Генеративные нейронные сети

Санкт-Петербургский государственный университет
Кафедра статистического моделирования

Санкт-Петербург, 2025

Генеративные сети или **генеративно-состязательные сети (Generative Adversarial Networks, GAN)** — это нейронные сети, которые учатся генерировать реалистичные образцы данных, на которых они обучались.

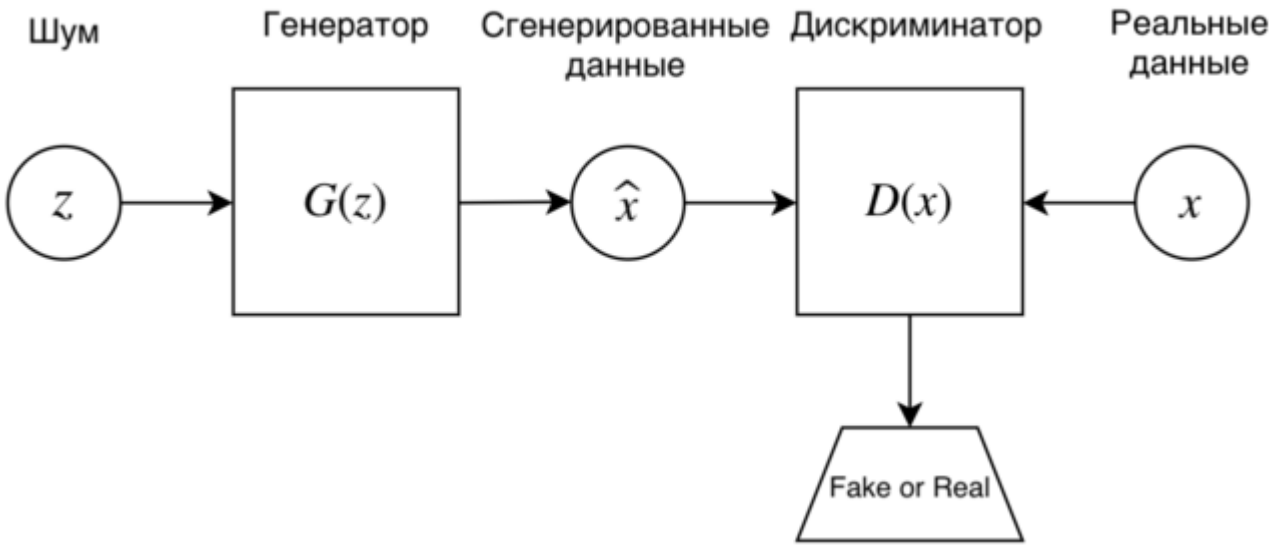


Рис.: Архитектура **GAN**

Архитектура GAN

GAN состоит из двух частей:

1. **Генератор** — это нейронная сеть, которая создает новые данные, основываясь на случайном шуме.

   Задача: генерировать данные, которые настолько похожи на реальные, что дискриминатор не сможет отличить их от настоящих.

2. **Дискриминатор** — это нейронная сеть, которая оценивает данные и пытается определить, являются ли они реальными или сгенерированными.

   Задача: стать настолько точным, чтобы уметь различать настоящие данные и подделки, созданные генератором.

Принцип работы **GAN** можно описать как состязание между генератором и дискриминатором. Генератор пытается обмануть дискриминатор, создавая реалистичные данные, в то время как дискриминатор пытается улучшить свои навыки различения.

Постановка задачи дискриминатора

Имеем следующие параметры:

1. x — реальные данные, z — шумовой вектор, $\hat{x} = G_\theta(z)$ — сгенерированные данные;
2. $p(x)$ — распределение реальных данных, $p(z)$ — априорное распределение шума;
3. G_θ — генератор с параметром θ. Генератор отображает векторы $z \sim \mathcal{N}(0, I)$ в $\hat{x} \sim q(x)$, распределение которых приближает реальное распределение данных p(x);
4. D_ϕ — дискриминатор с парамтером ϕ. Дискриминатор каждому реальному сэмплу x и фейковому $\hat{x}$ ставит в соответствие вероятность $D(x)$, которая оценивает степень принадлежности x к реальным данным, то есть он решает задачу бинарной классификации. Решим при помощи минимизации бинарной кросс-энтропии:

$$\min_\phi \mathbb{E}_{x\sim p(x)} - \log D_\phi(x) + \mathbb{E}_{\hat{x}\sim q(x)} - \log\left[1 - D_\phi(\hat{x})\right].$$

