https://doi.org/10.25559/SITITO.021.202501.90-102
УДК 004.932.72'1

Оригинальная статья

Методы сжатия пространства для скрытых диффузионных моделей

В. Г. Абрамов, М. А. Громов*
ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова», г. Москва, Российская Федерация
Адрес: 119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1
* 2sge@mail.ru

Аннотация
Диффузионные модели представляют собой семейство генеративных моделей, позволяющих получить наилучшее качество во многих областях, таких как генерация изображений, видео и аудио. Из-за итеративного характера работы диффузионных моделей их скорость в разы уступает другим методам генерации из-за чего кратно увеличивается стоимость и время обучения. В качестве решения этой проблемы было предложено сжать рабочее пространство диффузионной модели. Используя методы сжатия пространства удается решить основные проблемы диффузионных моделей, а также получать ранее недоступное качество генерации (например, генерация изображения с разрешением 4К).
На текущий момент многие новые работы по тематике сжатия пространства направлены на работу с видео, поскольку при генерации видео в высоком разрешении по-прежнему требуется слишком большое количество ресурсов, из-за чего ограничивается максимальная длительность сгенерированного видео.
Развитие методов сжатия пространства помогает решать многие практические задачи. В работе представлен обзор методов сжатия пространства для скрытых диффузионных моделей.

Ключевые слова: диффузионные модели, скрытые диффузионные модели, сжатие пространства, нейросетевое сжатие, генерация изображений

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Абрамов В. Г., Громов М. А. Методы сжатия пространства для скрытых диффузионных моделей // Современные информационные технологии и ИТ-образование. 2025. Т. 21, № 1. С. 90-102. https://doi.org/10.25559/SITITO.021.202501.90-102

Original article

Spatial Compression Methods for Latent Diffusion Models

V. G. Abramov, M. A. Gromov*
Lomonosov Moscow State University, Moscow, Russian Federation
Address: 1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation
* 2sge@mail.ru

Abstract

Diffusion models are a family of generative models that provide the best possible quality in many areas, such as image, video, and audio generation. Due to the iterative nature of the work of diffusion models, their speed is several times inferior to other generation methods, which increases the cost and training time many times.

As a solution to this problem, it was proposed to compress the workspace of the diffusion model. Using spatial compression methods, it is possible to solve the main problems of diffusion models, as well as to obtain previously unavailable generation quality (for example, 4K image generation).

Currently, many new works on spatial compression are aimed at working with video, since too many resources are still required when generating high-resolution videos, which limits the maximum duration of the generated video.

The development of spatial compression methods helps to solve many practical problems. The paper provides an overview of space compression methods for latent diffusion models.

Keywords: diffusion models, latent diffusion models, spatial compression, neural compression, image generation

Conflict of interests: The authors declare no conflict of interest.

For citation: Abramov V.G., Gromov M.A. Spatial Compression Methods for Latent Diffusion Models. *Modern Information Technologies and IT-Education.* 2025;21(1):90-102. https://doi.org/10.25559/SITITO.021.202501.90-102

Введение

Диффузионные модели [1-4] – это семейство генеративных нейросетей, которые доказали свое превосходство над генеративными состязательными нейросетями *(Generative Adversarial Network, GAN)* [5], которые долгое время держали лидерство в сложной задаче синтеза изображений [2-4], [6]. Помимо этого, потенциал диффузионных моделей раскрылся и в других областях: компьютерное зрение [7], обработка естественного языка [8], генерация речи [9] и др.

За последнее время было опубликовано большое число методов, нацеленных на улучшение диффузионных моделей: повышение качества генерации [10], расширение возможностей применения моделей с теоретической точки зрения [11], повышение скорости генерации [12].

Но самым важным улучшением стала идея, предложенная в [13]: сжать рабочее пространство диффузионной модели, чтобы уменьшить количество вычислений, требуемых для использования и обучения модели. От качества метода сжатия рабочего пространства зависит результирующее качество генерации.

1. Основы диффузионных моделей

Диффузионные модели – это семейство вероятностных генеративных моделей. Под генерацией будем понимать процесс превращения одного распределения в другое. При генерации изображений (например, StyleGAN [14, 15]) модели обучаются генерировать изображение $x \sim p(x)$ по входному скрытому коду $z \sim p(z)$.

Предположим, что необходимо создать модель, которая будет одно распределение $p(z)$ превращать в другое распределение $p(x)$, и наоборот. Для решения этой задачи могут использоваться две нейросети:

- Энкодер $\mathcal{E} : p(x) \to p(z)$
- Декодер $\mathcal{D} : p(z) \to p(x)$

Тогда исходные распределения будут аппроксимированы распределениями $p(x|z)$ и $q(z|x)$. Если распределение $p(z)$ будет задано заранее и, следовательно, будет известно как из него сэмплировать, то с помощью декодера можно получать примеры из $p(x|z)$, которое аппроксимирует $p(x)$. Поэтому декодер $\mathcal{D}$ можно считать генератором.

Большинство моделей генерации работают за один шаг, что делает задачу весьма сложной, поскольку от модели требуется всего за один шаг превратить одно распределение в другое.

В работе [1] вместо одношагового процесса было предложено создать цепочку преобразований, которая позволит разбить на несколько шагов сложную задачу трансформации одного распределения в другое. С этой целью определяются два Марковских процесса: прямой и обратный. В обоих процессах рассматривается последовательность переменных $\mathbf{x_0}, \mathbf{x_q}, \ldots, \mathbf{x_T}$, совместное распределение которых обозначается как $q(\mathbf{x_{0:T}}), p(\mathbf{x_{0:T}})$:

Прямой процесс:

$$\mathbf{x_0} \to \mathbf{x_T} : q(\mathbf{x_{0:T}}) = q(\mathbf{x_0}) \prod_{t=1}^{T} q(\mathbf{x_t} \mid \mathbf{x_{t-1}})$$

Обратный процесс:

$$\mathbf{x_T} \to \mathbf{x_0} : p(\mathbf{x_{0:T}}) = p(\mathbf{x_T}) \prod_{t=1}^{T} p(\mathbf{x_{t-1}} \mid \mathbf{x_t}).$$

Диффузионные модели основываются на предложенном выше подходе и генерируют выборку используя предобученный марковский процесс, который начинается с белого шума и итеративно удаляет его из выборки в течении *T* шагов. Основной принцип диффузионных моделей заключается в зашумлении информации во время прямого процесса и последующем ее восстановлении во время обратного процесса.

Р и с. 1. Иллюстрация работы диффузных моделей

F i g. 1. Illustration of how diffusion models work

Источник: здесь и далее в статье все рисунки составлены авторами.

Source: Hereinafter in this article all figures were drawn up by the authors.

В процессе обучения на изображения итеративно (в течении T шагов) добавляется шум. На последнем шаге T все изображение целиком состоит из шума. Затем обучается модель (например, глубокая нейронная сеть), которая учится обращать процесс зашумления вспять (рис. 1).

