

# Лабораторная работа: применение GAN для генерации изображений из датасета

Федоров Артем Максимович

Январь 2025

## Abstract

Настоящий документ представляет отчёт о выполненной работе в рамках вступительного испытания в лабораторию байесовских методов. В работе исследуются современные архитектуры и методы обучения генеративных состязательных сетей (GAN)<sup>1</sup> для генерации изображений на произвольных датасетах. В качестве результатов представлены фреймворк для обучения GAN, методы логирования и оценки качества обучаемых моделей, а также сами обученные модели и сгенерированные ими изображения с последующим сравнительным анализом результатов.

## 1 Постановка задачи

Пусть задано множество изображений  $\mathcal{I} = \{x_i\}_{i=1}^N$ , где  $x_i \in \mathbb{R}^{3 \times d \times d}$ . Определим множества:

$$\mathcal{I}_{train}, \mathcal{I}_{test} : \mathcal{I}_{train} \cap \mathcal{I}_{test} = \emptyset, \mathcal{I}_{train} \cup \mathcal{I}_{test} = \mathcal{I}$$

Для заданных подмножеств рассмотрим параметрическую генеративную модель  $G_{\theta_g} : \mathbb{R}^h \rightarrow \mathcal{I}$  и поставим задачу оптимизации функционала  $J(\theta_g, \mathcal{I}_{train}) \rightarrow \min_{\theta_g}$ . Валидация результатов обучения будет осуществляться с использованием метрик качества  $M_i$  и эмпирической оценки. В дальнейшем распределение реальных данных обозначим как  $\pi(x)$ .

## 2 Данные задачи

В качестве данных для проведения экспериментов выбран датасет изображений размерности  $3 \times 64 \times 64$ , содержащий примеры различных блюд, таких как паста, пироги, мороженое, пончики, рис и другие. Датасет разделён на две подвыборки: обучающую (train) и тестовую (test) для обучения и валидации моделей соответственно, с общей численностью 100,957 изображений (98,937 в обучающей и 2,020 в тестовой выборках).

Изображения изначально классифицированы по типам блюд, при этом распределение классов в обеих подвыборках совпадает ( $JSD = 2.4 \cdot 10^{-4}$ ). Исходя из данного наблюдения, задача генерации изображений решается без учёта меток классов, поскольку алгоритм восстановления плотности должен также восстановить априорное распределение классов. Примеры данных рис. 1.



Figure 1: Примеры изображений датасета

<sup>1</sup>GAN – Generative Adversarial Network – архитектура генеративных сетей из класса моделей с неявной плотностью распределения (implicit density).

### 3 Оценки качества валидации

Для оценки качества обучения модели будут использованы следующие метрики:

1. **Fréchet Inception Distance (FID)**<sup>2</sup> – метрика, основанная на расстоянии Фреше между распределениями признаков изображений, извлечённых с использованием предобученной модели InceptionV3.
2. MS-SSIM – расширение метрики SSIM (Structural Similarity Index Measure), основанное на структурном сходстве изображений, вычисляемое по парам изображений исходя из компонентов яркости, контраста и структуры.
3. Kernel-Inception Distance (KID) – метрика, основанная на расстоянии между распределениями признаков изображений, полученных с помощью предобученной модели InceptionV3, вычисляемая с использованием ядерной функции.

*Подробнее про используемые метрики качества см. в разделе B.*

### 4 Эксперименты

**Конфигурация системы:**

- ОС: Ubuntu 20.04.3 LTS
- CPU: Intel Xeon Gold 6336Y 8 vCPU
- GPU: NVIDIA GeForce RTX 2080 Ti (11 GB)
- RAM: 16 GB
- CUDA: 12.4

**Общие параметры обучения:**

- Число обучающих шагов: 2,000,000
- Шаг логирования потерь: 200
- Шаг логирования весов и метрик: 15,000

*Подробнее про используемые модели см. в разделе A.*

#### 4.1 Первые запуски «VerySimple...»

Для получения BaseLine была выбрана архитектура, предложенная в качестве референса в задании. При этом параметры сети не изменялись относительно предложенного конфига.

В качестве первого метода улучшения качества работы и обучения GAN принято решение использовать методику зашумления данных, призванную предотвратить коллапс обучения оригинального GAN. Причина использования данной механики заключается в предположении, что зашумление данных понижает вероятность получения пустого пересечения носителей распределений генерации и реальных данных. Процедура зашумления производится путём добавления гауссовского шума с центром в 0 и дисперсией  $\sigma_t = \sigma_0 \cdot \gamma_t$  (рис. 2).

Архитектура показала экстремально малую скорость обучения (553 часа для полного процесса), ввиду чего обучение принудительно завершено и далее в отчёте не приводится.

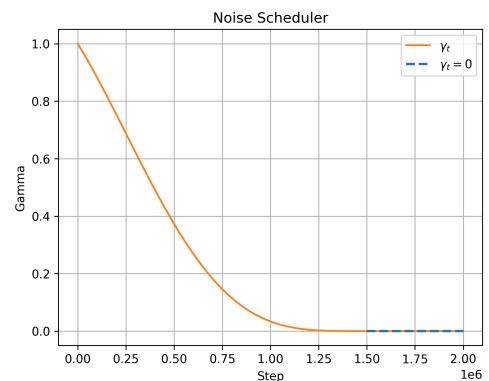


Figure 2: Зависимость коэффициента шума  $\gamma_t$  от итерации обучения

<sup>2</sup>FID – основная метрика качества генерации в данном исследовании.

## 4.2 Обучение WGAN-GP

На замену базовой возвращём WGAN-GP на сверточных сетях. Для определения конфигурации модели заметим, что изначальное предположение о расположении всех изображений датасета на малом аффинном многообразии говорит о том, что GAN должен обучать два объекта: базис (ЛНЗ оболочку векторов, span) и метод определения коэффициентов этих векторов. Предположим, что задача определения базиса многообразия полностью лежит на плечах, в нашем случае, сверточных слоёв. Значит, латентное пространство задаёт коэффициенты для линейной комбинации  $\Rightarrow$  размерность обучаемого многообразия не больше размерности  $\mathcal{Z}$ .

Случайным образом выберем 100 изображений каждого класса в  $\mathcal{I}_{train}$ , вытянем все изображения, сохранив каналы, и образуем матрицу  $X \in \mathbb{R}^{101 \times 100 \times 12288}$ , где 12288 – размерность вытянутого изображения. Для неё определим поведение сингулярных чисел (рис. 3). Видно, что они ( $\hat{\sigma}_i$ ) стремительно убывают, и в районе 100-150 сингулярного числа их можно считать нулевыми. Это, в свою очередь, говорит о малой эффективной размерности  $\mathcal{I}$ .

В предположении, что примерно уже 100-150 эффективных векторов должно хватить для качественной генерации, сошлемся на ряд статей, декларирующих увеличение скорости обучения и выразительности модели с увеличением числа параметров<sup>3</sup>. Возьмём в два раза большее размерность  $dim\{\mathcal{Z}\} = 256$  и достаточно большое  $f = 256$ . Получаемый размер моделей в параметрах составляет 44М для критика и 52М для генератора. Далее данная модель будет именоваться **large-gan**. При этом использовалась поканальная нормализация изображения с параметрами, взятыми из  $\mathcal{I}_{train}$ , и не использовалась функция активации  $\tanh$  на выходе генератора.