Постановка задачи генератора

Учитывая обозначение $\hat{x} = G_\theta(z)$, и то, что мы пытаемся максимизировать вероятность принадлежности к реальным данным, как её оценивает дискриминатор, задачу, которую решает генератор, можно расписать так:

$$\theta^* = \operatorname*{argmax}_\theta \mathbb{E}_{z\sim p(z)} D_\theta(G_\theta(z)) =$$

$$= \operatorname*{argmin}_\theta \mathbb{E}_{z\sim p(z)} - D_\theta(G_\theta(z)) =$$

$$= \operatorname*{argmin}_\theta \mathbb{E}_{z\sim p(z)} [1 - D_\theta(G_\theta(z))] =$$

$$= \operatorname*{argmin}_\theta \mathbb{E}_{z\sim p(z)} \log [1 - D_\theta(G_\theta(z))] =$$

$$= \operatorname*{argmax}_\theta \mathbb{E}_{z\sim p(z)} - \log [1 - D_\theta(G_\theta(z))].$$

Постановка задачи генератора и дискриминатора

Это равенство позволяет записать задачи, которые решают генератор и дискриминатор, вместе (избавимся от лишних минусов, сделав так, чтобы дискриминатор решал задачу максимизации):

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p(x)} \log D_{\phi}(x) + \mathbb{E}_{z \sim p(z)} \log \left[1 - D_{\phi}(G_{\theta}(z))\right].$$

Получается, что генератор и дискриминатор пытаются оптимизировать одну функцию: генератор её минимизирует, а дискриминатор максимизирует. Обозначим эту функцию (минус бинарную кросс-энтропию) как $\mathcal{L}_{\theta,\phi}$. Тогда эту задачу оптимизации можно записать в сокращённом виде:

$$\min_{\theta} \mathcal{L}_{\theta,\phi}.$$

Оптимальный дискриминатор

Исходная задача для дискриминатора:

$$\min_{\phi} \mathbb{E}_{x\sim p(x)} \log D_\phi(x) + \mathbb{E}_{\hat{x}\sim q(x)} \log\left[1 - D_\phi(\hat{x})\right].$$

В непрерывной форме:

$$\int \Big[p(x) \log D(x) + q(x) \log\left[1 - D(x)\right]\Big] \,\mathrm{d}x.$$

Мы хотим максимизировать это выражение по $D(x)$ для каждого x. Дифференцируем подынтегральное выражение по $D(x)$:

$$\frac{\mathrm{d}}{\mathrm{d}D}\Big[p(x) \log D(x) + q(x) \log\left[1 - D(x)\right]\Big] = \frac{p(x)}{D(x)} - \frac{q(x)}{1 - D(x)} = 0$$

Оптимальный дискриминатор D^* при любом фиксированном генераторе G:

$$D^*(x) = \frac{p(x)}{p(x) + q(x)}.$$

Интуитивное понимание работы генератора в GAN

Интуиции работы метода обучения GAN со стороны генератора можно сформулировать следующим образом:

1. Мы замеряем, насколько реалистичными являются сгенерированные сэмплы $\hat{x}_1, \ldots, \hat{x}_n$, используя для этого оптимальный дискриминатор $D^*(x)$.
2. Мы хотим увеличить отклик дискриминатора на каждом сэмпле, т.е. пытаемся модифицировать каждый предсказанный элемент $\hat{x}_i$ так, чтобы на нём стало выше значение $D^*(\hat{x}_i)$.
3. Как нужно модифицировать плотность $q(x)$, чтобы она стала ближе к $p(x)$, если к плотности распределения мы имеем доступ только через сэмплы из него?