1.1 Вероятностные модели шумоподавляющей диффузии

В вероятностных моделях шумоподавляющей диффузии *(Denoising diffusion probabilistic model, DDPM)* [4] для переходов используются гауссовские распределения, поскольку гауссовская функция обладает следующим важным свойством: если и вероятность, и априорное распределение являются гауссовскими, то последующее распределение будет оставаться гауссовским. Следовательно, если каждое из приведенных выше переходных распределений является гауссовским, то совместное распределение также является гауссовским. Помимо этого свойства на практике плюсом использования гауссовского распределения является простота вычислений (поскольку гауссиан полностью характеризуется всего двумя величинами: средним значением и дисперсией).

Диффузионные модели являются моделями скрытых переменных вида:

$$p_\theta(\mathbf{x_0}) = \int p_\theta(\mathbf{x_{0:T}})\, dx_{1:T}, \tag{1}$$

где $\mathbf{x_1}, \ldots, \mathbf{x_T}$ – это скрытые переменные той же размерности, что и данные $\mathbf{x_0} \sim q(x_0)$. Совместное распределение $p_\theta(x_{0:T})$ – обратный процесс, определяемый как цепь Маркова с изученными гауссовскими переходами, начинающимися с $p(\mathbf{x_T}) = \mathcal{N}(\mathbf{x_T}; 0, \mathbf{I})$.

Для распределения данных $x_0 \sim q(x_0)$ определен прямой процесс зашумления q, который генерирует скрытые переменные от x_1 до x_T путем добавления гауссовского шума в момент времени $t \in [1, T]$ с дисперсией $\beta_t \in (0, 1)$ следующим образом:

$$q(x_1,\ldots,x_T|x_0) \& = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2}$$

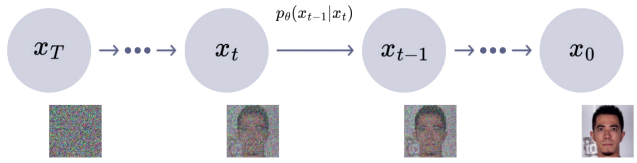
$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{3}$$

Дисперсии прямого процесса β_t являются гиперпараметрами (или могут быть получены путем повторной параметризации [16]). Если использовать достаточно большое количество шагов T и определить β_t, тогда x_T будет являться близким к изотропным гауссовским распределениям.

Если будет известно обратное распределение $p(x_{t-1}|x_t)$, можно просемплировать из $x_T \sim \mathcal{N}(0, \mathbf{I})$ и запустить процесс обратной диффузии, чтобы получить выборку из $q(x_0)$. Но так как $p(x_{t-1}|x_t)$ не известно, то необходимо обучить модель (например, нейронную сеть) с параметрами θ, позволяющую аппроксимировать $p(x_{t-1}|x_t)$:

$$p_\theta(\mathbf{x_{0:T}}) = p(\mathbf{x_T}) \prod_{t=1}^{T} p_\theta(\mathbf{x_{t-1}} \mid \mathbf{x_t}) \tag{4}$$

С помощью обученной модели, начиная с чистого шума $x_T \sim \mathcal{N}(0, \mathbf{I})$ и пройдя T шагов обратного процесса шумоподавления, можно получать примеры из обучаемого распределения (рис. 2).



Р и с. 2. Обратный процесс шумоподавления [4]

F i g. 2. Reverse process of noise reduction [4]

2. Скрытые диффузионные модели

Из-за итеративного характера диффузионных моделей *(Latent Diffusion Model, LDM)* генерация изображений требует большое количество вычислительных ресурсов, поскольку при их обучении и использовании вычисления производятся в многомерном пространстве RGB-изображений. Например, обучение диффузионной модели часто занимает тысячи часов использования графических процессоров (например, от 3000-30000 часов использования Nvidia V100 [6]). Для оценки модели после обучения необходимо сгенерировать выборку размером 50 тыс. примеров, что занимает около 120 часов на одном графическом процессоре Nvidia A100.

Для того чтобы снизить вычислительные затраты при обучении и использовании диффузионных моделей в работе [13] разделяют обучение на 2 этапа:

1. Обучение автоэнкодера (комбинация энкодера и декодера), который сжимает пространство RGB-пикселей. Полученное сжатое пространство называется скрытым.
2. Обучение диффузионной модели в скрытом пространстве автоэнкодера.

Благодаря такому подходу вычислительная сложность модели значительно снижается, поскольку процесс шумоподавления выполняется в пространстве с меньшей размерностью, чем RGB-изображения. Благодаря сохранению пространственной структуры в качестве нейросетевой модели можно использовать эффективные сверточные сети (например, UNet [17]).

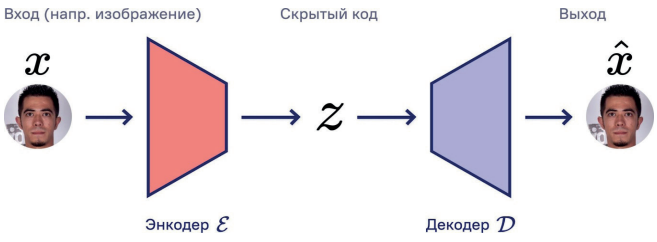
2.1 Модели для сжатия пространства

2.1.1 Нейросетевой автоэнкодер

Одним из самых первых нейросетевых методов для сжатия и генерации изображения было использование автоэнкодера *(AutoEncode, AE).* Автоэнкодер состоит из двух соединенных нейронных сетей: модели энкодера $\mathcal{E}$ и модели декодера $\mathcal{D}$. Основная цель автоэнкодера – нахождение метода кодирования в сжатое представление (скрытое пространство) и декодирования из сжатого представления. После декодирования из скрытого пространства восстановленные данные должны быть такими же, как до декодирования:

$$z = \mathcal{E}(x)$$
$$\hat{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x))$$

В случае, когда данные представляют собой изображения, энкодер $\mathcal{E}$ переводит входное изображение x в меньшее по объему представление z (латентный код). Декодер $\mathcal{D}$ можно считать генеративной моделью, способной из кода z генерировать восстановленное изображение $\hat{x}$ (рис. 3).



Р и с. 3. Иллюстрация работы автоэнкодера на примере человеческого лица
F i g. 3. Illustration of the operation of an autoencoder using the example of a human face

Энкодер и декодер обучаются вместе на заданном множестве данных. Функция потерь (например, среднеквадратичное отклонение (MSE)) штрафует нейросеть за создание выходных данных, отличающихся от входных. Таким образом, энкодер $\mathcal{E}$ обучается сохранять как можно больше полезной информации в скрытом пространстве и разумно отбрасывать неважную информацию – например, шум. Декодер $\mathcal{D}$ обучается превращать сжатую информацию в скрытом пространстве обратно в исходные данные.

Автоэнкодеры обычно применяются для сокращения размерности и удаления шумов. Можно случайно семплировать код $z \in Z$ и использовать декодер $\mathcal{D}$ для генерации нового изображения.

