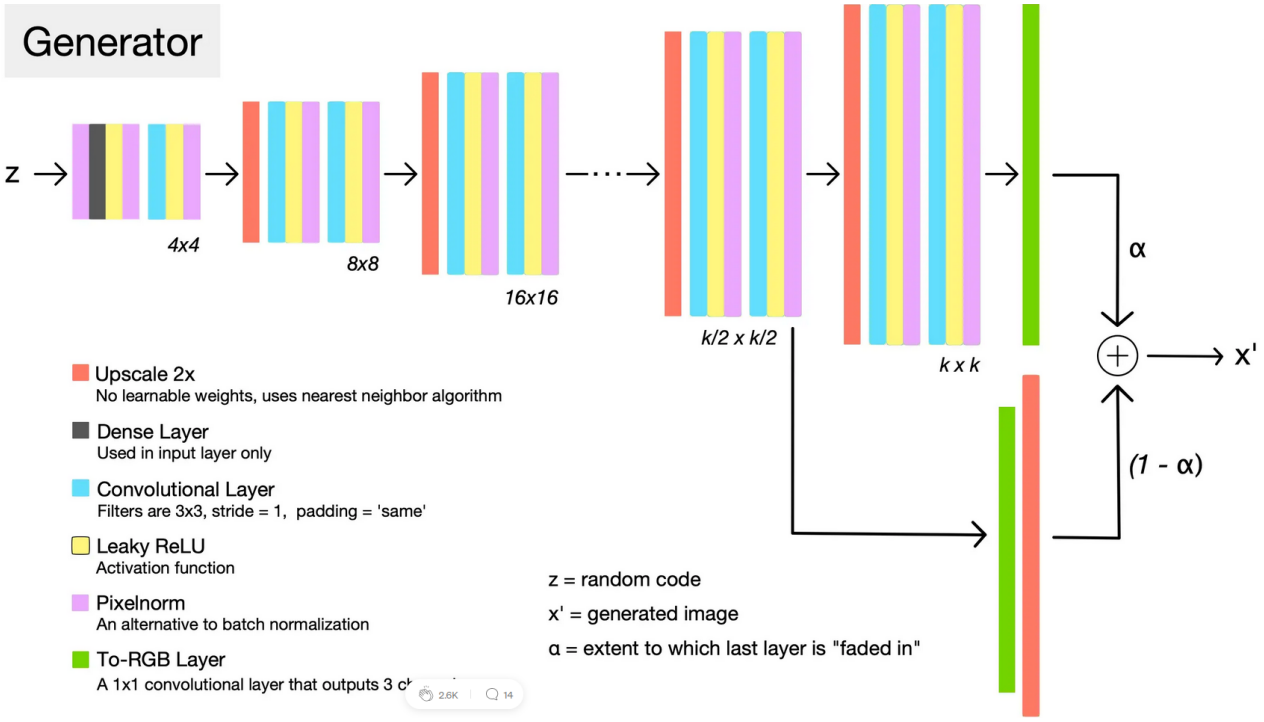

Рисунок 2.2 — Архитектура генератора

2.2 Нормализация пикселей

Вместо использования пакетной нормализации, как это обычно делается, авторы статьи используют пиксельную нормализацию. Слой, отвечающий за нормализацию, не имеет обучаемых весов. Он нормализует вектор объектов в каждом пикселе до единичной длины и применяется после свёрточных слоев в генераторе. Это сделано для того, чтобы величины сигнала не выходили из-под контроля во время тренировки.

Нормализация пикселей выполняется по следующей формуле (значения каждого пикселя (x, y) по каналам C нормализуются до фиксированной длины):

$$b_{x,y} = \frac{a_{x,y}}{\sqrt{\frac{1}{C}\sum_{i=0}^{C} a_{x,y}^{i} + \epsilon}}, \tag{2.1}$$

a - входной тензор, b - выходной тензор, ε - небольшое значение, предотвращающее деление на ноль.

2.3 Стандартное отклонение мини-выборки

Как правило, генеративно-состязательные сети создают выборки с меньшим разбросом, чем в обучающем наборе. Один из подходов к борьбе с этим заключается в том, чтобы дискриминатор вычислял статистику мини-выборке и использовал эту информацию, чтобы помочь отличить реальные данные от сгенерированных. Это побуждает генератор создавать больше разнообразия, так что статистические данные, вычисленные по сгенерированной мини-выборке, более близки к статистическим данным из обучающей выборки.

В ProGAN это делается путем вставки слоя «стандартное отклонение мини-выборки» ближе к концу дискриминатора. Этот слой не имеет обучаемых параметров. Он вычисляет стандартные отклонения пикселей объектов по всей мини-выборке и добавляет их в качестве дополнительного канала.

2.4 «Затухание» в новых слоях и равномерный темп обучения

В процессе обучения сети постепенно расширяются за счет добавления нового набора слоев, чтобы удваивать разрешение каждый раз, когда обучение завершается с существующим разрешением. Когда добавляются новые слои, параметры в предыдущих слоях остаются обучаемыми.

Чтобы предотвратить сильные изменения в ранее существовавших нижних слоях от внезапного добавления нового верхнего слоя, предыдущие слои линейно «затухают». Это затухание контролируется параметром α, который линейно интерполируется от 0 до 1 в течение многих итераций обучения. Таким образом, как видно на *Рисунке 1.3* окончательно сгенерированное изображение

представляет собой взвешенную сумму последнего и предпоследнего слоев в генераторе.

Также авторы обнаружили, что для обеспечения правильной конкуренции между генератором и дискриминатором важно, чтобы слои обучались с одинаковой скоростью. Чтобы достичь такой равномерной скорости обучения, они масштабируют веса слоя в соответствии с тем, сколько парметров имеет данный слой. Делаются это при каждом прямом проходе во время обучения, а не только при инициализации. Перед выполнением свертки с f фильтрами размера [k, k, c] масштабирование весов этих фильтров происходит по следующей формуле:

$$W_f = W_i * \sqrt{\frac{2}{k * k * c}}. \tag{2.2}$$

Благодаря этому вмешательству для инициализации веса не требуется никаких сложных трюков — просто инициализация весов с помощью стандартного нормального распределения работает нормально.

2.5 Функция потерь

Ни одно из вышеуказанных улучшений не зависит от конкретной функции потерь, поэтому предоставляется возможным использовать любую из популярных функций потерь, которые появились за последние несколько лет.

Однако в статье авторы используют улучшенную функцию потерь Вассерштейна, также известную как WGAN-GP. Это одна из наиболее популярных функций потерь. Было показано, что она стабилизирует процесс обучения и повышает шансы на сходимости[3].

$$Loss_G = -D(x_{gen}), \tag{2.3}$$

$$GP = (||\nabla_D(a * x_{gen} + (1 - a) * x_{real})||_2 - 1)^2, \tag{2.4}$$

$$Loss_G = -D(x_{real}) + D(x_{gen}) + \lambda * GP, \tag{2.5}$$

где GP - штраф градиента, помогающий стабилизировать обучение;

a - коэффицент от 0 до 1, выбираемый случайным образом;

λ - настраиваемый коэффицент(обычно выбирается равным 10).

Важно отметить, что функция потерь WGAN-GP ожидает, что $D(x_{real})$ и $D(x_{gen})$ будут неограниченными вещественными числами. Другими словами, не ожидается, что выходные данные дискриминатора будут иметь значение от 0 до 1. Это немного отличается от традиционной формулировки GAN, которая рассматривает выходные данные дискриминатора как вероятность.