Уменьшение расстояния между распределениями (1)

Подставив выражение для оптимального дискриминатора в $\mathcal{L}$, мы избавимся от внутренней максимизации в исходной задаче и оставим только внешнюю минимизацию по параметрам генератора. Мы получим в явном виде функцию потерь, которую минимизирует генератор, обозначим её за $\mathcal{D}_\theta$:

$$\mathcal{D}_\theta = \max_\phi \mathcal{L}_{\theta,\phi} = \mathbb{E}_{x\sim p(x)} \log D^*(x) + \mathbb{E}_{x\sim q(x)} \log \left[1 - D^*(x)\right] =$$

$$= \mathbb{E}_{x\sim p(x)} \log \frac{p(x)}{p(x)+q(x)} + \mathbb{E}_{x\sim q(x)} \log \left[1 - \frac{p(x)}{p(x)+q(x)}\right] =$$

$$= \int p(x) \log \frac{p(x)}{p(x)+q(x)} \,\mathrm{d}x + \int q(x) \log \frac{q(x)}{p(x)+q(x)} \,\mathrm{d}x.$$

Уменьшение расстояния между распределениями (2)

Упростим выражение для $\mathcal{D}(x)$ еще раз, прибавив и отняв $\log 4$, а также учитывая, что $\int p(x)\,\mathrm{d}x = 1$ и $\int q(x)\,\mathrm{d}x = 1$:

$$\mathcal{D}_\theta = -\log 4 + \int p(x) \log \frac{2p(x)}{p(x)+q(x)}\,\mathrm{d}x + \int q(x) \log \frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$$

$$= -\log 4 + KL\left(p \,\middle\|\, \frac{p+q}{2}\right) + KL\left(q \,\middle\|\, \frac{p+q}{2}\right)$$

$$= -\log 4 + 2 \times JSD\left(p \,\|\, q\right).$$

KL-дивергенция (Кульбака-Лейблера):

$$KL\left(P \,\middle\|\, Q\right) = \int P(x) \log \frac{P(x)}{Q(x)}\,\mathrm{d}x,$$

которая показывает, насколько два распределения отличаются друг от друга. Через $JSD\left(p \,\|\, q\right)$ обозначаем дивергенцию Йенсена-Шеннона.

Уменьшение расстояния между распределениями (3)

Вывод

Получается, что при оптимальном дискриминаторе генератор, решая внешнюю задачу оптимизации, уменьшает расстояние между распределениями реальных и фейковых данных, действительно приближая их друг к другу.

Исходя из этого и предполагая, что параметризация генератора и дискриминатора позволяет достичь оптимума, мы можем сформулировать алгоритм обучения GAN.

Алгоритм обучения GAN

1. Решить внутреннюю задачу максимизации по ϕ, повторяя шаги ниже до сходимости по параметрам дискриминатора ϕ ко оптимальному значению ϕ^*:
   - Составить мини-батч сэмплов шума $z_1, \ldots, z_n$ из $p(z)$;
   - Составить мини-батч сэмплов данных $x_1, \ldots, x_n$ из $p(x)$;
   - Обновить дикриминатор, сделав шаг вверх по его градиенту:

$$\nabla_\phi \frac{1}{n} \sum_{i=1}^{n} \left[\log D_\phi(x_i) + \log \left(1 - D_\phi\big(G_\theta(z_i)\big) \right) \right].$$

2. Сделать шаг SGD для внешней задачи минимизации по θ:
   - Составить мини-батч сэмплов шума $z_1, \ldots, z_n$ из $p(z)$;
   - Обновить генератор, сделав шаг вниз по его градиенту:

$$\nabla_\theta \frac{1}{n} \sum_{i=1}^{n} \log \left(1 - D_{\phi^*}\big(G_\theta(z_i)\big) \right) = \frac{1}{n} \sum_{i=1}^{n} - \frac{\nabla_\theta D_{\phi^*}\big(G_\theta(z_i)\big)}{1 - D_{\phi^*}\big(G_\theta(z_i)\big)}.$$

Качественная оценка модели

Для качественной оценки изображений, созданных генератором, используем следующие характеристики:

1. Сходство с изображениями обучающей выборки;
2. Отсутствие дубликатов из обучающей выборки;
3. Разнообразие изображений;
4. Отсутствие артефактов;

Frechet Inception Distance

Принцип работы: **FID** сравнивает два распределения высокоуровневых признаков для реальных и сгенерированных картинок, используя в качестве их приближения многомерные гауссианы.