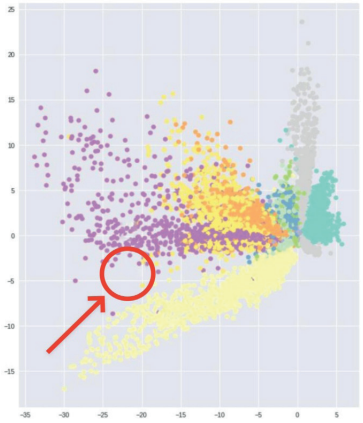
Основная проблема такого подхода – это неопределенность распределения кодов Z после обучения автоэнкодера. Самый простой способ решить эту проблему – добавить ограничение на скрытое пространство (например, добавить штраф за выход значений компонент вектора z из диапазона $[-1; 1]$). Тогда для генерации картинки можно семплировать латентный код $z \in Z$ из нормального распределения:

$$z \sim \mathcal{N}(0; \mathbf{I}), \quad (8)$$

и генерировать изображение, используя декодер:

$$\hat{x} = \mathcal{D}(z) \quad (9)$$

Как правило, скрытое пространство Z, создаваемое энкодером, сильно разрежено (рис. 4). Значения скрытых кодов группируются в отдельные области, что создает проблемы для генерации новых изображений. Найти скрытое значения z, для которого декодер знает, как восстановить изображение, почти невозможно.



Р и с. 4. Визуализация скрытого пространства на примере набора данных MNIST [18]
F i g. 4. Visualization of Latent Space Using the MNIST Dataset as an Example [18]

2.1.2 Вариационный автоэнкодер

Вариационный автоэнкодер *(Variational Autoencoder, VAE)* [16], [19], [20] решает проблемы классических автоэнкодеров, заставляя латентные коды z соответствовать нормальному распределению – таким образом VAE получает контроль над латентным пространством.

Вариационный автоэнкодер использует распределения вероятностей для описания $\mathbf{X}$ и $\mathbf{Z}$. Определим эти распределения:$\mathbf{X}$ и $\mathbf{Z}$. Определим эти распределения:

1. $p(\mathbf{X})$ – исходное распределение данных $\mathbf{X}$.
2. $p(\mathbf{Z})$ – распределение латентного кода $\mathbf{Z}$.
3. $p(\mathbf{Z}|\mathbf{X})$ – условное распределение, которое сообщает нам вероятность $\mathbf{Z}$ при заданном $\mathbf{Z}$ при заданном $\mathbf{Z}$ при заданном $\mathbf{X}$.
4. $p(\mathbf{X}|\mathbf{Z})$ – условное распределение, которое сообщает нам апостериорную вероятность получения $\mathbf{X}$ при заданном $\mathbf{Z}$.

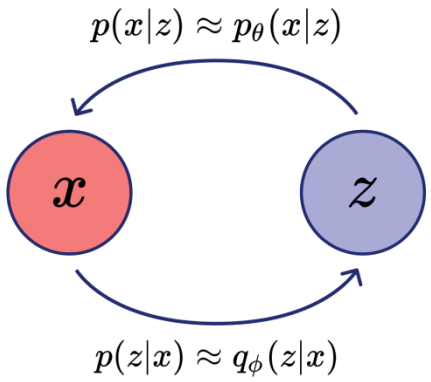
Вместо того, чтобы использовать детерминированную процедуру преобразования $\mathbf{X}$ в $\mathbf{Z}$ (как в классическом автоэнкодере), VAE обучается таким образом, что распределение $p(\mathbf{X})$ может быть сопоставлено с желаемым распределением $p(\mathbf{Z})$ и может быть возвращено к $p(\mathbf{X})$.

Распределение $p(\mathbf{X})$ нам не известно. Вместо него доступна выборка из этого распределения (например, набор изображений). Распределение $p(\mathbf{Z})$ может быть любым (так как это распределение создано нами искусственно), но удобнее сделать его нормальным.

Из-за того, что между $p(\mathbf{X})$ и $p(\mathbf{Z})$ не существует никакой связи, то неизвестны и условные распределения $p(\mathbf{Z}|\mathbf{X})$, $p(\mathbf{X}|\mathbf{Z})$. Вместо них в VAE рассматривают гауссовские апроксимирущие распределения:

1. $q_\phi(\mathbf{Z}|\mathbf{X})$ для $p(\mathbf{Z}|\mathbf{X})$.
2. $p_\theta(\mathbf{X}|\mathbf{Z})$ для $p(\mathbf{X}|\mathbf{Z})$.

С помощью ппроксимирующих распределений устанавливается связь между $\mathbf{X}$ и $\mathbf{Z}$ (рис. 5).



Р и с. 5. Иллюстрация связи между *x* и *z*
F i g. 5. Illustration of the relationship between *x* and *z*

ϕ и θ – параметры модели, которые нужно оптимизировать на основе обучающих выборок. Для этого необходимо использовать функцию потерь в терминах ϕ и θ. Так функцию потерь называется нижней границей доказательства (ELBO) [19]:

$$\mathrm{ELBO}(\mathbf{x}) = E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \quad (10)$$

ELBO – это нижняя граница априорного распределения $\log p(\mathbf{x})$:

$$\log p(\mathbf{x}) = E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] + \mathbb{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}|\mathbf{x}))$$
$$\geq E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] = \mathrm{ELBO}(\mathbf{x}) \quad (11)$$

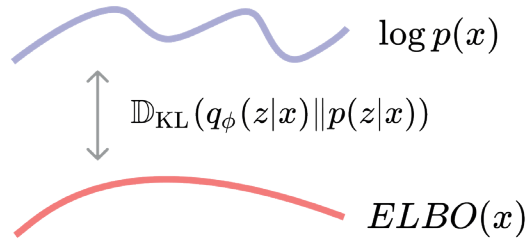
Учитывая, что KL-дивергенция всегда неотрицательна, ELBO является допустимой нижней границей для $\log p(\mathbf{x})$. Следовательно, для максимизации априорного распределения $\log p(\mathbf{x})$ можно максимизировать ELBO. В случае, если аппроксимирующее распределение $q_\phi(\mathbf{z}|\mathbf{x})$ сможет точно соответствовать истинному распределению $p(\mathbf{z}|\mathbf{x})$, то KL-дивергенция из уравнения (11) станет равна 0 и $\log p(\mathbf{x})$ в точности будет равна ELBO (рис. 6).