2.6 Эффективность

Постепенно увеличивая разрешение, мы постоянно просим сеть изучить гораздо более простую часть исходной задачи. Постепенный процесс обучения значительно стабилизирует обучение. Это способоствует снижению вероятности *коллапса модели*.

Способ перехода от низкого разрешения к высокому вынуждает постепенно растущие сети сначала сосредоточиться на структуре высокого уровня (фрагменты, различимые в наиболее размытых версиях изображения), а затем заполнять детали. Это улучшает качество конечного изображения за счет снижения вероятности того, что сеть получит какую-то высокоуровневую структуру совершенно неправильно.

Постепенное увеличение размера сети также является более эффективным с точки зрения вычислений, чем более традиционный подход к инициализации всех уровней одновременно. Меньшее количество слоев быстрее поддается обучению, так как в них просто меньше параметров. Поскольку все итерации обучения, кроме последнего набора, выполняются с подмножеством конечных слоев, это приводит к впечатляющему повышению эффективности. Авторы статьи обнаружили, что ProGAN обычно обучается примерно в 2-6 раз быстрее, чем соответствующий исходный GAN.

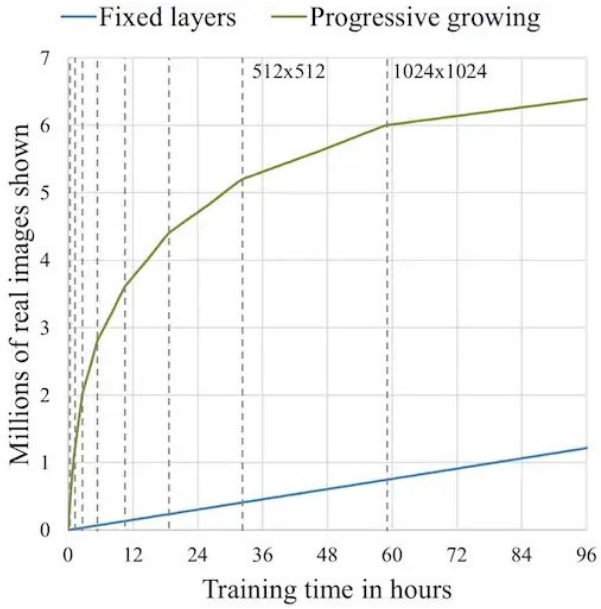

Рисунок 2.3 — Эффективность ProGANs

ГЛАВА 3

АРХИТЕКТУРА ГЕНЕРАТОРА, ОСНОВАННАЯ НА СТИЛЕ (STYLEGANS)

3.1 Архитектура StyleGAN

3.1.1 Вектор стиля и нормализация

В 2019 году авторы статьи, которой была посвящена Глава 2, перепроектировали архитектуру генератора таким образом, чтобы обеспечить новые способы управления процессом создания изображений на основе переноса стиля.

До этого скрытый вектор поступал к генератору через входной уровень, который представлял собой полносвязную нейросеть (см. Рисунок 3.1, слева). Авторы статьи [6] предложили не использовать входной слой, а вместо этого начинать с обученной константы (см. Рисунок 3.2, справа). Таким образом из скрытого вектора z во входном скрытом пространстве Z нелинейная сеть, используя отображение $f : Z \rightarrow W$, выдает $w \in W$.

Для простоты размерность обоих пространств (Z и W) берется равной 512. Отображение f реализовано с помощью 8-слойного многослойного перцептрона (MLP).

Затем обучаемые аффинные преобразования A преобразуют полученный вектор w в вектор стиля $y = (ys, yb)$, который отвечает за адаптивную нормализацией(AdaIN) после каждого свёрточного слоя сети генератора, которая определяется следующим образом:

$$\text{AdaIN}(x_i, y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \tag{3.1}$$

Смысл данной формулы заключается в нормализации x_i, а затем масштабировании и смещении с использованием соответствующих скалярных компонент вектора стиля y.

3.1.2 Добавление шума

Также генератор имеет инструмент для генерации стохастической детализации - введение явных шумовых входных данных, которые представляют собой одноканальные изображения, состоящие из некоррелированного гауссовского шума.

Данный шум передается на каждый уровень генератора, предварительно пройдя через обучаемые коэффициенты масштабирования B, а затем добавляется к выходным данным соответствующей свертки, как показано на Рисунке 3.1.

В итоге архитектура сети выглядит следующим образом:

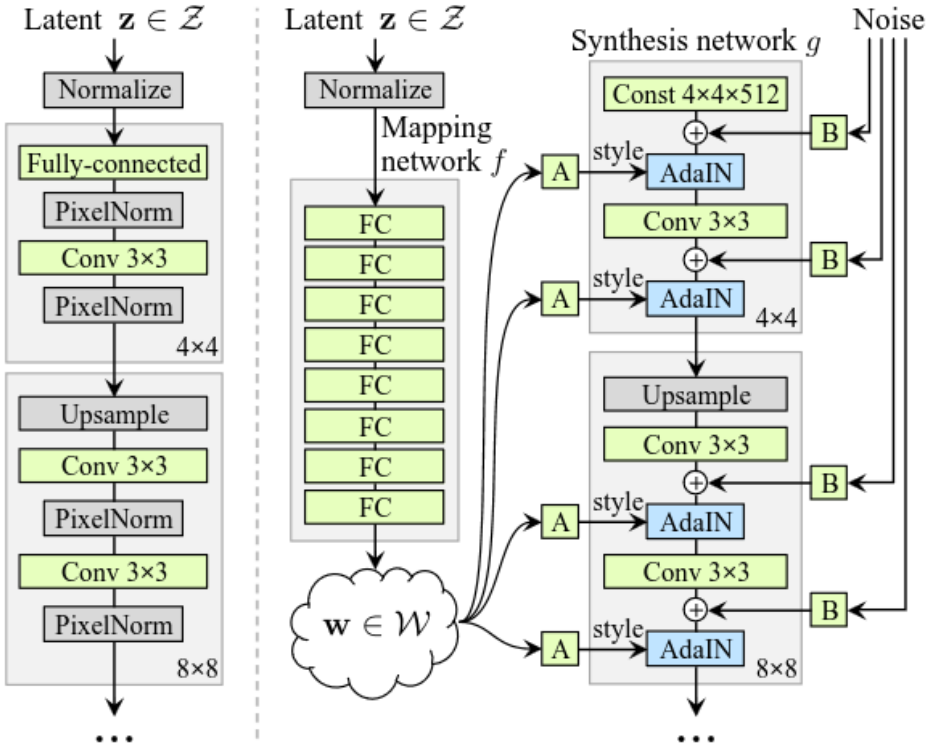

Рисунок 3.1 — Архитектура StyleGAN

3.2 Смешивание стилей как способ локализации

3.2.1 Локализация

Полученная архитектура генератора позволяет управлять созданием изображений с помощью модификаций вектора стилей в зависимости от масштаба. Сеть f, преобразующая вектор z в вектор w, и аффинные преобразования рассматривается как способ получения изображений для каждого стиля из исходного распределения, а сеть генерации - как способ создания нового изображения на основе нескольких стилей. Влияние каждого стиля локализованы в сети, т.е. изменение определенного подмножества стилей повлияет только на определенные части изображения.