### 4.2.1 Процесс обучения large-gan

**Large-gan** обучался на протяжении 15 часов с зашумлением трейна, за которые прошёл 1.2М шагов обучения и был превентивно остановлен ввиду своего неустойчивого обучения и плохого качества генерации. На графиках 4 видно, что при условии стабильного роста критика в начале обучения, его падение со временем отсутствует, что является одним из признаков неспособности генератора подстроиться под обучающие данные. Подтверждение этому видно на графике потерь самого генератора 4а: тренд на уменьшение лосса присутствует, тем не менее с числом итераций скорость его уменьшения падает, само же обучение очень нестабильно, что видно по осцилляциям графика.

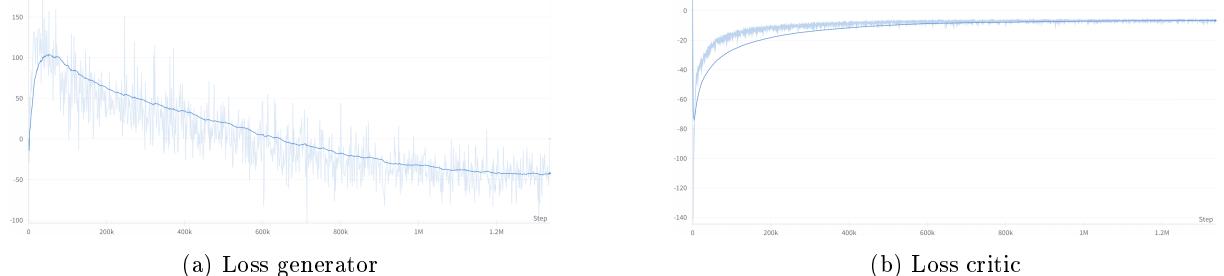


Figure 4: Поведение функционалов потерь в процессе обучения

При этом заметим, что распределения весов критика по слоям также становятся менее неустойчивыми по ходу обучения (рис. 5). Присутствует колоколообразное распределение (можно так сказать даже для 3 блока) с ярко выраженной модой, но также заметны и относительно тяжёлые «хвосты», которых быть не должно для теоретической обоснованности работы WassersteinGAN. Значит, GP не справляется для такой сложной переобученной модели.

<sup>3</sup>Такие статьи: Scaling description of generalization with number of parameters in deep learning и Understanding How Over-Parametrization Leads to Acceleration: A case of learning a single teacher neuron

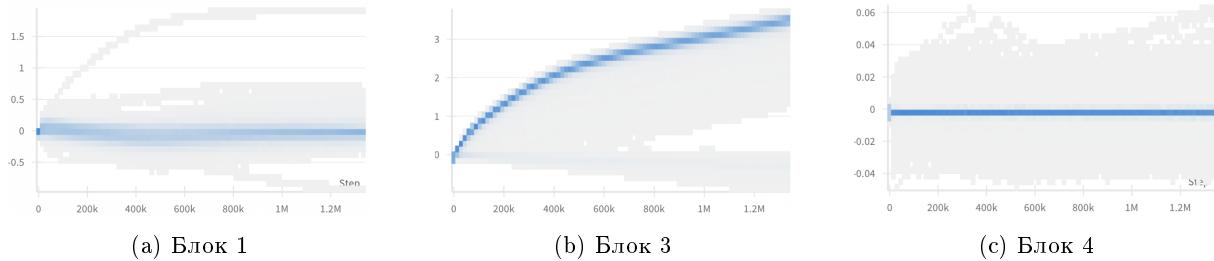


Figure 5: Распределение весов сверточных слоёв `large-gan`

#### 4.2.2 Качество работы large-gan

Данная модель показала свою несостоительность генерировать качественные изображения из обучающего датасета. **Large-gan** требовал относительно огромное время (сравнивая с дальнейшими моделями), чтобы начать генерировать что-либо, кроме шума. Лишь через несколько часов модель смогла начать генерировать области на изображениях с градиентами цвета (ассоциируемыми между собой соседними пикселями) вместо случайных раскрашенных пикселей.

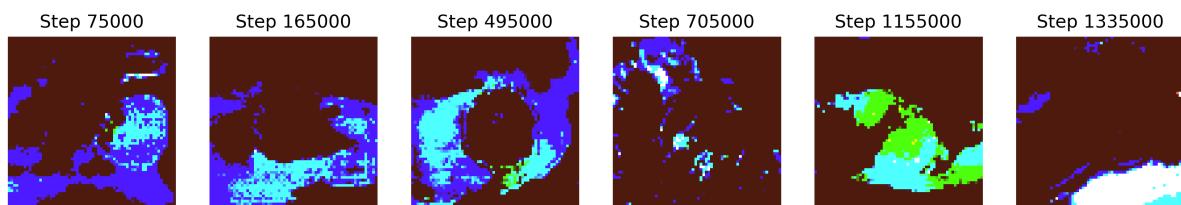


Figure 6: Лучшие изображения с нескольких этапов обучения `large-gan`

При помощи метрики MS-SSIM отберём самые лучшие сгенерированные изображения на нескольких вручную отобранных эпохах обучения. Метрика изображения получается из сравнивания её со случайно выбранными 50 изображениями валидации (выбор нестрогий). На изображении 6 видно, что полученная модель генерирует крайне плохие по качеству изображения: большинство пикселей заняты константным значением, цвета не согласуются с цветами в  $\mathcal{I}$ , а также присутствует множество артефактов на изображениях (все изображение сплошной артефакт).

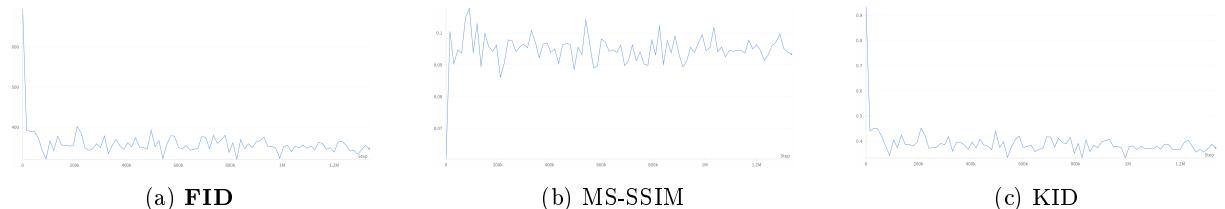


Figure 7: Изменения метрик качества модели **large-gan**

На графиках изменения оценок качества (рис. 7) мы можем видеть, что метрики **FID** и **KID** падают (хотя и остаются крайне большими на всём протяжении обучения), **MS-SSIM** растет (что и ожидается при падении прочих). Данное наблюдение контрастирует с приведенными выше примерами плохой генерации изображений.

## Выводы:

- Подобное поведение модели **large-gan** говорит об излишней параметризации модели на столь малом датасете простых объектов (картинки  $3 \times 64 \times 64$ ). Очевидно, требуется упростить модель, что также было предпринято исходя из оценки эффективной размерности данных.
  - Возможной проблемой генерации выступает специфичный процесс нормализации данных. Мы не ограничиваем область генерируемых значений для генератора. Обращаясь к примерам решений на GitHub<sup>4</sup>, можно найти успешные реализации с другими методами нормализации.