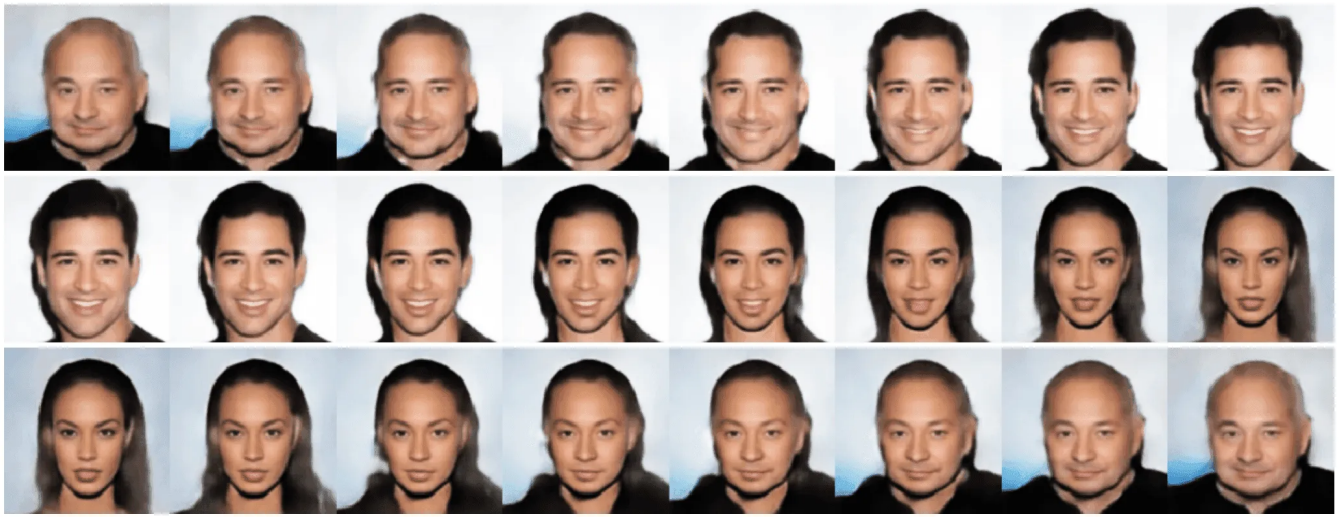
Для измерения расстояния между этими двумя распределениями используется метрика Вассерштейна:

$$\textbf{FID} = ||\mu - \hat{\mu}||^2 - \textbf{Tr}\Big(\Sigma + \hat{\Sigma} - 2\big(\Sigma\hat{\Sigma}\big)^{\frac{1}{2}}\Big),$$

где $\mu \in \mathbb{R}^C$ и $\Sigma \in \mathbb{R}^{C\times C}$ — среднее и матрица ковариаций глубоких признаков $F_i \in \mathbb{R}^{C\times H\times W}$, которые считаются по выборке из N реальных картинок. При этом как средние, так и матрицы ковариаций считаются по объединению всех признаков со всех картинок без учёта пространственной размерности. То же самое делается для сгенерированных картинок, для них средние и ковариации обозначены как $\hat{\mu}$ и $\hat{\Sigma}$.

Интерполяции в скрытом пространстве

Возьмём два случайных вектора z_1 и z_2 из $p(z)$. Рассмотрим все векторы, которые лежат между ними $z = \alpha z_1 + (1 - \alpha) z_2$, где $\alpha \in [0, 1]$. К каждому такому вектору z применим наш генератор и получим $\hat{x}$ для промежуточных векторов и $\hat{x}_1$, $\hat{x}_2$ для z_1, z_2. Для правильно обученного **GAN** мы увидим следующую картинку: при изменении коэффициента α изображение $\hat{x}$ должно плавно меняться и перетекать из $\hat{x}_1$ в $\hat{x}_2$. При этом каждая промежуточная картинка должна быть так же реалистичным сэмплом.