Такая локализации обосновывается адаптивной нормализацией(см. Уравнение 3.1), которая сначала нормализует каждый канал до нулевого среднего значения и единичной дисперсии, а только затем применяет масштаб и смещения, основанные на векторе стиля y. Новая статистика для каждого канала, в соответствии с определением вектора y, изменяет относительную важность объектов для последующей операции свертки, но они не зависят от исходной статистики из-за нормализации. Таким образом, каждый стиль управляет только одной сверткой, прежде чем быть переопределенным следующей операцией AdaIN.

3.2.2 Смешивание стилей

Для дальнейшего увеличения локализации используется регуляризация смешивания, при которой определенный процент изображений генерируется с использованием двух случайных скрытых векторов вместо одного во время обучения. При генерации такого изображения происходит переключение с одного скрытого вектора на другой в случайно выбранной точке сети генерации. Такая операция имеет название смешивание стилей. Во время выполнения данной операции два скрытых вектора $z1, z2$ проходят через сеть f и получаем соответствующие векторы стилей $w1, w2$, которые управляют стилями таким образом, чтобы $w1$ применялся до точки пересечения, а $w2$ - после нее. Такой подход к регуляризации не позволяет сети предполагать, что соседние стили коррелированы.

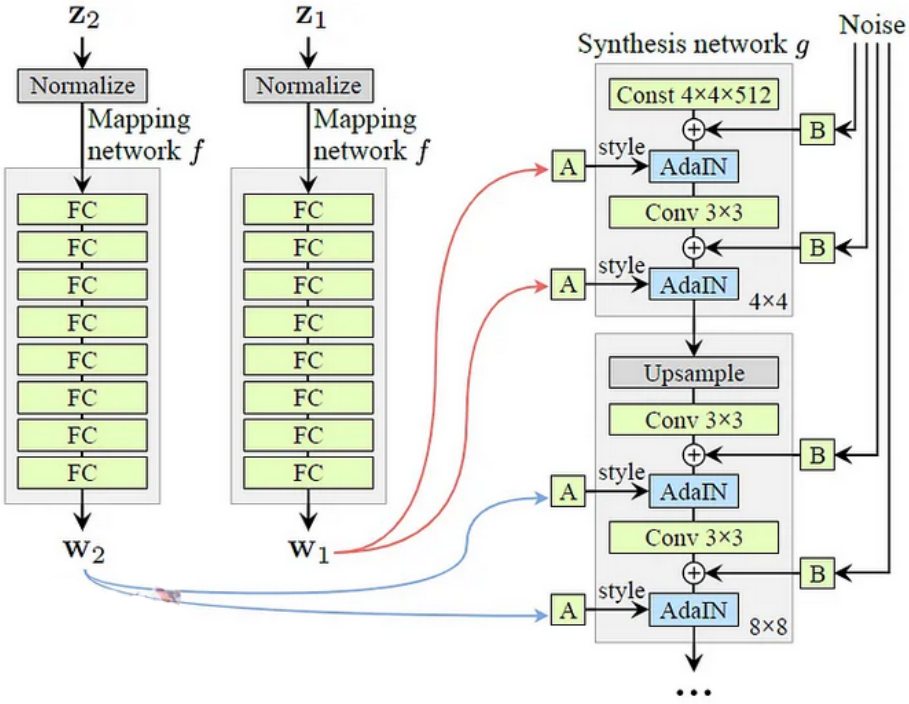

Рисунок 3.2 — Смешивание стилей

3.3 Проблемы при создании изображений

3.3.1 Проблемы и причины

В процессе создания изображений с помощью StyleGAN на многих изображениях заметны одинаковые размытые части, которые не похожи на все остальное изображение. Данная проблема возникает из-за введенной в Разделе 3.1 адаптивной нормализации.

AdaIN нормализует среднее значение и дисперсию каждой группы признаков по отдельности, тем самым потенциально уничтожая любую информацию, найденную в признаках объектов относительно друг друга. В силу этого можно говорить, что генератор намеренно скрывает информацию о силе сигнала

после нормализации объекта: создавая сильный локализованный всплеск, который доминирует в статистике, генератор может эффективно масштабировать сигнал так, как ему нравится в других местах. Подтверждается данный факт на практике: при удалении шага нормализации из генератора, как подробно описано ниже, проблема полностью исчезает.

3.3.2 Детальная архитектура StyleGAN

Для решения возникшей проблемы для начала рассмотрим архитектуру StyleGAN более подробно. На Рисунке 3.3, слева показана оригинальная сеть StyleGAN, а справа та же сеть, но более детально: показаны веса, смещения операцию AdaIN разбита на две составные части: нормализацию и модуляцию. Такое представление позволяет перерисовать блоки сети таким образом, чтобы каждый блок указывал на ту часть сети, где активен один стиль (т.е. «блок стилей»). Оригинальный StyleGAN применяет смещение и шум внутри блока стилей, в результате чего их относительное влияние обратно пропорционально значениям текущего стиля.

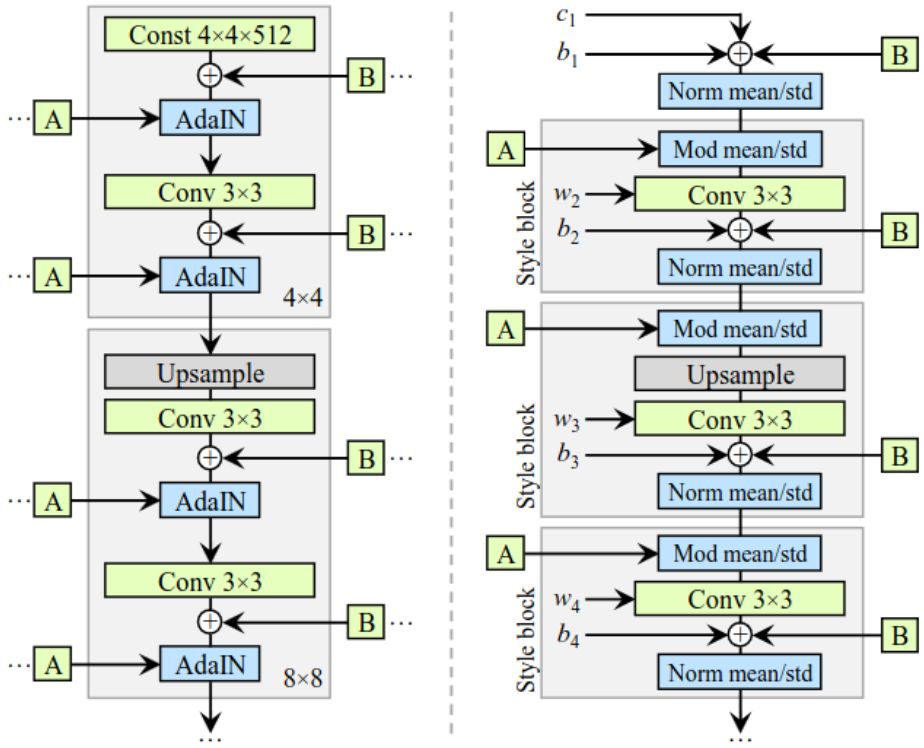

Рисунок 3.3 — StyleGAN подробно

После выполнения такого преобразования эмпирически можно заметить следующее:

- при перемещении этих операций за пределы блока стилей, где они оперируют нормализованными данными, получаются более предсказуемые результаты;
- нормализации и модуляции достаточно оперировать только стандартным отклонением (т.е. среднее значение не требуется);

- применение смещения, шума и нормализации к постоянному входному сигналу также может быть безопасно устранено без заметных недостатков.