---

<sup>4</sup>OpenAI-ImprovedGAN, foodGAN, flowerGAN

### 4.3 Обучение small-gan

Упрощение модели **large-gan** далее будем упоминать как конфигурацию **small-gan**. Для нее размерность латентного пространства уменьшена вдвое, что отбиралось по принципу локтя на графике сингулярных чисел (рис. 3), что теперь составляет  $\dim\{\mathcal{Z}\} = 100$ . Параметр  $f$  был уменьшен радикально в 4 раза и составляет  $f_{small} = 64$ . Все так же используем зашумление данных. Однако меняем метод нормализации данных:  $mean = \text{const} = [0.5, 0.5, 0.5]$ ,  $std = \text{const} = [0.5, 0.5, 0.5]$ , выход генератора ограничивается  $\tanh$ .

#### 4.3.1 Процесс обучения small-gan

Модель обучалась на протяжении 23 часов и прошла 2M шагов. В отличии от предыдущей модели, **small-gan** показала более стабильное обучение генератора, проблема критика с оставшимся лосом на плато решена, теперь он уменьшается после определенного момента (рис. 8).

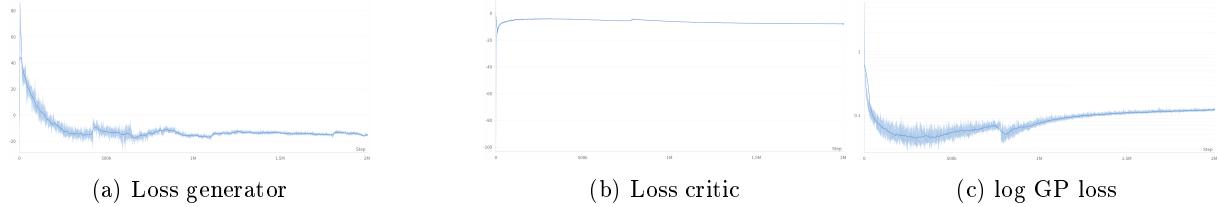


Figure 8: Поведение функционалов потерь **small-gan**

На графиках весов критика так же видны положительные изменения, выражющиеся в уменьшении дисперсии и носителя распределений (рис. 9). Блоки 0, 1 и 4 показывают тенденции к уменьшению носителя, веса более равномерно распределены внутри «колокола». Распределения блоков 2 и 3, напротив, склонны к увеличению носителя, однако веса заметно меньше по модулю весов **large-gan** (очевидно, это повлияло на малый рост GP loss 8с).

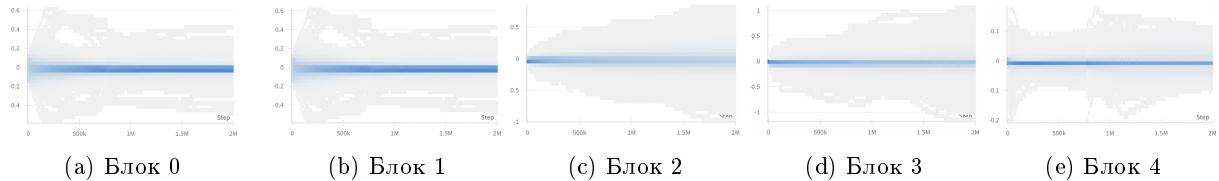


Figure 9: Поведение весов Conv блоков критика **small-gan**

#### 4.3.2 Качество работы small-gan

Улучшения архитектуры затронули не только обучение, но и итоговое качество генерации. Теперь метрики FID и KID принимают значения в разы меньшие таковым у **large-gan** (рис. 10). Однако с MS-SSIM происходит неожиданное изменение: в процессе обучения данная оценка падает.

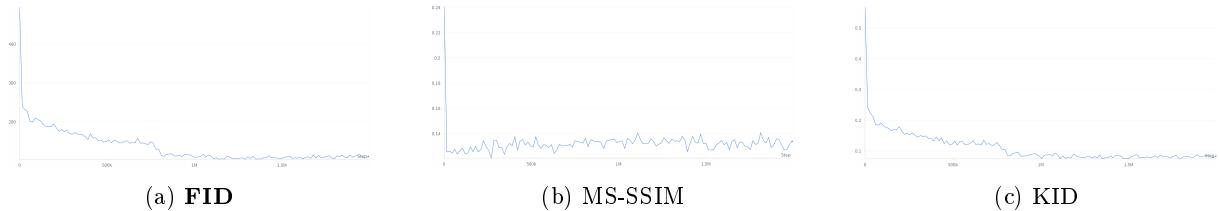


Figure 10: Поведение оценок качества генерации модели **small-gan**

Поведение MS-SSIM должно следовать из условия, что GAN не учится копировать  $\mathcal{I}_{train}$ , а лишь подражать ему. В данном случае это выражается в том, что генерация имеет те же внешние признаки, что и оригинал, тем не менее не имеет схожих зрительных черт – отличаются объекты, их размещение, сочетание цветов и так далее, что не затрагивает FID и KID, но отражается на MS-SSIM. Далее данный факт опускается.

На графиках наблюдается перелом по метрикам FID и KID примерно в 600-700 тысячных шагах обучения, что отражается на графике потерь генератора (рис. 8а) повышенной осцилляцией. При этом на метрику ms-ssim это никак не повлияло. Возможной причиной такого поведения является адаптация генератора под критика, избавление от каких-то артефактов генерации, незаметных человеку, но явных для сверточной сети.

#### 4.3.3 Примеры генерации small-gan

Данная конфигурация способна начать генерировать уже на первом логировании изображений, и достаточно быстро выходит на уровень изображений с различными очертаниями еды, что представлено на изображении 11. Можно увидеть, что примерно на рубеже 500 и 900 тысячных итераций пропадают шумовые артефакты на изображениях.



Figure 11: Прогрессия генерации модели **small-gan**

На изображении 22 приведены samples генерации полностью обученного **small-gan**. Данная нейросеть уже способна генерировать качественные изображения, большая часть которых не отличима от оригинала (проверено на своих знакомых, что является частью эмпирического теста). Мы не наблюдаем ярковыраженного явления «mode collapsing», однако при этом наблюдаем изображения с похожими очертаниями и цветовыми гаммами, что так же является проблемой генеративной модели. Вместе с этим и качество генерации не идеально – присутствуют изображения с артефактами либо с непонятными блюдами.

#### 4.4 Улучшение обучения small-gan. Multistep-critic и no-noise-gan

После скачка качества генерации на **small-gan** принято решение оставить именно данную конфигурацию модели. Тем не менее, у нее остаются проблемы как с обучением, так и с конечным качеством генерации, для решения которых рассматриваются два метода:

- **Отказ от добавления шума:** ожидается более стабильное обучение, уменьшение числа артефактов на изображениях, а так же ухудшение разнообразия и метрик качества. Модель **no-noise-gan**.
- **Несколько оптимизационных шагов критика:** на каждом шагу обучения предлагается производить 3 шага обновления критика и только один генератора<sup>5</sup>. Ожидается улучшение качества генерации, вместе с ухудшением устойчивости обучения. Модель **multistep-critic**.

##### 4.4.1 Сравнение обучения моделей

На графиках 12 приведено сравнение функционалов потерь трех рассмотренных моделей. Видно, что несмотря на изначальные предположения, именно **multistep-critic** имеет самый низкий итоговый лосс для генератора и критика, хотя обучение генератора все же не обладает монотонным характером. Для модели **no-noise-gan** характерна меньшая осцилляция лоссов, что и ожидалось.