Поскольку $p(\mathbf{x},\mathbf{z})$ не известно, необходимо разложить ELBO на известные части:

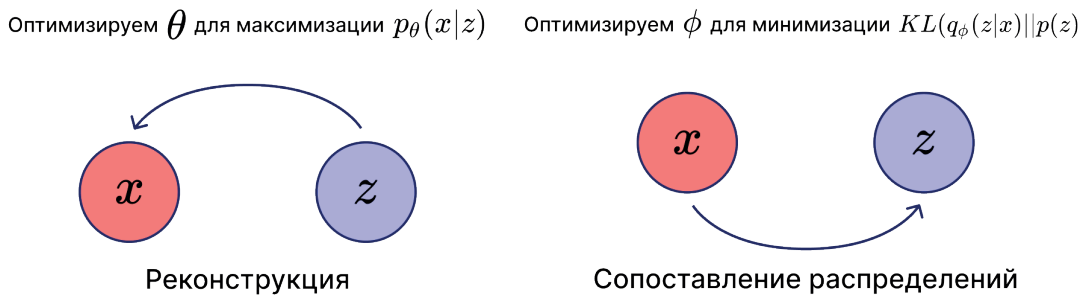
$$\mathrm{ELBO}(\mathbf{x})$$
$$= E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$
$$= E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$
$$= E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] + E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$
$$p(\mathbf{x}|\mathbf{z}) \to p_\theta(\mathbf{x}|\mathbf{z})$$
$$= E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathbb{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

Рассмотрим два получившихся слагаемых.

Оптимизируем $\mathbb{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ таким образом, чтобы энкодер $\mathcal{E}$ превращал $\mathbf{x}$ в скрытый вектор $\mathbf{z}$ таким образом, чтобы скрытый вектор следовал нормальному распределению $N(0, I)$. Оптимизируя $E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$ максимизируется $\log p_\theta(\mathbf{x}|\mathbf{z})$ так, чтобы максимизировать вероятность наблюдения изображения из исходного распределения.



Р и с. 6. Разница между log *p(x)* и ELBO определяется *KL*-дивергенцией
F i g. 6. The difference between log *p(x)* and ELBO is determined by the *KL* divergence



Р и с. 7. Иллюстрация двух задач оптимизации во время обучения
F i g. 7. Illustration of two optimization problems during training

На рисунке (рис. 7) проиллюстрированы две задачи оптимизации при обучения вариационного автоэнкодера. Во время обучения предполагается, что доступны $\mathbf{z}$ и $\mathbf{x}$, где $\mathbf{z}$ необходимо извлечь из $q_\phi(\mathbf{z}|\mathbf{x})$. Для сопоставления распределений ищем такие параметры энкодера ϕ, чтобы минимизировать *KL*-дивергенцию между распределениями. Стоит заметить, что при каждом обновлении параметров ϕ распределение $q_\phi(\mathbf{z}|\mathbf{x})$ меняется, что усложняет обучение модели. Для реконструкции $\mathbf{x}$ из $\mathbf{z}$ оптимизируются параметры декодера, чтобы максимизировать $p_\theta(\mathbf{x}|\mathbf{z})$.

2.1.3 Обучение вариационного автоэнкодера на практике

Для обучения нейросети необходим набор данных из пар $(\mathbf{x},\mathbf{z})$. $\mathbf{x}$ – это изображения из обучающего набора данных. $\mathbf{z}$ получается из распределения $q_\phi(\mathbf{z}|\mathbf{x})$, которое является гауссовой функцией.

Пусть μ и $\sigma^2 I$ – это среднее значение и ковариационная матрица для $q_\phi(\mathbf{z}|\mathbf{x})$. С помощью нейронной сети (энкодера $\mathcal{E}$) для каждого изображения $\mathbf{x_i}$ предсказывается μ_i и σ_i^2:

$$\mu_i, \sigma_i^2 = \mathcal{E}_\phi(\mathbf{x_i}) \quad (12)$$

После чего выборку $\mathbf{z_i}$ можно сэмплировать из распределения $N(\mathbf{z} \mid \mu_i, \sigma_i^2 I)$.

Работа энкодера $\mathcal{E}$ проиллюстрирована на рисунке (рис. 8). Задача декодера $\mathcal{D}$- из скрытого кода $\mathbf{z_i}$ сгенерировать изображение $\widehat{\mathbf{x}}$ (рис. 9):

$$\widehat{\mathbf{x}}_\mathbf{i} = \mathcal{D}_\theta(\mathbf{z_i}), \quad (13)$$

где θ – это параметры декодера.

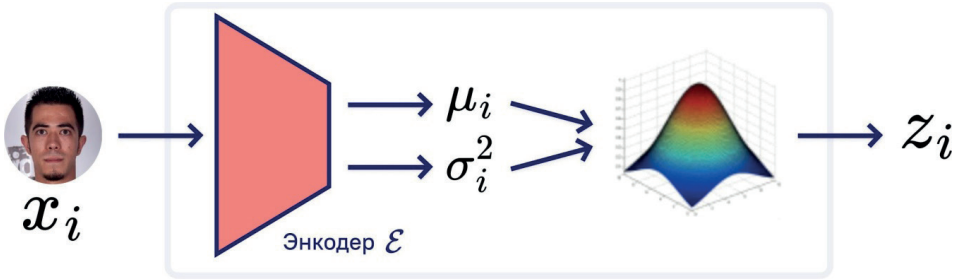
Предположим, что распределение ошибки реконструкции между $\mathbf{x}$ и $\widehat{\mathbf{x}}$ является гауссовским:

$$(\widehat{\mathbf{x}} - \mathbf{x}) \sim N(0, \sigma_{\mathrm{err}}^2)$$

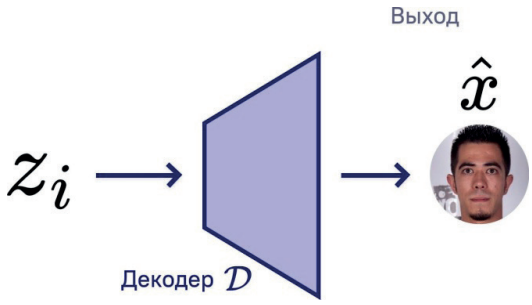
Из этого следует, что распределение $p_\theta(\mathbf{x}|\mathbf{z})$ равно $\log N(\mathbf{x} \mid \mathcal{D}_\theta(\mathbf{z}), \sigma_{\mathrm{err}}^2 I)$, что по определению равно:

$$\log \frac{1}{\sqrt{(2\pi\sigma_{\mathrm{err}}^2)^{dim}}} \exp\left\{-\frac{\|\mathbf{x} - \mathcal{D}_\theta(\mathbf{z})\|^2}{2\sigma_{\mathrm{err}}^2}\right\} = -\frac{\|\mathbf{x} - \mathcal{D}_\theta(\mathbf{z})\|^2}{2\sigma_{\mathrm{err}}^2} - \underbrace{\log \sqrt{(2\pi\sigma_{\mathrm{err}}^2)^{dim}}}, \quad (14)$$

где *dim* – размерность пространства *X*.



Р и с. 8. Иллюстрация работы VAE энкодера
Р и с. 8. Illustration of the VAE encoder in operation



Р и с. 9. Иллюстрация работы VAE декодера
F i g. 9. Illustration of the VAE decoder in operation

Уравнение (14) показывает, что для оптимизации декодера $\mathcal{D}$ достаточно в качестве функции ошибки использовать среднеквадратичное отклонение между декодированным изображением $\widehat{\mathbf{x}}$ и исходным изображением $\mathbf{x}$.