Сделав такие выводы, архитектуру нейросети можно изменить, как показано на Рисунке 3.4 слева.

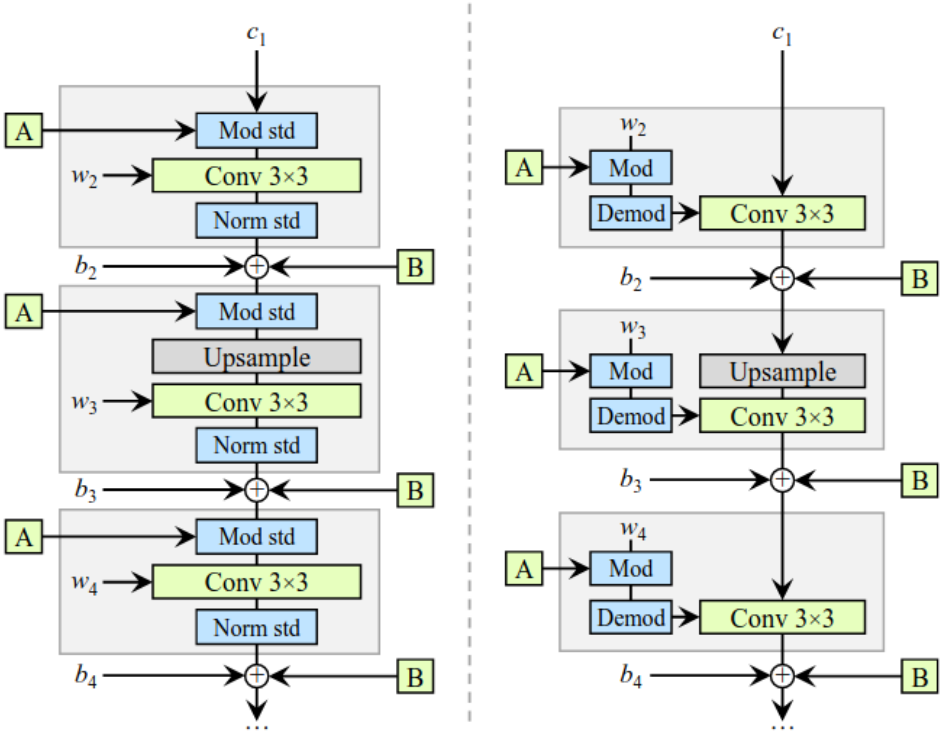

Рисунок 3.4 — Изменение архитектуры StyleGAN

Заметим, что проделанные изменения оказывают либо нейтральное, либо небольшое положительное влияние с точки зрения показателей качества.

3.3.3 Способ решения (StyleGAN2)

Одним из способов решения проблемы является удаление нормализации, но при это теряется возможность управления генерируемыми изображениями с помощью вектора стиля, что является одной из главных сильных сторон StyleGAN.

Существует другой способ, позволяющий решить возникшую проблему с размытыми частями изображения и при этом сохранить управление стилем. Модуляция стиля может на порядок усилить определенные группы признаков, но чтобы в таком случае работало смешивание стилей, необходимо явно противодействовать данному усилению - в противном случае последующие слои не смогут осмысленно оперировать данными.

Идея состоит в том, чтобы основывать нормализацию на ожидаемой статистике входящих картах признаков (с англ. feature map), но делать это неявно, а не как это делалось в AdaIN. Блок стилей (см. Рисунок 3.4, слева) состоит из модуляции, свертки и нормализации. Модуляция масштабирует каждую входную карту признаков свертки на основе входящего стиля, который в качестве альтернативы может быть реализован путем масштабирования весов свертки:

$$w'_{i,j,k} = s_i * w_{i,j,k} \tag{3.2}$$

где w, w' - исходный и модулированный веса соответственно, s_i - масштаб, соответствующий $i-$ входной карте признаков, а j и k перечисляют выходные карты признаков и пространственный след свертки соответственно.

Далее цель нормализации состоит в том, чтобы удалить эффект, оказываемый s, из статистики выходных карт признаков свертки. Предположив, что входные активации являются случайными величинами с единичным стандартным отклонением, после модуляции и свертки выходные активации имеют стандартное отклонение, равное

$$\sigma_j = \sqrt{\sum_{i,k} w'^2_{i,j,k}} \tag{3.3}$$

Теперь выходные данные масштабируются по норме L2 соответствующих весов. Далее необходимо вернуть выходные данные обратно к стандартному отклонению, равному 1. Исходя из Уравнения 3.3, это достигается, если отмасштабировать («демодулировать») каждую выходную карту признаков j на $\frac{1}{\sigma_j}$. Данный этап также может быть включен в веса свертки, которые после этого примут вид:

$$w''_{i,j,k} = \frac{w'_{i,j,k}}{\sqrt{\sum_{i,k} w'^2_{i,j,k} + \epsilon}} \tag{3.4}$$

где ϵ - небольшая константа, позволяющая избежать деления на нуль.

Таким образом блок стилей преобразован в один слой свертки, веса которого скорректированы на основе s с использованием Уравнений 3.1 и 3.3 (Рисунок 3.4, справа).

По сравнению с нормализацией, полученная демодуляции слабее, поскольку основана на статистических предположениях о сигнале, а не на фактическом содержимом карт признаков.

3.3.4 Изменение прогрессивного роста модели

Прогрессивный рост, который был рассмотрен в Главе 2, способствует стабилизации создания изображений с высоким разрешением, но при этом возникает проблема, о которой говорилось в подразделе 3.3.2. Возникает это из-за того, что постепенно расширяющийся генератор отдает предпочтение расположению деталей изображения, потому что при постепенном увеличении каждое разрешение на мгновение служит выходным разрешением, что способствует генерации максимально частотных характеристик. Это приводит к тому, что обученная сеть имеет чрезмерно высокие частоты в промежуточных слоях, включая инвариантность к сдвигу.

В то время как StyleGAN использует простые схемы прямой связи в генераторе и дискриминаторе, существует огромный объем работ, посвященных изучению более совершенных сетевых архитектур. Пропускные соединения, остаточные сети и иерархические методы оказались весьма успешными также в контексте генеративных методов.

На Рисунке 3.5, слева показана упрощенная конструкция прогрессивного роста, полученная путем увеличения(Up) размерности и суммирования вкладов выходных сигналов RGB, соответствующих различным разрешениям. В каждый блок дискриминатора аналогично передается изображение с уменьшенной размерностью(Down). Во всех операциях увеличения и уменьшения размерности используется билинейная фильтрация. На Рисунке 3.5, справа дополнительно введена модификация, использующая остаточные соединения.