Из общих наблюдений видно, что обе модели смогли превзойти **small-gan** по значениям функционалов, а значит:

<sup>5</sup>Лекция Александра Оганова ММП 2024, выбор константного числа шагов на каждом step github

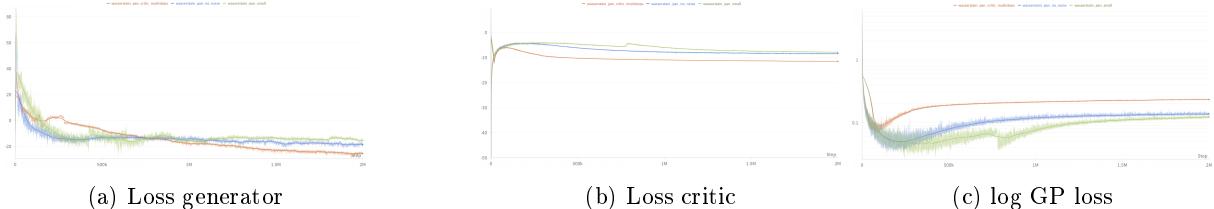


Figure 12: Сравнение функционалов потерь трех моделей: оранжевый –**multistep-critic**, синий –**no-noise-gan**, зеленый –**small-gan**

- Добавление шума должно сопровождаться добавлением multistep процедуры для критика.
- Большой шум замедляет обучение критика, что приводит к ухудшению обучения и генератора.

Положение по распределению весов для двух моделей остается таким же, что и для предшествующей модели **small-gan**

#### 4.4.2 Качество работы multistep-critic и no-noise-gan

При сравнении оценок качества моделей 15 видна доминация двух новых моделей над **small-gan** по всем двум метрикам FID и KID. При том, что все три модели выходят на сравнительно одно плато, **multistep-critic** и **no-noise-gan** делают это значительно быстрее и с лучшими значениями в конце.

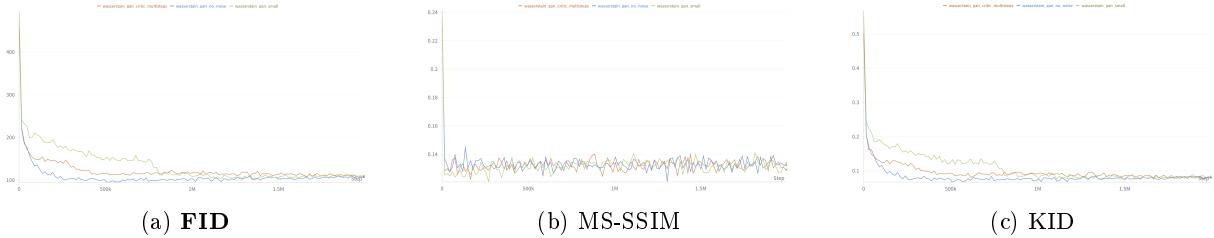
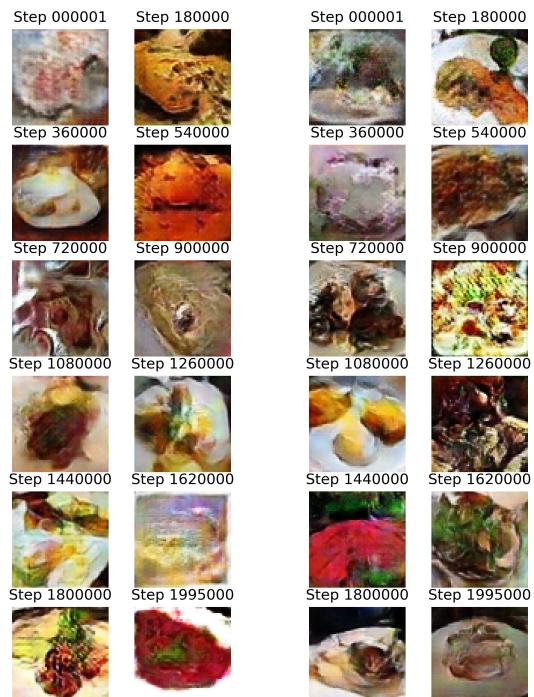


Figure 13: Сравнение метрик качества: оранжевый –**multistep-critic**, синий –**no-noise-gan**, зеленый –**small-gan**

#### 4.4.3 Примеры генерации multistep-critic и no-noise-gan

Примеры генерации **no-noise-gan** приведены на в секции C.2, для **multistep-critic** – в секции C.3 соответственно. Мы видим улучшение качества генерации на глаз, по сравнению с моделью **small-gan** появилось больше разнообразия, больше блюд имеют различимые очертания, меньше артифактов.

На изображениях 14a и 14b приведена прогрессия генерации изображений моделями. Наблюдения лишь подтверждают ранее указанные предположения по метрикам. Модели действительно начинают генерировать качественные изображения быстрее, чем **small-gan**, в них раньше пропадает большинство артефактов, свойственных предыдущей модели.



7 (a) Прогрессия генерации **no-noise-gan** (b) Прогрессия генерации **multistep-critic**

## 4.5 Добавление fade-in

Одним из способов эффективного обучения GAN является процедура Fade-In, заключающаяся в постепенном смешивании выходов низко- и высокоуровневых слоев генератора/дискриминатора, увеличивая разрешение изображений по мере обучения. Применим данную практику и обучим модель fade-in-gan с добавлением малого шума и без мультистепа критика на 3 шагах fade-in – 3, 4 и соответственно 5 блоках.

### 4.5.1 Процесс обучения fade-in-gan

Обучение данной модели сопровождается скачками лоссов на каждом шагу fade-in, их большой осцилляцией, что сильно отличает модель от предыдущих (рис. ??). Это можно объяснить требованием обучить адаптеры моделей на каждом шагу fade-in. В особенности проблемным кажется этап обучения на генерации изображений  $3 \times 32 \times 32$

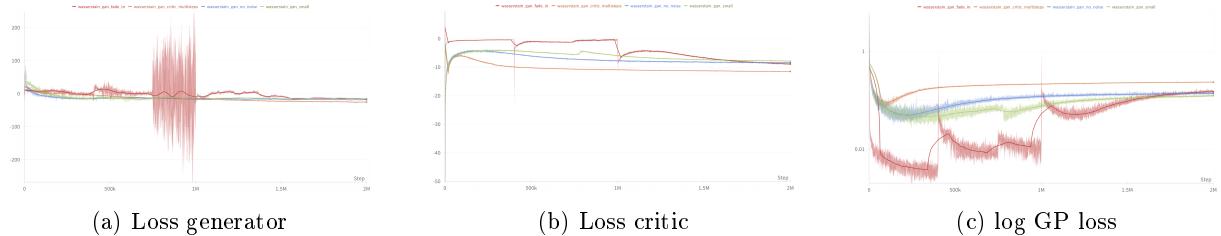


Figure 15: Сравнение метрик качества: оранжевый –**multistep-critic**, синий – **no-noise-gan**, зеленый – **small-gan**, красный – **fade-in-gan**

### 4.5.2 Качество работы fade-in-gan

Для получения метрик качества **fade-in-gan** в методы, высчитывающие оценки, подавались валидирующие изображения того же размера, что и генерируемые GAN, что достигалось линейной интерполяцией изображений. Fade-in показывает себя с лучшей стороны, к концу обучения **fade-in-gan** превосходит все остальные по метрикам **FID** и **KID**.