Итак, результирующая функция ошибки:

$$\text{argmax}_{\phi,\theta}\left\{E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\right\} \quad (15)$$

Так как обучающий набор данных конечен, вместо математического ожидания

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] \quad (16)$$

для обучения энкодера используется среднее по набору данных:

$$\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(\mathbf{x_i}|\mathbf{z_i}) \quad (17)$$

Используя аналитическое решение для KL-дивергенции получаем:

$$\mu_\phi(\mathbf{x_i}), \sigma_\phi(\mathbf{x_i})^2 = \mathcal{E}_\phi(\mathbf{x_i}) \quad (18)$$

$$\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x_i}) \parallel p(\mathbf{z})) = \frac{1}{2}\left((\sigma_\phi^2(\mathbf{x_i}))^{dim} + \mu_\phi(\mathbf{x_i})^T\mu_\phi(\mathbf{x_i}) - dim * \log(\sigma_\phi^2(\mathbf{x_i}))\right), \quad (19)$$

где *dim* – размерность скрытого пространства $\mathbf{z}$.

Аналитическое решение (19) дифференцируемо, следовательно с помощью него можно обучать декодер используя классический для нейросетей метод – обратное распространение градиента.

2.1.4 Дискретное скрытое пространство

Следующим этапом развития автоэнкодеров стал переход от непрерывного скрытого пространства к дискретному, которое более естественно подходит для многих модальностей. В работе [21] *(Vector Quantized-Variational AutoEncoder, VQ-VAE)* представлено новое семейство генеративных моделей, успешно сочетающих структуру вариационного автоэнкодера с дискретным скрытым представлением. Оно основано на новой параметризации апостериорного распределения, а также на новом способе обучения, основанным на векторном квантовании *(Vector quantization, VQ)*.

Вместо использования непрерывных скрытых кодов вводится скрытое векторное пространство $e \in R^{K\times D}$, где K – размер дискретного скрытого пространства, а D – размерность каждого скрытого вектора e_i. Иначе говоря, вводится скрытое пространство, в котором существует K скрытых векторов $e_i \in R^D, i \in 1, 2, ..., K$.

На рисунке (рис. 10) показана схема работы VQ-VAE.

Модель VQ-VAE аналогично VAE имеет энкодер $\mathcal{E}$ и декодер $\mathcal{D}$, но при этом добавляется обучаемая таблица векторов (часто называемая термином «кодовая книга»). Принимая на вход x энкодер $\mathcal{E}$ выдает $z_e(x)$ (матрица непрерывных векторов). Далее для каждого вектора из $z_e(x)$ находится ближайший вектор из таблицы векторов e:

Матрица индексов M ближайших векторов и будет являться дискретным скрытым пространством, так как каждый элемент матрицы может принимать только дискретные значения от 1 до K.

Для того, чтобы получить изображение из скрытого пространства, матрица индексов M преобразуется в тензор путем замены каждого элемента k на соответствующий ему вектор e_k. Далее декодер $\mathcal{D}$ на основе тензора E реконструирует изображение $\widehat{x}$.

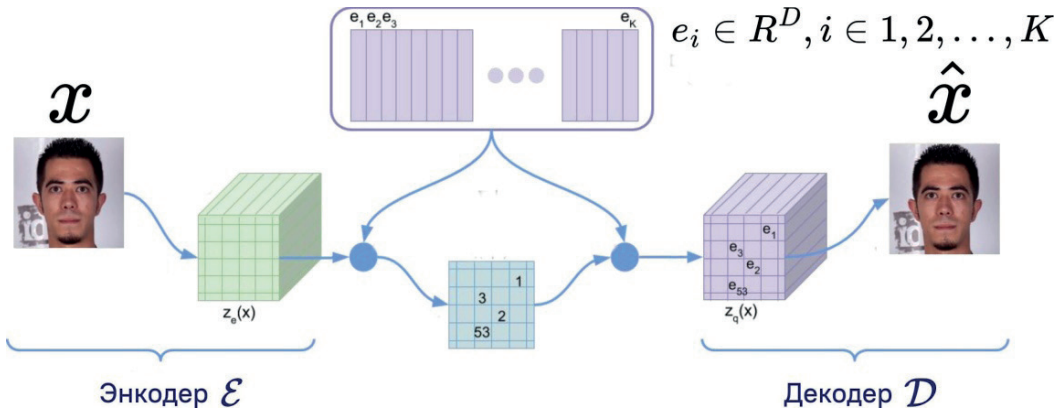
При использовании дискретного скрытого пространства, апостериорное и априорное распределения являются категориальными. Вероятности апостериорного категориального распределения $q(z|x)$ однозначно определяются следующим образом:

$$q(z=k|x) = \begin{cases} 1 & \text{если k} = \text{argmin}_j\|z_e(x) - e_j\|_2, \\ 0 & \text{иначе} \end{cases}, \quad (21)$$

$q(z=k|x)$ является детерминированным, а распределение над z является равномерным из-за чего KL-дивергенция становится равной $\log K$.

Для VQ-VAE применяются три функции ошибки, каждая из которых используются для оптимизации различных частей модели.

$$z_q(x) = e_{k,\text{ где}}\, k = \text{argmin}_j\|z_e(x) - e_j\|_2 \quad (20)$$



Р и с. 10. Иллюстрация работы VQ-VAE [21]
F i g. 10. Illustration of VQ-VAE operation [21]

Первая функция ошибки определяет потери при восстановлении изображений. Она оптимизирует энкодер и декодер, уменьшая разницу между $\mathbf{x}$ и $\widehat{\mathbf{x}}$:

$$\|x - \widehat{x}\|_2^2 \quad (22)$$

Поскольку при обучении для получения ближайшего вектора (20) не определен градиент, он просто копируется из входных данных декодера $z_q(x)$ в выходные данные энкодера $z_e(x)$. Это возможно, поскольку выходное представление энкодера и входные данные декодера имеют одинаковую размерность пространства. А градиенты содержат полезную информацию о том, как энкодер должен изменять свои выходные данные, чтобы снизить потери при восстановлении изображения.

Через функцию поиска ближайшего вектора невозможно провести градиент, поскольку она не дифференцируема. Из-за невозможности провести градиент через операцию поиска ближайшего вектора нельзя оптимизировать таблицу векторов. Для решения этой проблемы, используется один из простейших алгоритмов – векторное квантование (VQ). Алгоритм векторного квантования использует среднеквадратичную ошибку L_2 для перемещения векторов e к выходам энкодера $z_e(x)$:

$$\|\mathrm{sg}[z_e(x)] - e\|_2^2, \quad (23)$$

где *sg* обозначает операцию остановки градиента, который определяется как тождественный слой (выход слоя равен входу) во время прямого вычисления и имеет нулевые частные производные.