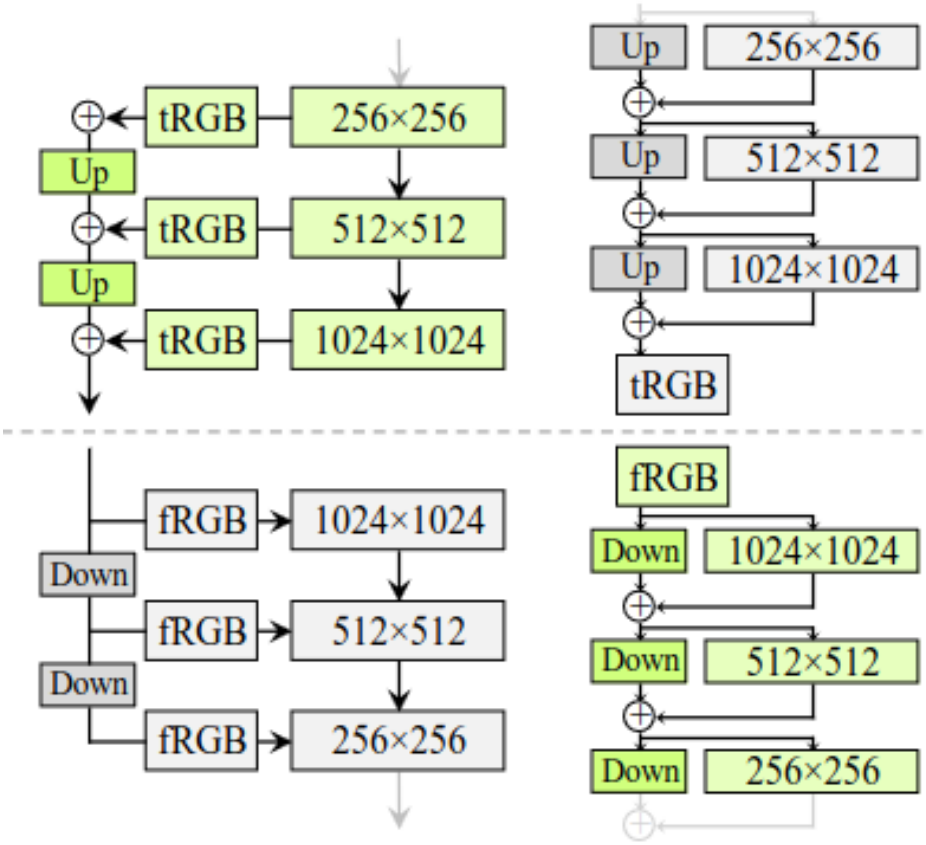

Рисунок 3.5 — Изменение прогрессивного роста

ГЛАВА 4

ОБУЧЕНИЕ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНЫХ СЕТЕЙ НА ОГРАНИЧЕННОМ НАБОРЕ ДАННЫХ

4.1 Проблемы небольшого набора данных для GANs

Архитектуры генеративно-состязательных сетей, рассмотренные в предыдущих главах, действительно показывают хорошие результаты при генерации изображений[1, 2, 3, 6, 7], но их использование ограничено при небольшом размере исходных данных. Проблема заключается в том, что дискриминатор переобучается на небольшом количестве обучающих данных, что в том числе приводит к коллапсу модели (см. Раздел 4.2). Также его связь с генератором становится бессмысленной, и обучение начинает расходиться[8].

Почти во всех областях глубокого обучения увеличение набора данных(с англ. augmentation, также далее будет использоваться термин преобразование) является стандартным решением переобучения. Например, обучение классификатора при использовании вращений изображения, добавлении шума и т.д. приводит к повышению инвариантности к этим искажениям, что хорошо влияет на обучение классификатора.

В генеративно-состязательных сетях, наоборот, обучение с использованием аналогичных преобразований приводит к генерации расширенного распределения. Такая «утечка» преобразований в сгенерированные образцы крайне нежелательна, так как она приводит к зашумленным результатам.

4.2 Регуляризация сбалансированной согласованности

По определению, любое дополнение, которое применяется к обучающему набору данных, будет унаследовано сгенерированными изображениями [1]. Чжао и др. [9] предложили регуляризацию сбалансированной согласованности (BCR) в качестве решения, которое, как предполагается, не приведет к утечке преобразований к сгенерированным изображениям. Регуляризация согласованности утверждает, что два набора преобразований, примененных к одному и тому же входному изображению, должны давать одинаковый результат. Чжао и др. добавили условия регуляризации согласованности для потери дискриминатора и обеспечили согласованность дискриминатора как для реальных, так и для сгенерированных изображений. В то время при обучении генератора не применяются никакие изменения или условия потери согласованности (Русунок 4.1, слева). Таким образом, их подход эффективно направлен на обобщение дискриминатора, делая для него набор изменений невидимыми. Однако при

этом утечка изменений становится возможным для генератора, который может свободно создавать изображения, содержащие их.

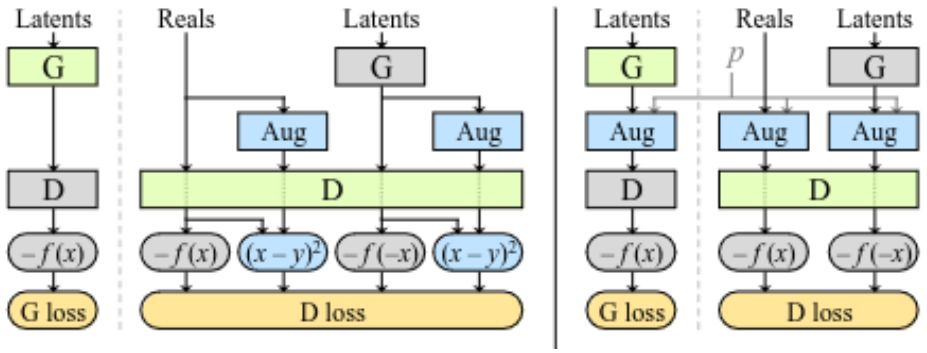

Рисунок 4.1 — Применение изменений

4.3 Стохастические расширения дискриминатора

Авторы статьи [10] предложили аналогично bCR применять набор преобразований ко всем изображениям, отображаемым дискриминатору. Однако вместо добавления отдельных условий потери CR оценивать дискриминатор только с использованием дополненных изображений и делать это также при обучении генератора (Русиунок 4.1, справа). Данный подход они назвали *увеличением стохастического дискриминатора*. Справедливость данного подхода обсуждается ниже, ведь не очевидно, работает ли данный способ вообще: если дискриминатор никогда не видит, как на самом деле выглядят обучающие изображения, неясно, сможет ли он правильно управлять генератором.

Бора и др.[11] рассмотрели аналогичную проблему при обучении GAN в условиях искаженных измерений и показали, что обучение неявно устраняет искажения и находит правильное распределение, при условии, что процесс искажения представлен обратимым преобразованием распределений вероятностей по пространству данных. Такие операторы увеличения набора данных они назвали непротекающими.

Сила таких обратимых преобразований заключается в том, что они позволяют делать выводы о равенстве или неравенстве базовых множеств, наблюдая только за дополненными множествами. Важно понимать, что это не означает, что изменения, выполненные на отдельных изображениях, должны быть необратимыми. Например, такие экстремальные увеличения набора данных, как установка входного изображения равным нулю в 90 % случаев, обратимо в смысле распределения вероятностей: даже человеку было бы легко рассуждать об исходном распределении, игнорируя черные изображения до тех пор, пока не останется только 10 изображений. С другой стороны, случайные вращения, равномерно выбранные из 0°, 90°, 180°, 270° не являются обратимыми: невозможно различить различия между ориентациями после увеличения набора данных.