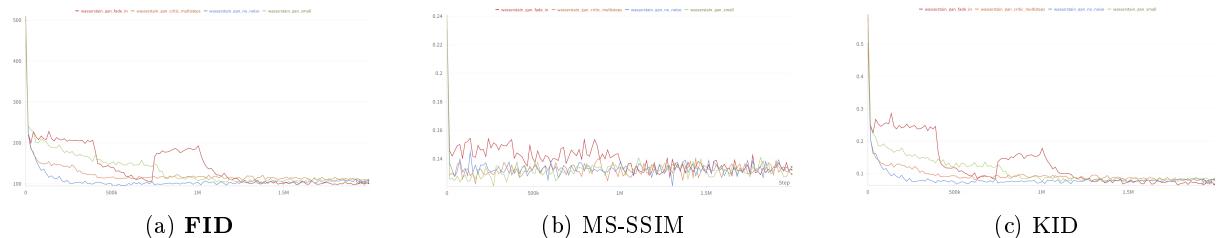


Figure 16: Сравнение метрик качества: оранжевый –**multistep-critic**, синий – **no-noise-gan**, зеленый – **small-gan**, красный – **fade-in-gan**

### 4.5.3 Примеры генерации fade-in-gan

Подробнее примеры генерации приведены в секции C.4. Прогрессия генерации модели **fade-in-gan** приведена далее на изображении 17. На прогрессии генерируемых изображений четко видны моменты, где GAN генерировал изображения  $3 \times 16 \times 16$ ,  $3 \times 32 \times 32$  и  $3 \times 64 \times 64$

## 5 Выводы

В данном исследовании были затронуты самые широкоиспользуемые современные методы обучения GAN, применение которых оправдалось инкрементальным улучшением качества генерации и отобразилось на оценках качества. В таблице 1 приведено сравнение моделей по метрикам **FID**, **MS-SSIM** и **KID** после всех экспериментов. Основной метрикой исследований является **FID**, по



Figure 17: Прогрессия генерации модели **fade-in-gan**

которой выигрывает безоговорочно **fade-in-gan** (ожидаемо, по метрике KID так же соблюдается его доминация).

В итоге мы получили две лучших модели, отличающиеся от остальных своим разнообразием и отсутствием шума – **fade-in-gan** и **multistep-critic**.

Модель	Размер модели	Время обучения	FID	MS-SSIM*	KID
small-gan	2.7M/7.5M	23 часов	110.56	0.1339	0.0839
multistep-critic	2.7M/7.5M	54 часов	108.09	0.1291	0.0817
no-noise-gan	2.7M/7.5M	28 часов	106.88	0.1323	0.0797
<b>fade-in-gan</b>	2.7M/7.6M	28 часов	<b>101.45</b>	<b>0.1342</b>	<b>0.0664</b>

Table 1: Сравнение моделей по метрикам FID, MS-SSIM и KID  
\* – в данном исследовании MS-SSIM показала себя неинформативной

## A Архитектура генеративной модели. GAN

Стратегия генерации объектов на основе набора реальных данных с использованием генеративных состязательных сетей (GAN) была впервые предложена в статье Generative Adversarial Networks. Данная методология предполагает обучение двух нейронных сетей: генератора и дискриминатора. В дальнейшем нейросети обозначаются как  $G_{\theta_g} : \mathbb{R}^h \rightarrow \mathcal{I}$  и  $D_{\theta_d} : \mathcal{I} \rightarrow \mathbb{R}$  соответственно, где  $\mathcal{I} \subseteq \mathbb{R}^{3 \times d \times d}$  — пространство изображений,  $h$  — размерность латентного пространства и  $h \ll 3 \times d \times d$ .

Задача обучения заключается в попаренной оптимизации каждой сети с соответствующими функциями потерь:

- $G : \exists D_{\hat{\theta}_d} \Rightarrow \theta_g = \arg \min_{\theta} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\hat{\theta}_d}(G_{\theta}(z)))$
- $D : \exists G_{\hat{\theta}_g} \Rightarrow \theta_d = \arg \max_{\theta} \mathbb{E}_{x \sim \pi(x)} \log D_{\theta}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta}(G_{\hat{\theta}_g}(z)))$

Таким образом, конечная задача сводится к нахождению такого преобразования из семейства параметрических архитектур  $G$ , которое переводит распределение  $p(z)$  в распределение  $\pi(G(z))$  посредством обучения генератора под «обман» дискриминатора. Основное предположение, лежащее в основе GAN, заключается в том, что множество реальных изображений  $\mathcal{I}$  является областью малоразмерного аффинного многообразия в высокоразмерном пространстве изображений  $\mathbb{R}^{3 \times d \times d}$ . В процессе обучения GAN стремится “натянуть” распределение латентного пространства  $\mathcal{Z}$  на данное многообразие.

Далее предполагается, что распределение латентных переменных является нормальным:  $p(z) \sim \mathcal{N}(0, I)^6$ . В результате, задача попаренной оптимизации формулируется как поиск седловой точки функционала:

$$\min_G \max_D \mathbb{E}_{x \sim \pi(x)} \log D(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D(G(z)))$$

### A.1 Проблемы обучения классического GAN. Wasserstein GAN

Предложенная модель GAN для генератора основывается на оптимизации дивергенции Дженсена-Шеннона<sup>7</sup>, представленной следующим образом:

$$\log 4 + 2JSD(\pi | p_g) \rightarrow \min_G$$

В свою очередь данный функционал является крайне нестабильным и легко приводит к коллапсу GAN по той же причине, что и делает GAN работающим методом генерации — в случае равенства вероятностной меры множества пересечения носителей двух распределений (реального и генерируемого) JS равняется константе  $\ln 2$ , что приводит к коллапсу обучения. Решение обозначенной проблемы предлагается в статье Wasserstein GAN, приводящей новый функционал для обучения GAN на основе Wasserstein distance. Обращающаяся к теореме ККТ над функциональными пространствами распределений, теорема Канторовича-Рубинштейна приводит дуальную постановку задачи для GAN:

$$\min_G W(\pi || p) \approx \min_G \max_{\phi \in \Phi} [\mathbb{E}_{\pi(x)} f(\mathbf{x}, \phi) - \mathbb{E}_{p(\mathbf{z})} f(G(\mathbf{z}, \theta), \phi)]$$

Место дискриминатора  $D$  с областью значений  $[0, 1]$  отводится функции  $f$  — критику, с областью значений  $\mathbb{R}$ . В теореме приводится требование на Липшицевость функции критика, что в оригинальной статье решается путем ограничения чебышевской нормы параметров критика, что является крайне неудачным способом и не разбирается в данной работе. Вместо такого подхода рассматривается применение Gradient-Penalty, приведенный в статье Improved Training of Wasserstein GANs по условию того, что  $f$  — 1-Липшицева. Упрощая условие только на точки, лещащие

<sup>6</sup>Выбор нормального распределения обоснован общими наблюдениями из многочисленных исследований и рекомендациями, приведенными в сборнике советов по обучению GAN ganhacks на GitHub.

<sup>7</sup>Jensen-Shannon divergence — wiki.