Поскольку скорость обновления энкодера и таблицы векторов различная, энкодер может предсказывать $z_e(x)$, которые будут далеки от векторов $\mathcal{C}$. Чтобы этого избежать используется третья функция потерь, которая помогает энкодеру предсказывать $z_e(x)$ близкие к существующим векторам e из кодовой книги:

$$\|z_e(x) - \mathrm{sg}[e]\|_2^2 \quad (24)$$

Таким образом финальная функция потерь является суммой трех вышеперечисленных функций потерь:

$$L_V Q = \|x - \widehat{x}\|_2^2 + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \mathrm{sg}[e]\|_2^2, \quad (25)$$

где β – коэффициент, используемый для определения важности третьей функции потерь.

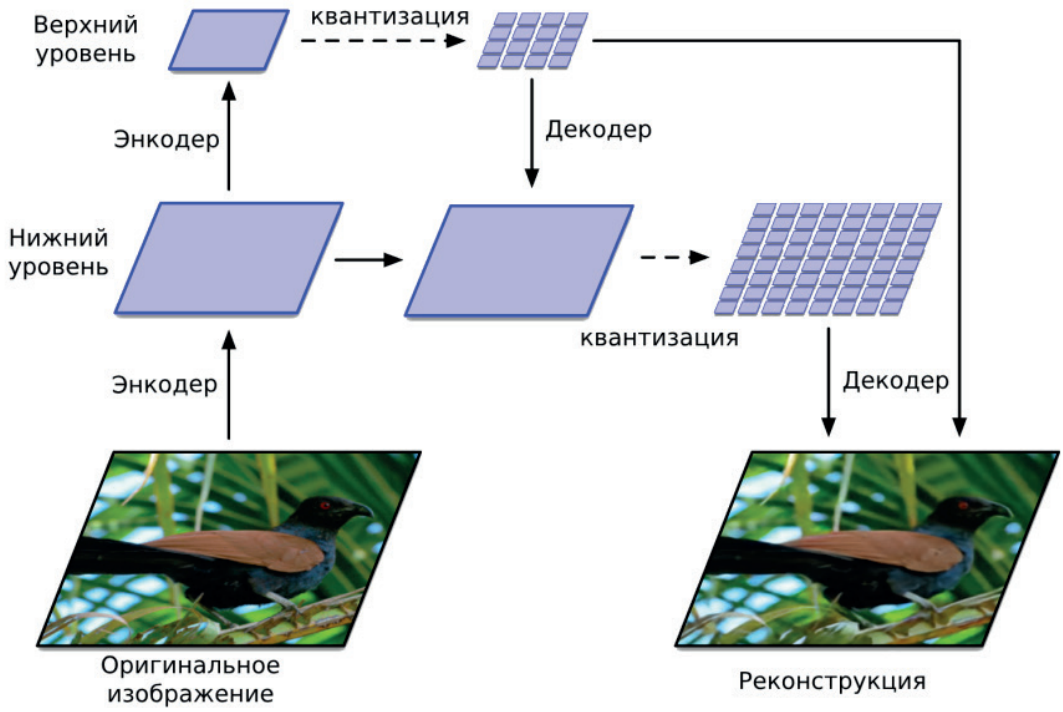
Вскоре после публикации работы VQ-VAE [21] было предложено улучшение (VQ-VAE-2 [22]) этого метода, которое использует иерархический способ сжатия изображения, вводя несколько скрытых пространств на разной глубине нейросети (рис. 11).

Одной из главных проблем для использования VQ-VAE на практике было низкое качество восстановления изображения с точки зрения человеческого взгляда. Восстанавливаемые изображения имели некоторую размытость, характерную при использовании L_2 функции потерь. Для того чтобы улучшить визуальное качество восстановления необходимо добавить функцию потерь, коррелирующую с человеческим восприятием, например, LPIPS [23]. Авторы VQGAN [24] предлагают вместо LPIPS использовать состязающиеся сети, которые также хорошо показывает себя в улучшении восприятия изображения. В 2022 году была предложена модификация модели VQ-VAE-2 под названием MoVQGAN [25]. Основным улучшением является использование пространственно обусловленных нормализаций (или *spatially conditional normalization*) в блоках декодера, которые позволяют повысить реалистичность восстанавливаемых изображений. Пространственно обусловленная нормализация рассчитывается по формуле:

$$F^i = \phi_\gamma(z_q)\frac{F^{i-1} - \mu(F^{i-1})}{\sigma(F^{i-1})} + \phi_\beta(z_q),$$

где F^{i-1} – промежуточная карта признаков, $\mu(F^{i-1}), \sigma(F^{i-1})$ – функции расчёта среднего и стандартного отклонения, а $\phi_\gamma(z_q)$, $\phi_\beta(z_q)$ – обучаемые аффинные преобразования.

Добавление слоя нормализации помогает избежать возникновения артефактов, которые появляются из-за процесса преобразования близких друг к другу векторов энкодера в одинаковые индексы таблицы векторов. Также предлагается использовать многоканальное представление латентного пространства, разделяя его на несколько частей по каналам и кодируя каждую часть по отдельности. Эти нововведения позволяют сократить размер таблицы векторов с 16384 до 1024. MoVQGAN значительно превзошел описанные ранее подходы в задаче восстановления изображений по метрикам FID, SSIM, PSNR и LPIPS.



Р и с. 11. Иллюстрация работы VQ-VAE-2 [22]
F i g. 11. Illustration of VQ-VAE-2 operation [22]

Вышеописанные методы хорошо показывают себя при умеренной степени пространственного сжатия изображений (8x), но не могут достичь хорошей точности восстановления изображений при высоких степенях пространственного сжатия (например, 64x). Для увеличения возможности пространственного сжатия изображений, в 2024 году был представлен автоэнкодер глубокой компрессии *(Deep Compression Autoencoder, DC-AE)* [26]. Предлагается два ключевых новшества:

- Для улучшения качества в автоэнкодере добавляются остаточные соединения – функции преобразования пространства в канал, для облегчения задачи оптимизации автоэнкодера:

$$H \times W \times C \xrightarrow{\text{пространство-в-канал}} \frac{H}{2} \times \frac{W}{2} \times 4C$$
$$\xrightarrow{\text{деление на 2 группы}} \underbrace{[\frac{H}{2} \times \frac{W}{2} \times 2C, \frac{H}{2} \times \frac{W}{2} \times 2C] \xrightarrow{\text{усреднение}} \frac{H}{2} \times \frac{W}{2} \times 2C.}_{\text{усреднение по каналам}}$$

- Трехэтапный алгоритм обучения для снижения рисков обобщения, присущим автоэнкодерам с высоким уровнем пространственного сжатия.

Благодаря этим разработкам коэффициент пространственного сжатия автоэнкодера был увеличен до 128, а качество восстановления при этом сохранилось (рис. 12).