Ситуация меняется, если это вращение выполняется только с вероятностью $p < 1$: это увеличивает относительную вероятность значения $0°$, и теперь расширенные распределения могут совпадать только в том случае, если сгенерированные изображения имеют правильную ориентацию. Аналогично, следующие стохастические изменений набора данных могут быть сконструированы таким образом, чтобы не допускать утечки при условии, что они пропускаются с ненулевой вероятностью:

- детерминированные отображения (например, базисные преобразования);
- аддитивный шум;
- группы преобразований (например, повороты изображения или цветового пространства, перевороты и масштабирование);
- проекции.

Сила изменений контролируется параметром $p \in [0, 1]$, так что каждое преобразование применяется с вероятностью p или пропускается с вероятностью $(1-p)$. Одно и то же значение p для всех преобразований. Случайный выбор выполняется отдельно для каждого изменения и для каждого изображения в мини-выборке.

4.4 Адаптивные расширения дискриминатора

Обозначим выходные данные дискриминатора через D_{train}, $D_{validation}$ и $D_{generated}$ для обучающего набора, набора проверки и сгенерированных изображений соответственно, а их среднее значение по N последовательным мини-выборкам через $E[\cdot]$. На практике $N = 4$, что соответствует 4 × 64 = 256 изображениям. Теперь введем две правдоподобные эвристики переобучения:

$$r_v = \frac{E[D_{tarin}] - E[D_{validation}]}{E[D_{tarin}] - E[D_{genearated}]} \tag{4.1}$$

$$r_t = E[sign(D_{train})] \tag{4.2}$$

Для обеих эвристик $r = 0$ означает отсутствие переобучения, а $r = 1$ указывает на полное переобучение. Цель состоит в том, чтобы скорректировать вероятность увеличения набора данных p таким образом, чтобы выбранная эвристика соответствовала подходящему целевому значению.

Первая эвристика, r_v, выражает выходные данные для набора проверки относительно обучающего набора и сгенерированных изображений. Вторая эвристика, r_t, оценивает ту часть обучающего набора, которая получает положительные выходные данные дискриминатора.

Параметр p контролируется следующим образом:

- p инициализируется значением равным нулю;
- значение p корректируется один раз через каждые 4 мини-выборки на основе выбранной эвристики переобучения;

- если эвристика указывает на слишком большую/малую переобучаемость, p необходимо увеличить/уменьшить на фиксированную величину.

Фиксированная величина изменения p выбирается таким образом, чтобы значение p могло достаточно быстро увеличиваться с 0 до 1. Такой принцип называется адаптивным расширением дискриминатора (ADA).

В статье говорится, что r_v и r_t эффективны в предотвращении переобучения, и что они оба улучшают результаты по сравнению с лучшим фиксированным значением p, найденным по сетке.

Полученные улучшения объясняются следующим: без увеличения набора данных градиенты, которые генератор получает от дискриминатора, со временем становятся очень упрощенными — дискриминатор начинает обращать внимание только на несколько признаков, и генератор может свободно создавать бессмысленные изображения, а с ADA градиентное поле остается гораздо более детализированным, что предотвращает такое ухудшение.

ГЛАВА 5
ЭКСПЕРИМЕНТЫ И РЕЗУЛЬТАТЫ

5.1 Генерация изображений

5.1.1 DCGAN

Для обучения глубокой свёрточной генеративно-состязательной сети использовался набор данных, состоящий из приблизительно 9000 изображений, разбитых на 3 класса, в котором каждое изображение является КТ-срезом лёгких человека. Набор разбит на классы в зависимости от того, на каком уровне был сделан снимок. Размер исходных изображений - 512 × 512 пикселей.

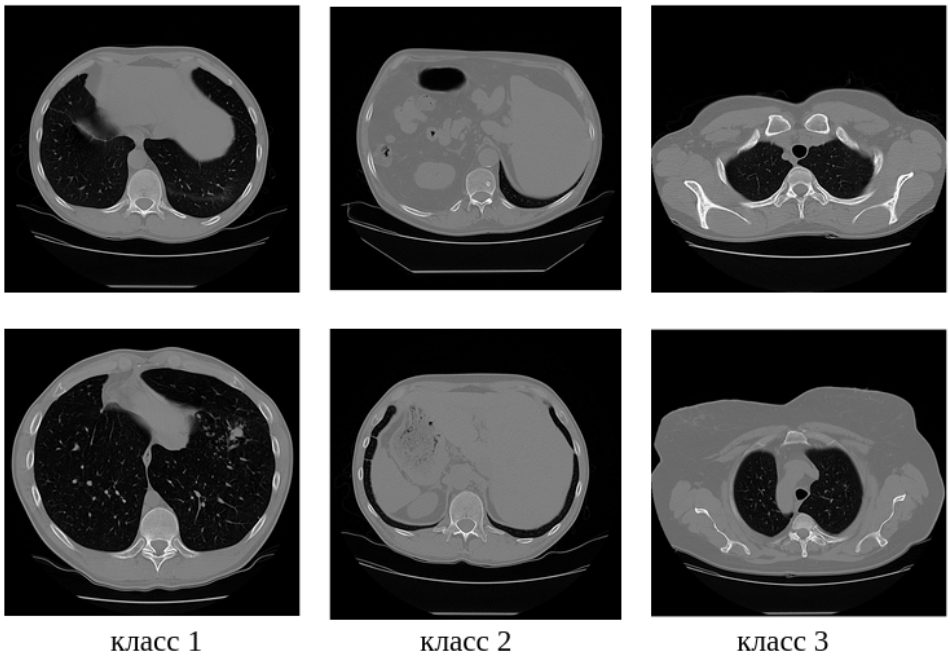

Рисунок 5.1 — Пример исходных данных

Ниже представлены изображения, сгенерированные с помощью DCGANs. Данный вид глубоких сетей не предназначен для генерации изображений невысокого разрешения. Поэтому сначала исходный набор данных был сжат до разрешения 64 × 64. Следовательно, полученные изображения также имеют разрешение 64 × 64.

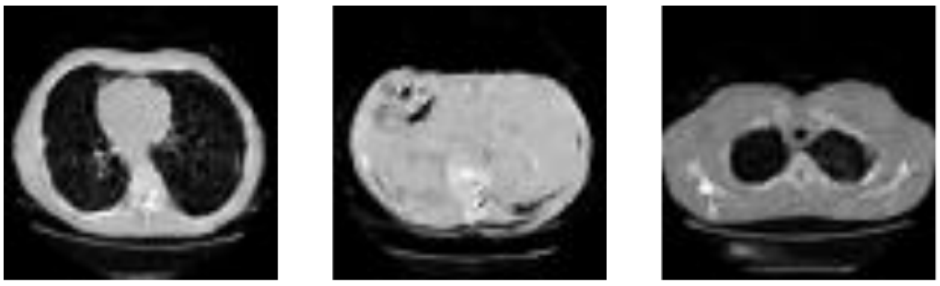

Рисунок 5.2 — Сгенерированные изображения (DCGANs)

Видно, что сеть неплохо восстанавливает форму и общие признаки исходных изображений, но всё же все полученные изображения низкого разрешения, а большинство из них имеют заметные искажения. Так же следует отметить, что затраты времени были не столь значительными для обучения и генерации, но на практике часто происходил коллапс моделей.