Проблемы обучения GAN

Большинство GAN подвержено следующим проблемам:

1. **Схлопывание мод распределения**: генератор выдает ограниченное количество разных образцов;
2. **Проблема стабильности обучения**: параметры модели дестабилизируются и не сходятся;
3. **Исчезающий градиент**: дискриминатор становится слишком сильным, а градиент генератора исчезает и обучение не происходит;
4. **Проблема запутывания**: выявление корреляции в признаках, не связанных (слабо связанных) в реальном мире;
5. Высокая чувствительность к гиперпараметрам.

Рекомендации при обучении GAN

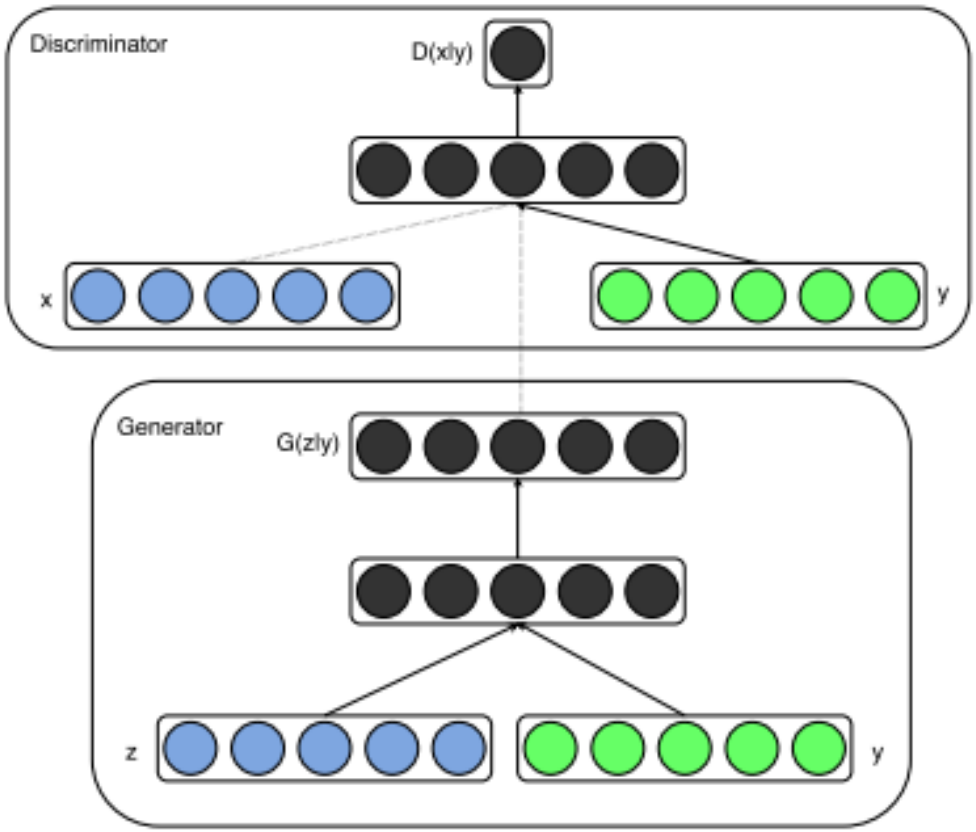
Практические советы, которые могут помочь при обучении GAN:

1. Нормализация данных. Все признаки в диапазоне $[-1, 1]$;
2. Замена функции ошибки для G с $\min \log (1 - D)$ на $\max \log D$, так как исходный вариант имеет маленький градиент на раннем этапе обучения и большой градиент при сходимости, а предложенный наоборот;
3. Сэмплирование из многомерного нормального распределения вместо равномерного;
4. Использовать метки для данных, если они имеются, то есть обучать дискриминатор еще и классифицировать образцы.