в наименьшем покрытии обоих носителей распределений, получаем итоговый функционал для обучения WGAN-GP:

$$\min_G \max_f \left[ \underbrace{\mathbb{E}_{\pi(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x}|\theta)} f(\mathbf{x})}_{\text{original critic loss}} - \lambda \underbrace{\mathbb{E}_{U[0,1]} \left[ (\|\nabla_{\hat{\mathbf{x}}} f(\hat{\mathbf{x}})\|_2 - 1)^2 \right]}_{\text{gradient penalty}} \right], \hat{\mathbf{x}} = \epsilon \mathbf{x} + (1 - \epsilon) G(z), \epsilon \sim \mathcal{U}[0, 1]$$

В связи с ограниченными вычислительными ресурсами (аренда GPU-серверов), обучение каждой модели является затратным как по времени, так и по финансовым затратам. По результатам нескольких тестовых прогонов стандартной модели, предлагаемой в качестве базовой (baseline) в данной работе, было принято решение использовать WGAN-GP в качестве основной архитектуры для обучения генеративной модели. Все дальнейшие методики обучения GAN будут основываться на сравнении WGAN-GP с использованием сверточной (Convolutional) архитектуры. 18

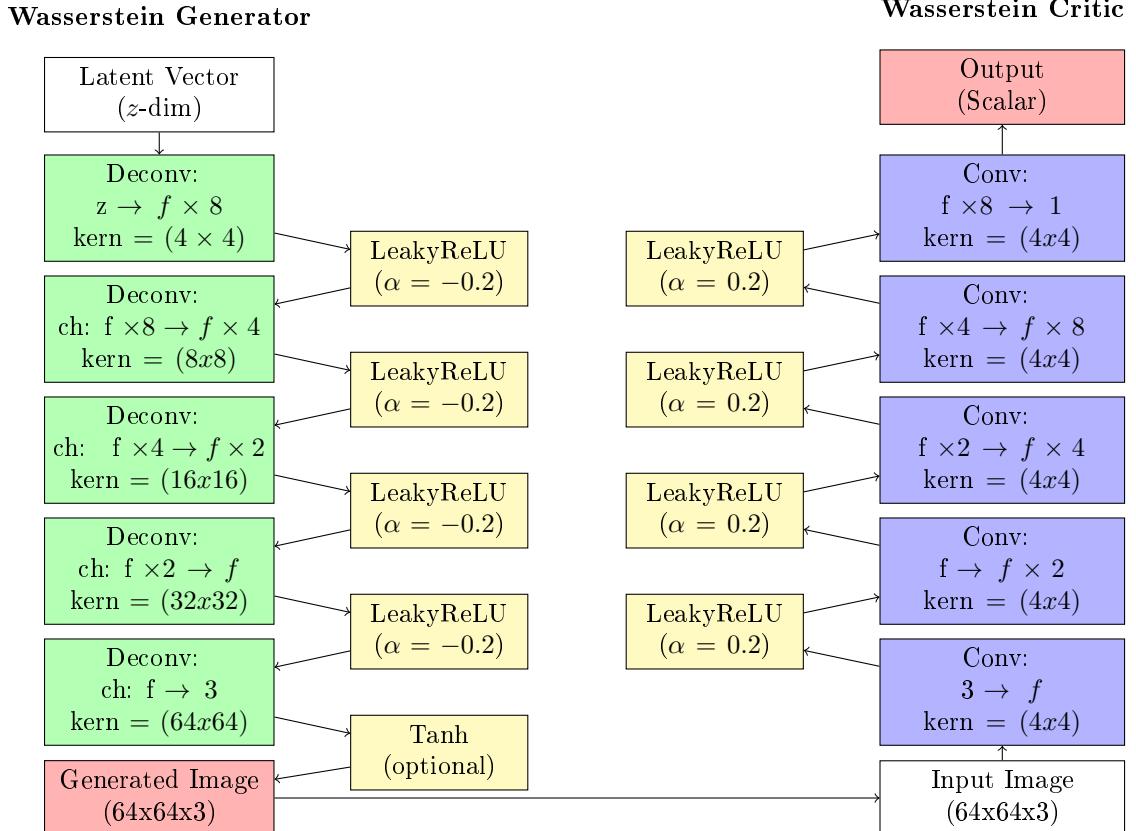


Figure 18: Архитектура WGAN на ConvolutionalNets

## B Оценки качества

Подробное описание с причинами использования оценок качества.

### B.1 Fréchet Inception Distance

FID измеряет различие между распределениями признаков реальных ( $P_r$ ) и сгенерированных ( $P_g$ ) изображений. Признаки извлекаются из предпоследнего слоя предобученной на *ImageNet* модели *Inception-v3*, после чего вычисляется расстояние Фреше при предположении о нормальном распределении признаков (расстояние между двумя многомерными гауссовыми распределениями).

*Реализация метрики взята из библиотеки torch-fidelity.*

## B.2 MS-SSIM

MS-SSIM (Multi-Scale Structural Similarity Index) является расширением популярной метрики SSIM (Structural Similarity Index Measure), учитывающим различия между изображениями на нескольких пространственных масштабах. Это повышает чувствительность метрики к мелким и крупным артефактам. MS-SSIM сравнивает близость изображений не по технической части, а по «семантической», из-за чего даже плохое значение метрики (малые значения) могут указывать не сколько на плохую генерацию, сколько на то, что генерируемые изображения слишком отличны от изначальных. Данная метрика также основывается на предобученных моделях, однако использует интерпретируемые признаки.

По указанным причинам было принято решение дополнить оценку качества FID метрикой MS-SSIM, учитывая ее потенциально большую чувствительность к артефактам изображений.

*Реализация метрики взята из библиотеки Lightning.ai - TorchMetrics.*

## B.3 Kernel-Inception Distance

KID (Kernel Inception Distance) — основана на идее FID, тем не менее не использует предположение об распределении признаков изображений. Вместо этого используется приближение KDE с Гауссовским ядром признаков изображений с предобученной модели *Inception-v3*. После чего KID вычисляет максимальное среднеквадратичное отклонение

Данная метрика принята дублировать основную метрику FID, в виду ее независимости от возможно неверного предположения о гауссовском распределении признаков.