5.1.2 ProGAN

Ниже представлены результаты изображений, сгенерированных с помощью ProGANs, обученной на том же наборе данных, что и сеть в Разделе 4.1 Так как особенностью данной архитектуры является постепенное увеличение разрешения сгенерировоанного изображения, то качество изображения после каждого шага улучшается, что как раз и можно видеть на изображениях ниже:

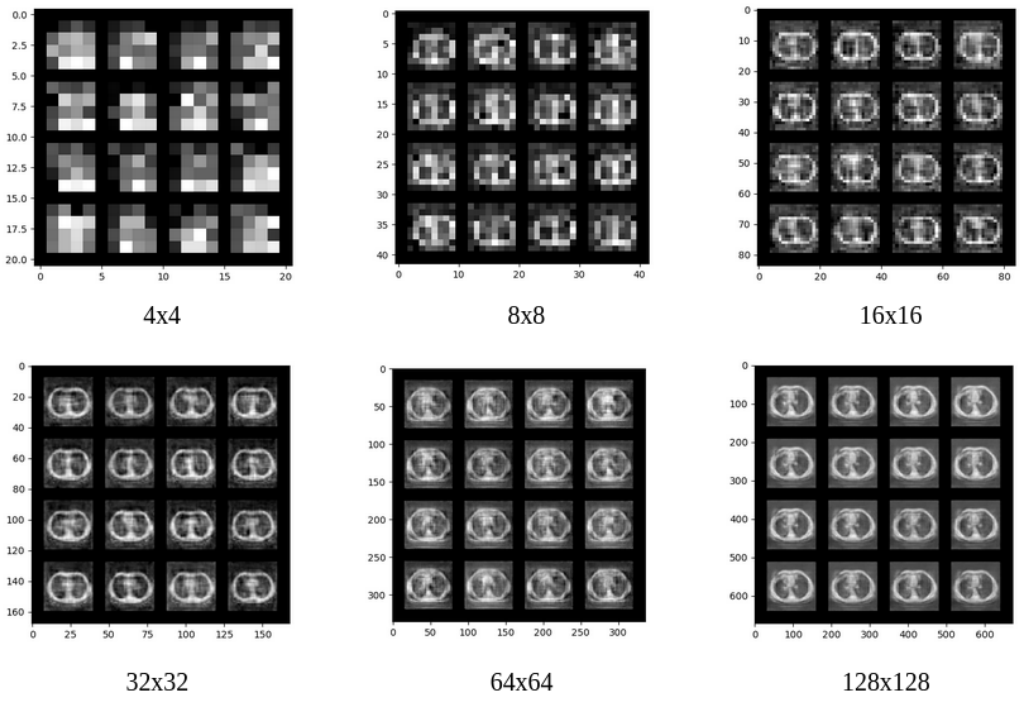

Рисунок 5.3 — Сгенерированные изображения(ProGANs)

Заметим, что изображение разрешением 128 × 128 пикселей уже достаточно похоже на исходное.

Но на практике были выявлены следующие проблемы:

- для обучения модели требуются большие вычислительные возможности;
- при создании изображений наблюдается коллапс модели, т.е. все генерируемые изображения схожи друг с другом, классового различия изображений уже нет (Рисунок 5.3).

5.1.3 StyleGAN2

Далее была обучена генеративно-состязательная сеть, основанная на стиле (StyleGAN2) с модификацией, о которой говорилось в разделе 4.4. Для обучения использовался исходный набор данных, но состоящий уже из 1500 изображений (для создания условий ограниченного набора данных для модели) и без меток классов.

В начале обучения на вход модели подается параметр kimg, который равен количеству тысяч изображений, на которых должен обучиться дискриминатор. Зависимость качества изображения от данного параметра отображена в следующем подразделе.

Были обучены сети со следующими параметрами:

- размерность 64x64, kimg = 500;
- размерность 128x128, kimg = 750;
- размерность 256x256, kimg = 500.

Ниже представлены часть сгенерированных изображений.

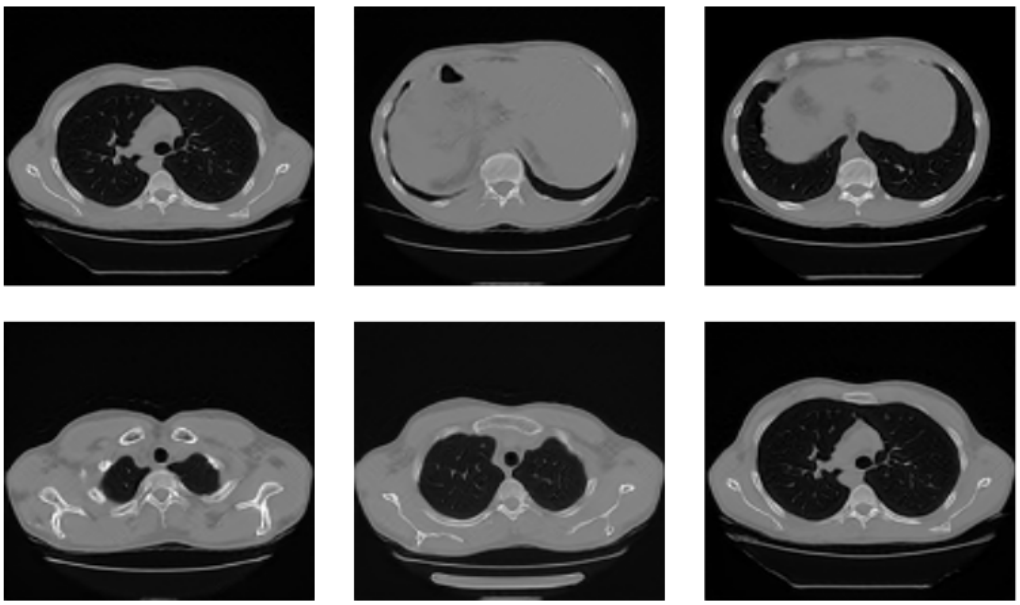

Рисунок 5.4 — Сгенерированные изображения(StyleGAN, 128x128)

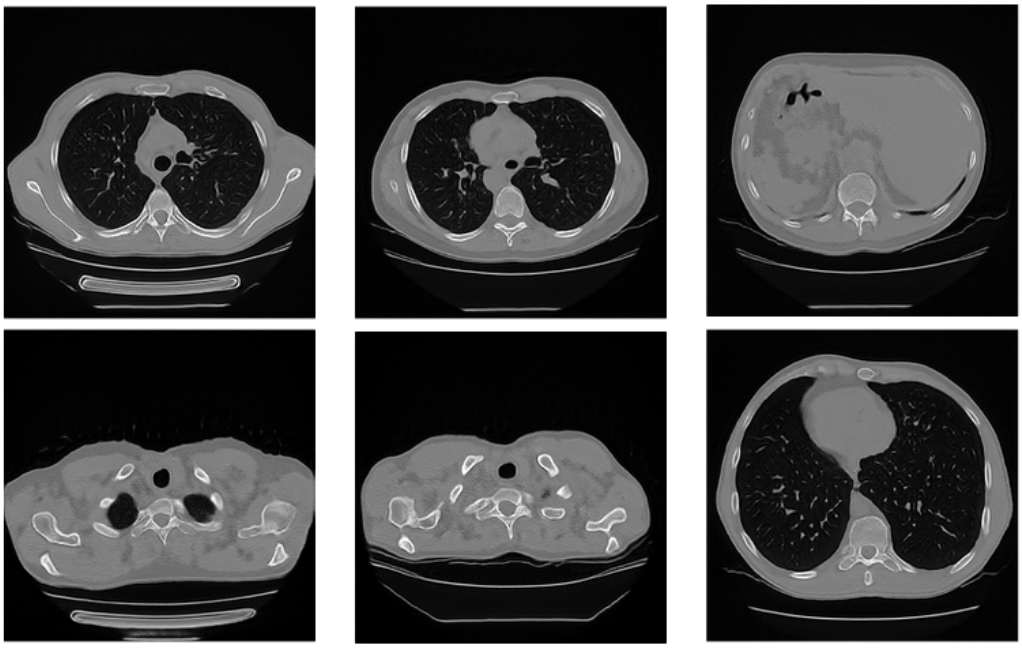

Рисунок 5.5 — Сгенерированные изображения(StyleGAN, 256x256)