Conditional GAN

y — дополнительное условие для генератора и дискриминатора (Метка класса, изображение или данные из других моделей):

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p(x)} \log D_{\phi}(x|y) + \mathbb{E}_{z \sim p(z)} \log \left[1 - D_{\phi}(G_{\theta}(z|y))\right].$$



StackGAN (1)

StackGAN — порождающая состязательная сеть для генерации фото-реалистичных изображений исходя из текстового описания. Генерировать фото-реалистичные изображения на обычных **GAN** сложно, поэтому была придумана двух-этапная модель генерации:

1. **Stage-I GAN** рисует скетчи с примитивными формами и цветами, основанные на текстовом описании, в низком разрешении.
2. **Stage-II GAN** принимает на вход изображения с первого этапа и текстовое описание, и генерирует изображение в высоком разрешении с фото-реалистичными деталями.

StackGAN (2)

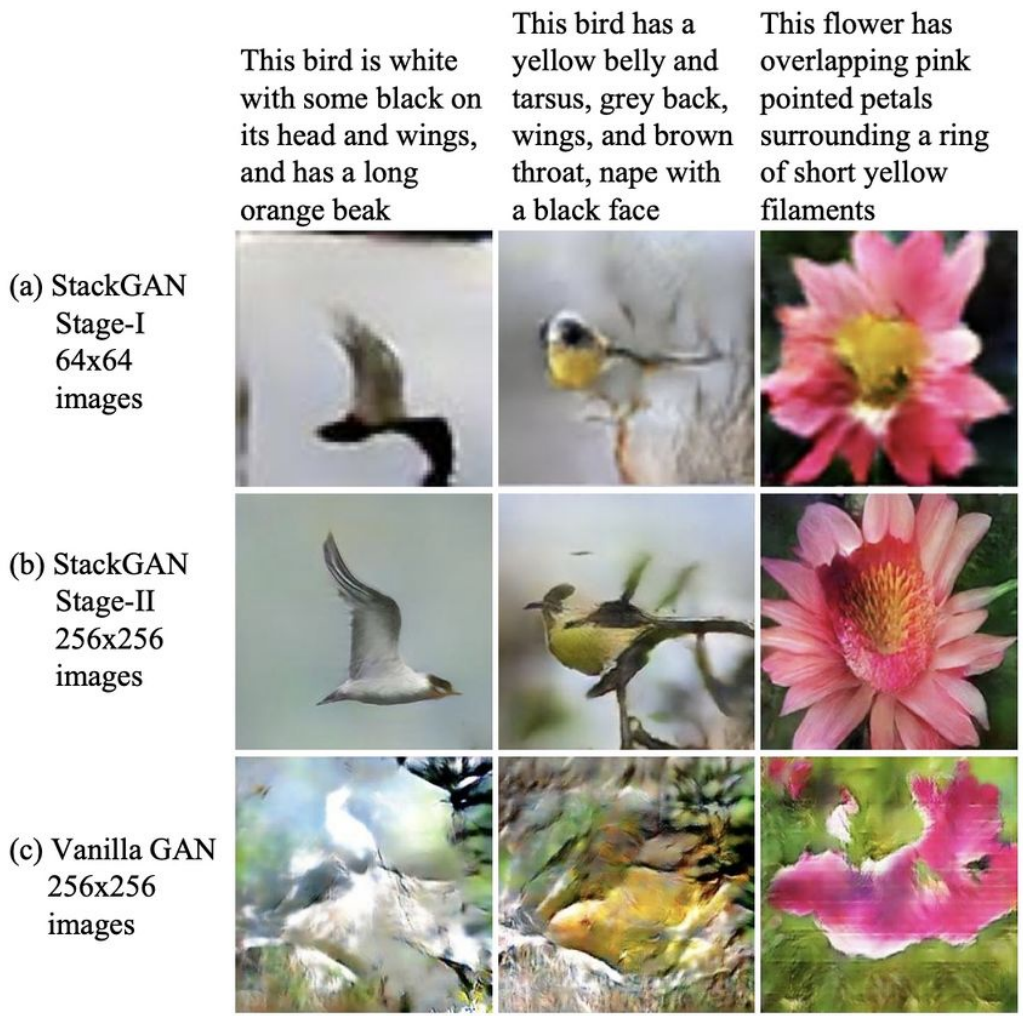


Рис.: Пример работы порождающей состязателной сети для генерации фото-реалистичных изображений StackGAN