*Реализация метрики взята с библиотеки torch-fidelity*

## C Примеры генерации

### C.1 small-gan

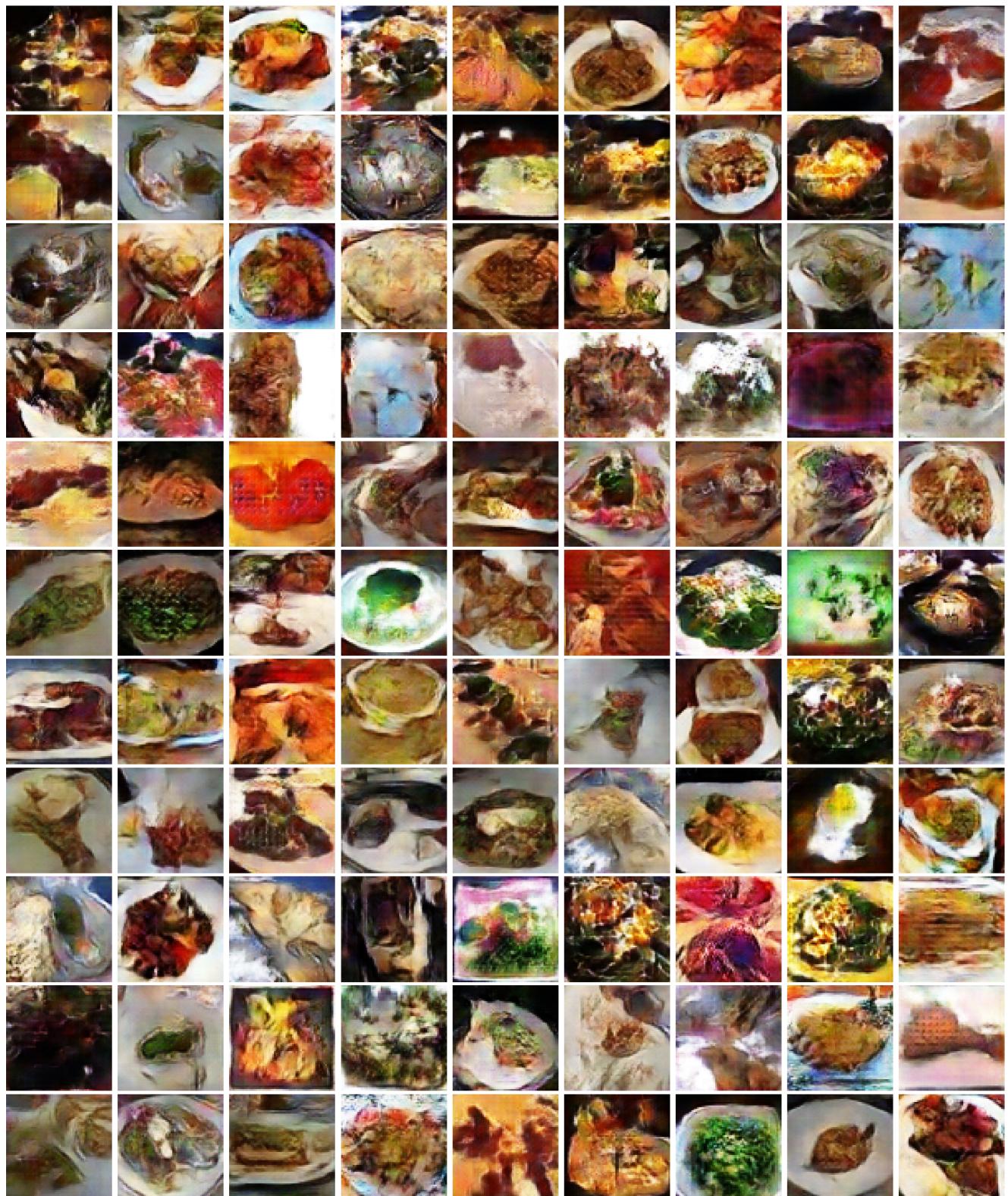


Figure 19: Пример 99 изображения сгенерированного **small-gan**

## C.2 no-noise-gan

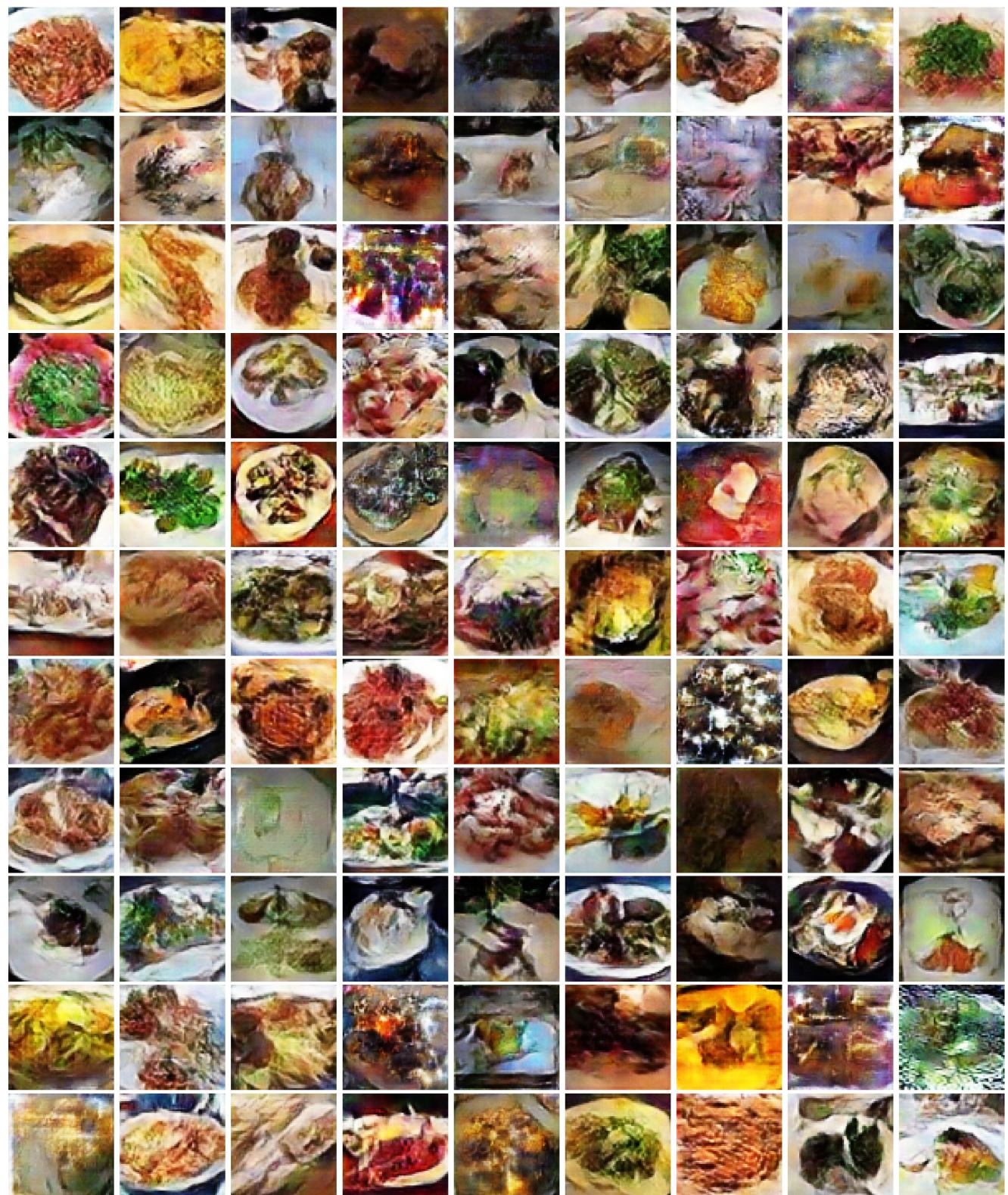


Figure 20: Пример 99 изображения сгенерированного **no-noise-gan**