Лучших результатов удалось получить при выставлении следующего набора параметров, отвечающих за увеличение исходного набора данных:

- xint max = 0.125;
- scale std = 0.2;
- rotate max = 1;
- aniso std = 0.2;
- brightness std = 0.2;
- contrast std = 0.5;
- noise std = 0.1;
- cutout size = 0.5;

Видно, что качество полученных изображений на порядок выше изображений, созданных другими сетями, и намного ближе к исходному набору данных.

Количественная метрика оценки качества рассмотрена в следующем разделе.

5.2 Оценка качества полученных изображений

В качестве оценки различимости была выбрана FID метрика (Fréchet Inception Distance). Данная метрика использует классификационую нейронную сеть Inception Net, чтобы отличать сгенерированные изображения от исходного набора данных, при этом измеряя Фреше расстояние между распределениями исходным и созданных изображений. Значения данной метрики лежит в интервале $[0, \infty]$, чем меньше данное значение, тем правдоподобнее сгенерированное изображение.

Результаты измерений данной метрики в зависимости от используемого метода и параметра kimg в StyleGAN для всех проведенных экспериментов при-

видены в таблице ниже.

Network	FID	kimg
DCGAN(64x64)	82.3	-
ProGAN(128x128)	138.3	-
StyleGAN(128x128)	55.3	400
StyleGAN(128x128)	33.4	750
StyleGAN(256x256)	46.9	400
StyleGAN(256x256)	39.6	500

Таблица 5.1 — Измерение FID метрики

Видно, что качество изображения напрямую зависит от размерности генерируемого изображения, т.к. при большей размерности сеть выучивает больше деталей изображения, что позволяют улучшить качество.

Лучшее качетво получено при самом большом значении параметра kimg и при наибольшем разрешении 256x256.

Зависимость качества получаемых изображений от параметра kimg при создании изображений размерносити 256x256 приведена на графике ниже.

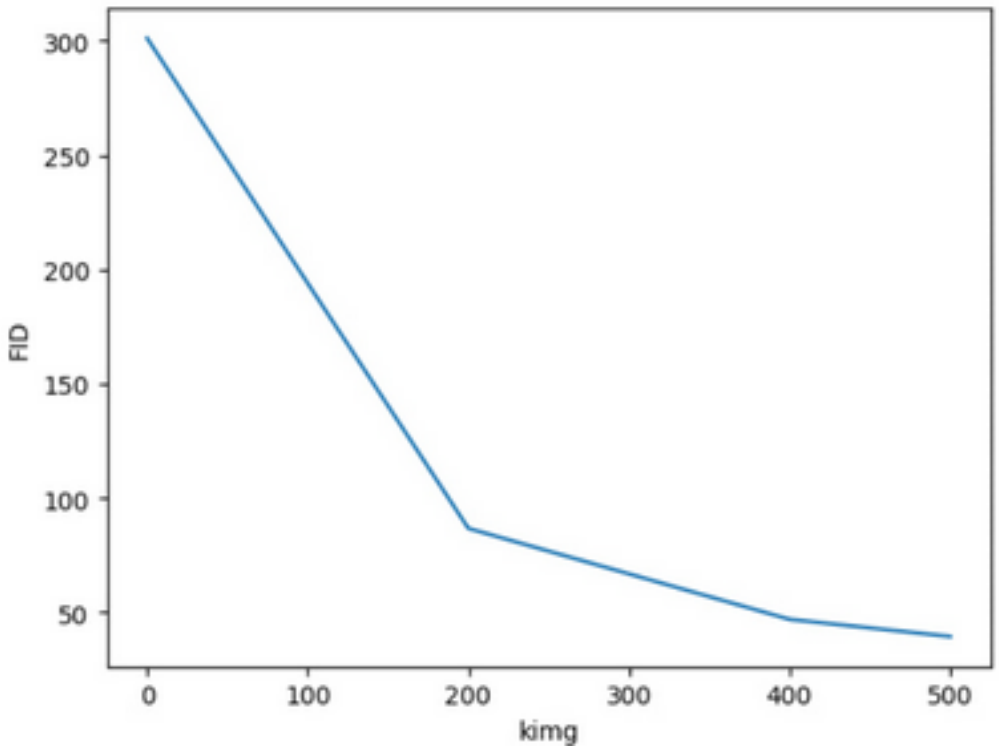

Рисунок 5.6 — График зависимости FID от kimg

Видно, что наблюдается зависимость качества получаемых изображений от данного параметра, следовательно, качество напрямую зависит от количества изображений, обработанных сетью.

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены и проанализированы такие архитектуры генерации изображений, как глубокие свёрточные генеративно-состязательные сети, прогрессивный рост генеративно-состязательных сетей и генеративно-состязательная сеть, основанная на стиле. Все сети позволили создать изображения на основе предоставленного набора КТ-снимков. Качество полученных наборов данных было измерено с помощью метрики начальное расстояние Фреше (FID).

Лучшие результаты были достигнуты при использовании сети StyleGAN с модификацией ADA, которая при обучении расширяет исходный набор данных.

Анализ полученных результатов позволяет сделать вывод, что генерация КТ-снимков, имеющих некоторые особенности, возможна и может показать неплохие результаты. Можно также сделать предположение, что при дальнейших изменениях архитектуры, учитывающих особенности конкретной задачи, возможно достигнуть хороших результатов.

Логическим продолжением выполненной работы является

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Generative Adversarial Nets / Ian J. Goodfellow, Jean Pouget-Abadie†, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
2. Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks / Alec Radford, Luke Metz
3. Progressive Growing Of GANs For Improved Quality, Stability, And Variation / Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen
4. Pros and Cons of GAN Evaluation Measures / Ali Borji
5. Generative Deep Learning / David Foster — O'REILLY, 2019 — 308 p.
6. A Style-Based Generator Architecture for Generative Adversarial Networks / Tero Karras, Samuli Laine, Timo Aila
7. Analyzing and Improving the Image Quality of StyleGAN / Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila
8. Towards Principled Methods for Training Generative Adversarial Networks / Martin Arjovsky, Léon Bottou
9. Improved Consistency Regularization for GANs / Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, Han Zhang
10. Training Generative Adversarial Networks with Limited Data / Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, Timo Aila
11. AmbientGAN: Generative models from lossy measurements / Ashish Bora, Eric Price, Alexandros G. Dimakis