StackGAN, Stage-I

1. Conditioning Augmentation — $\mathcal{N}\left(\mu(\phi_t), \Sigma(\phi_t)\right)$, где t — текстовое описание, а ϕ_t — векторное представление.
2. Регуляризация:

$$r = D_{KL}\left(\mathcal{N}\left(\mu(\phi_t), \Sigma(\phi_t)\right) \;||\, \mathcal{N}\left(0, I\right)\right).$$

3. Тренировка дискриминатора D_0 и генератора G_0:

$$L_{D_0} = \mathbb{E}_{(I_0,t)\sim p(x)}\left[\log D_0\left(I_0, \phi_t\right)\right]$$

$$+E_{z\sim t, t\sim p(x)}\left[\log\left(1 - D_0\left(G_0(z, \hat{c}_0), \phi_t\right)\right)\right];$$

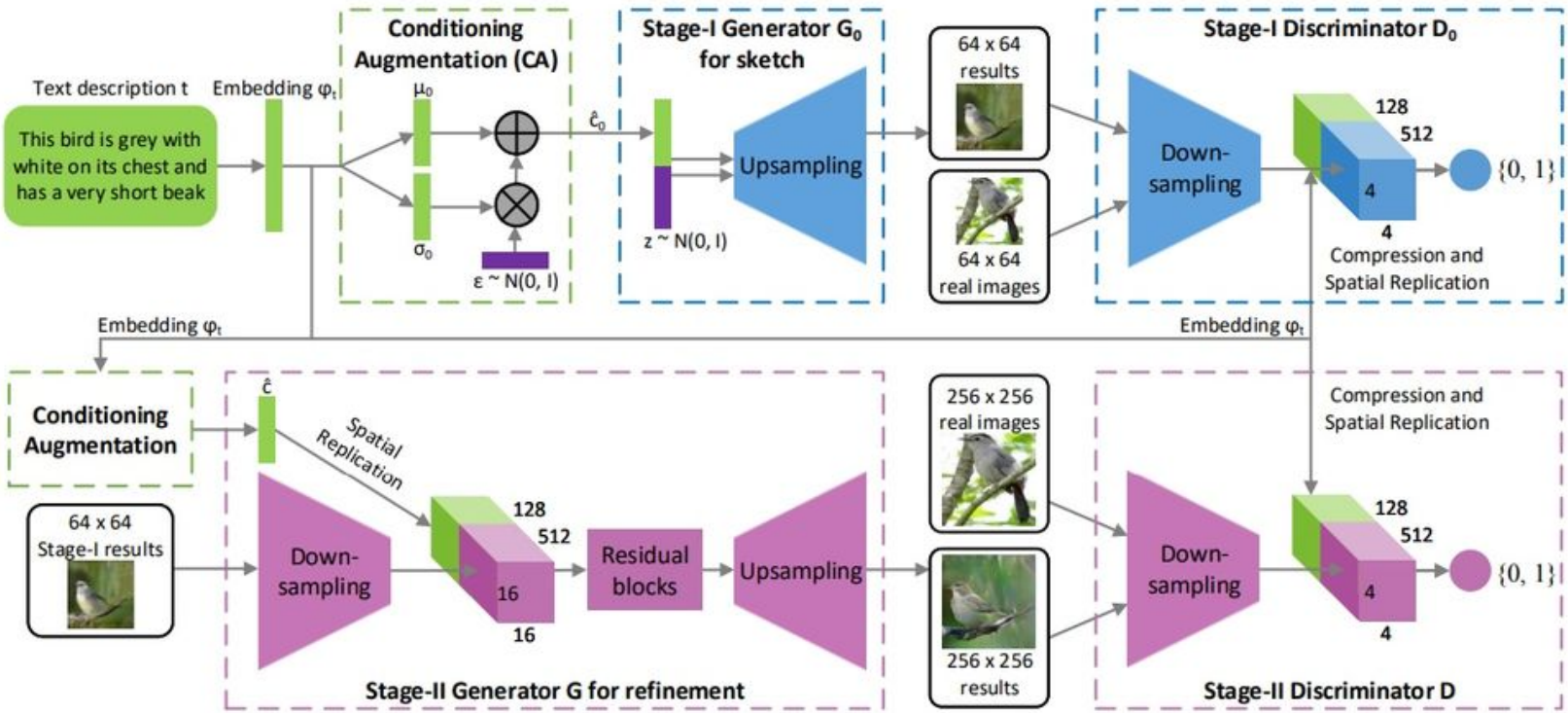
$$L_{G_0} = \mathbb{E}_{z\sim t, t\sim p(x)}\left[\log\left(1 - D_0\left(G_0(z, \hat{c}_0), \phi_t\right)\right)\right] + \lambda r,$$