Р и с. 12. Сравнение качества восстановления VAE и DC-AE при коэффициенте пространственного сжатия = 12 [26]
F i g. 12. Comparison of VAE and DC-AE reconstruction quality at spatial compression ratio = 12 [26]

2.2 Диффузные модели в скрытом пространстве

Впервые использование автоэнкодера для сжатия рабочего пространства было предложено в работе LDM [13]. Для улучшения визуального качества реконструкции при его обучения, помимо функций потерь от классического VAE, применяются следующие функции потерь:

1. Потеря восприятия (LPIPS [23])
2. Состязательная потеря на основе патчей (PatchGAN [27])

Для изображения $x \in \mathbb{R}^{H \times W \times 3l}$ в пространстве RGB, $\mathcal{E}$ кодирует x в скрытое представление $z = \mathcal{E}(x)$, а $\mathcal{D}$ восстанавливает изображение из скрытого, $\tilde{x} = \mathcal{D}r(z) = \mathcal{D}(\mathcal{E}(x))$, где $z \in \mathbb{R}^{\hat{X} w \times c}$.

Важно отметить, что $\mathcal{E}$ уменьшает разрешение пространства изображений в f раз, где $f = H/h = W/w$. При этом, из-за использования сверточной архитектуры, f является степенью двойки.

Для уменьшения дисперсии в скрытом пространстве авторы [13] экспериментируют с двумя видами регуляризации:

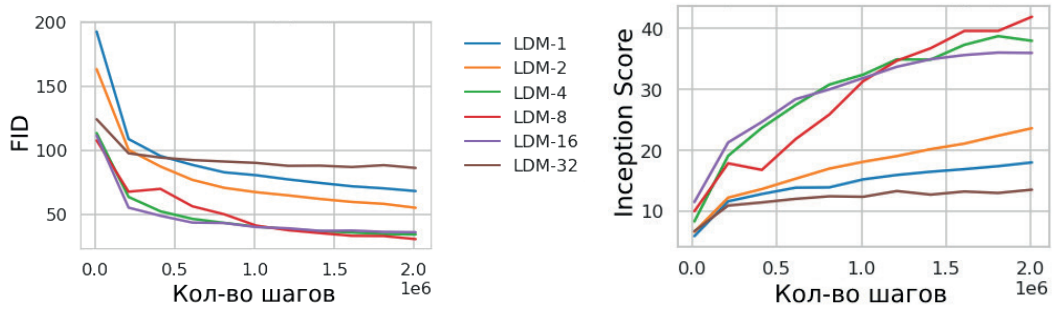
1. *KL-регуляризация* (аналогично VAE [16])
2. *VQ-регуляризация* (векторное квантование [21])

Не смотря на то, что используя *KL-регуляризацию* удается достичь лучшего качества реконструкции изображений, в ходе экспериментов [13] было обнаружено, что диффузионные модели, обученные в VQ-регуляризованных скрытых пространствах, достигают лучшего качества выборки.

С помощью предобученных моделей сжатия появилась возможность использовать эффективное скрытое пространство изображений, в котором абстрагируются высокочастотные детали изображения. Размерность скрытого пространства в f раз меньше, чем размерность пиксельного (RGB) пространства, что дает возможность обучать диффузионные модели намного быстрее, поскольку теперь модели будут фокусироваться на важных, семантических значимых деталях изображения. Выбор гиперпараметра f (коэффициент сжатия пространства) влияет на размер скрытого пространства, а значит и на эффективность обучения и качество восстановления изображения. Авторы работы [13] экспериментируют с различными значениями f (1,2,4,8,16,32).

Как видно из графиков обучения при различных f (рис. 13) модели показывают разную эффективность обучения при одинаковом количестве шагов обучения. При $f = 1$ (пространство без сжатия) модель требует значительно большего времени обучения по сравнению с моделями с большими коэффициентами f. Слишком сильное сжатие ($f = 32$) не позволяет модели достичь высокого качества генерации.

Для достижения максимального качества оптимальным выбором будет $f = 4$ или $f = 8$. Так, например, большинство лучших моделей безусловной генерации используют $f = 4$, а самая первая популярная модель для генерации изображения по тексту Stable Diffusion использует $f = 8$.



Р и с. 13. Сравнение скорости обучения при различном выборе коэффициента сжатия пространства f [13]

F i g. 13. Comparison of learning speed for different choices of space compression coefficient f [13]

2.3 Применение моделей сжатия пространства в авторегрессионых моделях генерации

Диффузионные модели демонстрируют потрясающее качество генерации, в особенности благодаря использованию методов сжатия пространства. В то же время, не смотря на попытки [28-30] применить диффузию в работе с естественным языком, в генерации текста продолжают лидировать большие языковые модели *(Large Language Model, LLM)*. Опираясь на успех больших языковых моделей, авторы LlamaGen [31] предложили применить LLM для генерации изображений.

При генерации текста LLM модели предсказывают токен за токеном в авторегрессионом режиме. При генерации следующего токена для каждого токена из словаря модель предсказывает вероятность его появления. Это возможно благодаря дискретной природе текстов. Несмотря на то, что теоретически каждому возможному пикселю можно сопоставить свой токен (но их количество в случае RGB пикселя будет больше 16 млн. токенов), изображения легче представлять как непрерывное распределение, которое не подходит для дискретной генерации.

Чтобы решить эту проблему авторы LlamaGen [31] предлагают свой метод сжатия пространства создающий дискретной скрытое пространство (аналогично VQ-VAE, VQ-GAN, MoVQGAN), благодаря которому решается проблема непрерывности. Также, при использовании модели для сжатия, уменьшается разрешение рабочего пространства, что уменьшает необходимое количество токенов, которое нужно предсказать нейросети. Авторы обучают несколько моделей различного размера: от 111 миллионов параметров до 3 миллиардов (3B). Самая большая модель LlamaGen-3B показала лучшее качество FID на датасете Imagenet (Таблица 1) [32], обходя даже современные диффузионные модели на основе трансформера (например, DiT [10]).

Таблица 1. **Сравнение моделей LDM, DiT, LlamaGen**
Table 1. **Comparison of LDM, DiT, LlamaGen models**

Модель	Размер	FID↓	IS↑
LDM-4 [13]	400M	3.60	247.7
DiT-XL [10]	675M	2.27	278.2
LlamaGen-3B [31]	3.1B	2.18	263.33

LlamaGen [31] показывает, что методы сжатия пространства используются не только в диффузионных, но и в других методах.

References

[1] Sohl-Dickstein J., et al. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In: Bach F., Blei D. (eds.) Proceedings of the 32nd International Conference on Machine Learning (PMLR). Vol. 37. Lille, France; 2015. p. 2256-2265. Available at: https://proceedings.mlr.press/v37/sohl-dickstein15.html (accessed 29.01.2025).

[2] Song Y., Ermon S. Generative modeling by estimating gradients of the data distribution. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS '19). Article number: 1067. Red Hook, NY, USA: Curran Associates Inc.; 2019. p. 11918-11930.

[3] Song Y., et al. Score-Based Generative Modeling through Stochastic Differential Equations. In: International Conference of Learning Representations (ICLR 2021). Austria; 2021. 36 p. Available at: https://openreview.net/forum?id=PxTIG12RRHS (accessed 29.01.2025).