### C.3 multistep-critic-gan

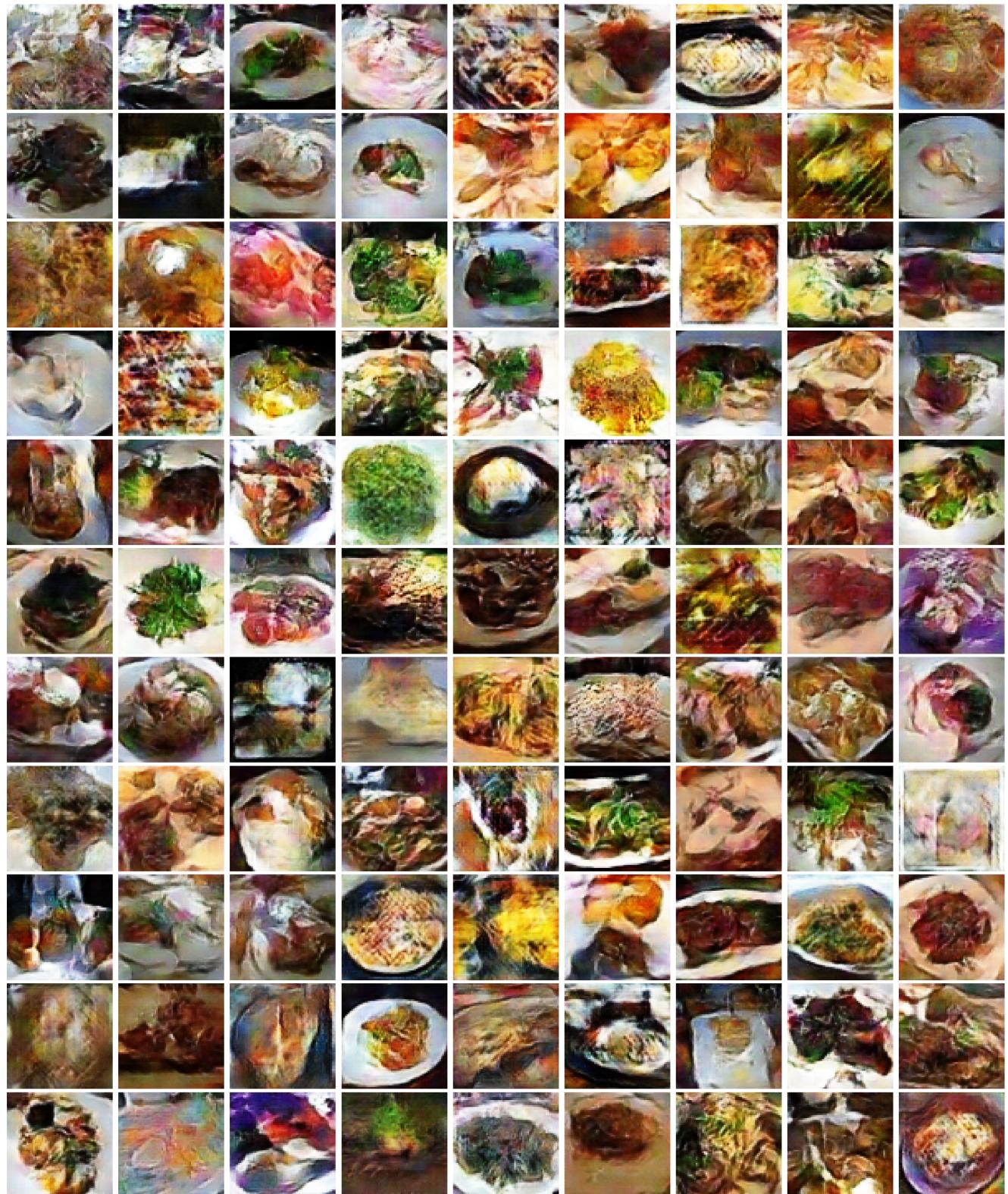


Figure 21: Пример 99 изображения сгенерированного **multistep-critic**

#### C.4 fade-in-gan

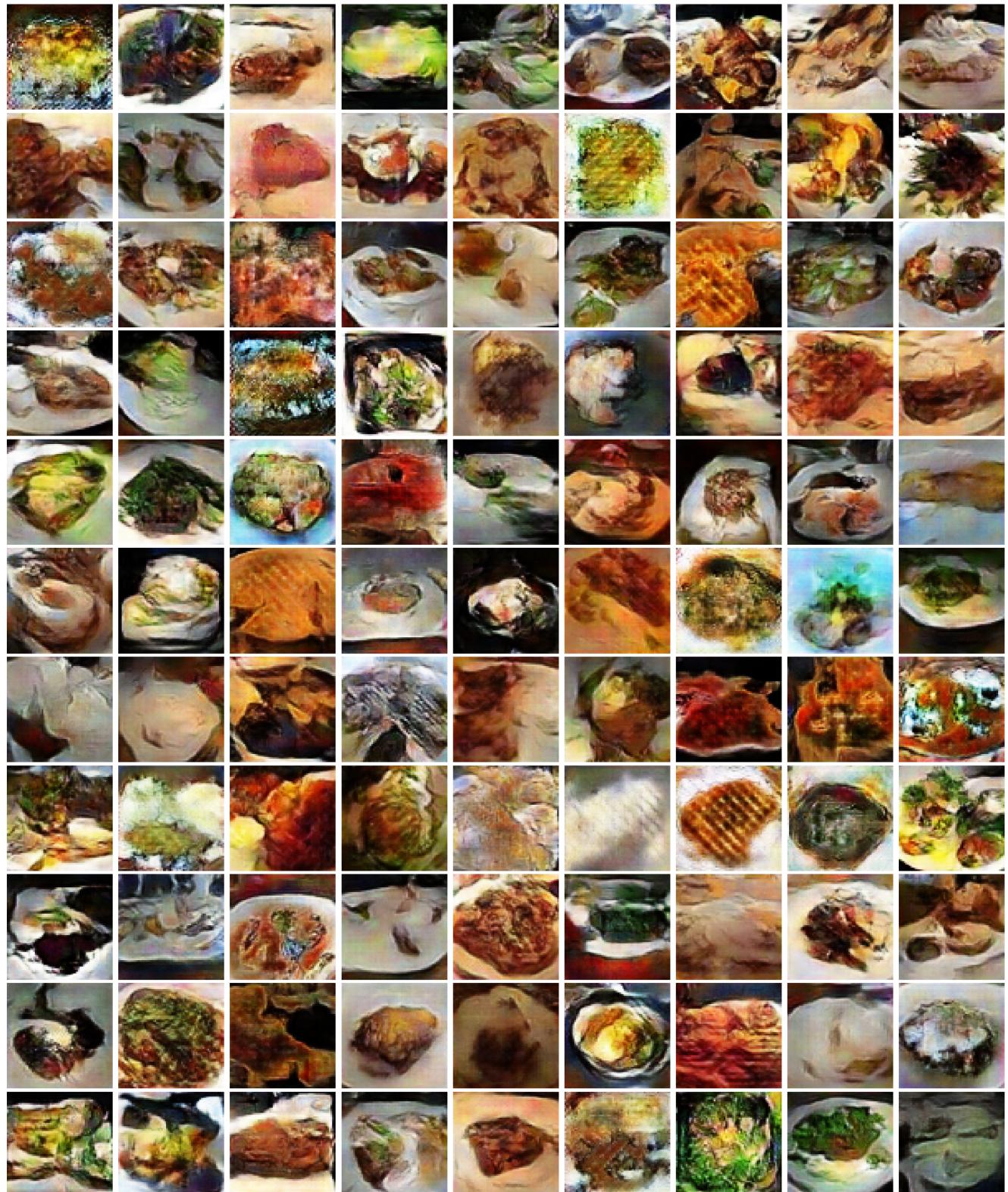


Figure 22: Пример 99 изображения сгенерированного **fade-in-gan**