[4] Ho J., Jain A., Abbeel P. Denoising Diffusion Probabilistic Models. In: Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Article number: 574. Red Hook, NY, USA: Curran Associates Inc.; 2020. p. 6840-6851. Available at: https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html (accessed 29.01.2025).

[5] Goodfellow I. et al. Generative Adversarial Networks. *Communications of the ACM*. 2020;63(11):139-144. https://doi.org/10.1145/3422622

[6] Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis. In: Ranzato M., et al. (eds.) Advances in Neural Information Processing Systems. Vol. 34. Curran Associates, Inc.; 2021. p. 8780-879. Available at: https://papers.nips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html (accessed 29.01.2025).

[7] Croitoru F.-A., Hondru V., Ionescu R.T., Shah M. Diffusion Models in Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2023;45(9):10850-10869. https://doi.org/10.1109/TPAMI.2023.3261988

[8] Yi Q., Chen X., Zhang C., Zhou Z., Zhu L., Kong X. Diffusion models in text generation: a survey. *PeerJ Computer Science*. 2024;10:e1905. https://doi.org/10.7717/peerj-cs.1905

[9] Kameoka H., Kaneko T., Tanaka K. VoiceGrad: Non-Parallel Any-to-Many Voice Conversion With Annealed Langevin Dynamics. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 2024;32:2213-2226. https://doi.org/10.1109/TASLP.2024.3379901

[10] Peebles W., Xie S. Scalable Diffusion Models with Transformers. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). Paris, France: IEEE Press; 2023. p. 4172-4182. https://doi.org/10.1109/ICCV51070.2023.00387

[11] Daras G., et al. Ambient Diffusion: Learning Clean Distributions from Corrupted Data. *arXiv:*2305.19256. 2023. https://doi.org/10.48550/arXiv.2305.19256

[12] Sauer A., et al. Fast High-Resolution Image Synthesis with Latent Adversarial Diffusion Distillation. *arXiv:*2403.12015. 2024. https://doi.org/10.48550/arXiv.2403.12015

[13] Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, LA, USA: IEEE Press; 2022. p. 10674-10685. https://doi.org/10.1109/CVPR52688.2022.01042

[14] Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2021;43(12):4217-4228. https://doi.org/10.1109/TPAMI.2020.2970919

[15] Karras T., Laine S., Aittala M., Hellsten J., Lehtinen J., Aila T. Analyzing and Improving the Image Quality of StyleGAN. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA: IEEE Press; 2020. p. 8107-8116. https://doi.org/10.1109/CVPR42600.2020.00813

[16] Kingma D.P., Welling M. Auto-Encoding Variational Bayes. *arXiv:*1312.6114. 2013. https://doi.org/10.48550/arXiv.1312.6114

[17] Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. *Lecture Notes in Computer Science*. Vol. 9351. Cham: Springer; 2015. p. 234-241. https://doi.org/10.1007/978-3-319-24574-4_28

[18] Lecun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE.* 1998;86(11):2278-2324. https://doi.org/10.1109/5.726791

[19] Kingma D.P., Welling M. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning.* 2019;12(4):307-392. http://dx.doi.org/10.1561/2200000056

[20] Pu Y., et al. Variational Autoencoder for Deep Learning of Images, Labels and Captions. In: Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16). Red Hook, NY, USA: Curran Associates Inc.; 2016. p. 2360-2368.

[21] van den Oord A., Vinyals O., Kavukcuoglu K. Neural Discrete Representation Learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 6309-6318. Available at: https://arxiv.org/pdf/1711.00937 (accessed 29.01.2025).

[22] Razavi A., van den Oord A., Vinyals O. Generating Diverse High-Fidelity Images with VQ-VAE-2. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'17). Red Hook, NY, USA: Curran Associates Inc.; 2019. p. 14866-14876. Available at: https://arxiv.org/pdf/1906.00446v1 (accessed 29.01.2025).

[23] Zhang R., Isola P., Efros A.A., Shechtman E., Wang O. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In; 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA; 2018. p. 586-595. https://doi.org/10.1109/CVPR.2018.00068

[24] Esser P., Rombach R., Ommer B. Taming Transformers for High-Resolution Image Synthesis. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Nashville, TN, USA: IEEE Press; 2021. p. 12868-12878. https://doi.org/10.1109/CVPR46437.2021.01268

[25] Zheng C., et al. MoVQ: Modulating Quantized Vectors for High-Fidelity Image Generation. *arXiv:*2209.09002. 2022. https://doi.org/10.48550/arXiv.2209.09002

[26] Chen J., et al. Deep Compression Autoencoder for Efficient High-Resolution Diffusion Models. *arXiv:*2410.10733. https://doi.org/10.48550/arXiv.2410.10733

[27] Isola P., Zhu J.-Y., Zhou T., Efros A.A. Image-to-Image Translation with Conditional Adversarial Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA: IEEE Press; 2017. p. 5967-5976. https://doi.org/10.1109/CVPR.2017.632

[28] Li X.L., Thickstun J., Gulrajani I., Liang P., Hashimoto T.B. Diffusion-LM improves controllable text generation. In: Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22). Red Hook, NY, USA: Curran Associates Inc.; 2022. Article number: 313. p. 4328-4343.

[29] He Z., et al. DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Vol. 1. p. 4521-4534. Available at: https://openreview.net/pdf?id=GPbiCYiDVqr (accessed 29.01.2025).

[30] Zhou K., Li Y., Zhao X., Wen J.-R. Diffusion-NAT: Self-Prompting Discrete Diffusion for Non-Autoregressive Text Generation. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics. Vol. 1. St. Julian's, Malta: Association for Computational Linguistics; 2024. p. 1438-1451. Available at: https://aclanthology.org/2024.eacl-long.86.pdf (accessed 29.01.2025).

[31] Sun P., et al. Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation. *arXiv:*2406.06525. 2024. https://doi.org/10.48550/arXiv.2406.06525

[32] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Miami, FL, USA: IEEE Press; 2009. p. 248-255. https://doi.org/10.1109/CVPR.2009.5206848

Поступила 29.01.2025; одобрена после рецензирования 24.03.2025; принята к публикации 03.04.2025.
Submitted 29.01.2025; approved after reviewing 24.03.2025; accepted for publication 03.04.2025.

Об авторах:

Абрамов Владимир Геннадьевич, доцент кафедры алгоритмических языков факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), кандидат физико-математических наук, доцент, **ORCID: https://orcid.org/0000-0002-6818-9378,** vlabr@cs.msu.ru

Громов Михаил Алексеевич, аспирант кафедры алгоритмических языков факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0009-0002-0572-5898,** 2sge@mail.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Vladimir G. Abramov, Associate Professor of the Department of Algorithmic Languages, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), Cand. Sci. (Phys.-Math.), Associate Professor, **ORCID: https://orcid.org/0000-0002-6818-9378,** vlabr@cs.msu.ru

Mikhail A. Gromov, Postgraduate Student of the Department of Algorithmic Languages, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0009-0002-0572-5898,** 2sge@mail.ru

All authors have read and approved the final manuscript.