где реальное изображение I_0 и описание текста t берутся из реального распределения данных $p(x)$, z — шумовой вектор.

$$L_D = \mathbb{E}_{(I,t)\sim p(x)}[\log D(I, \phi_t)] + \mathbb{E}_{s_0\sim p_{C_0}, t\sim p(x)}[\log(1 - D(G(s_0, \hat{c}), \phi_t))];$$

$$L_G = \mathbb{E}_{s_0\sim p_{C_0}, t\sim p(x)}\left[\log\left(1 - D\left(G(s_0, \hat{c}), \phi_t\right)\right)\right] + \lambda r,$$

где $s_0 = G_0\left(z, \hat{c}_0\right)$ — результат работы генератора **Stage-I GAN**.



LAPGAN (1)

LAPGAN — генеративная параметрическая модель, представленная пирамидой лапласианов с каскадом сверточных нейронных сетей внутри, которая генерирует изображения постепенно от исходного изображения с низким разрешением к изображению с высоким. На каждом уровне пирамиды обучается сверточная генеративная модель, используя подход порождающих состязательных сетей. Такая стратегия позволяет декомпозировать задачу генерации изображений на последовательность уровней, что упрощает ее решение.

Пирамида лапласианов — линейное обратимое представление изображений, состоящее из набора частотных полос изображений.

LAPGAN (2)

1. Пусть $d(\cdot)$ — операция сжатия изображения размера $j \times j$ так, что новое изображение $d(I)$ имеет размеры $j/2 \times j/2$
2. $u(\cdot)$ — операция расширения такая, что $u(I)$ имеет размеры $2j \times 2j$.

Тогда пирамида гауссианов имеет вид $\mathcal{G}(I) = [I_0, I_1, \ldots, I_k]$, где $I_0 = I$ и I_k представляет собой k раз выполненное применение $d(\cdot)$.

Коэффициенты h_k на каждом уровне пирамиды:

$$h_k = \mathcal{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = I_k - u\left(I_{k+1}\right).$$

LAPGAN (3)

Интуитивно каждый уровень захватывает структуру изображения. Конечный слой пирамиды лапласианов h_k это не разница изображений, а низко-частотное представление равное гаусиану $h_k = I_k$. Реконструкция по пирамиде лапласианов происходит обратным проходом по ней: $I_k = u(I_{k+1}) + h_k$.

Подход представленный в **LAPGAN** работает по такому же принципу, только на каждому шаге вместо коэфициентов h_k используются генераторы $\{G_0, \ldots, G_k\}$, каждый из которых захватывает распределение коэфициентов h_k для реальных изображений на разных уровнях пирамиды лапласиана:

$$\tilde{I}_k = u(\tilde{I}_{k+1}) + \tilde{h}_k = u(\tilde{I}_{k+1}) + G_k\big(z_k, u(\tilde{I}_{k+1})\big).$$

LAPGAN (4)

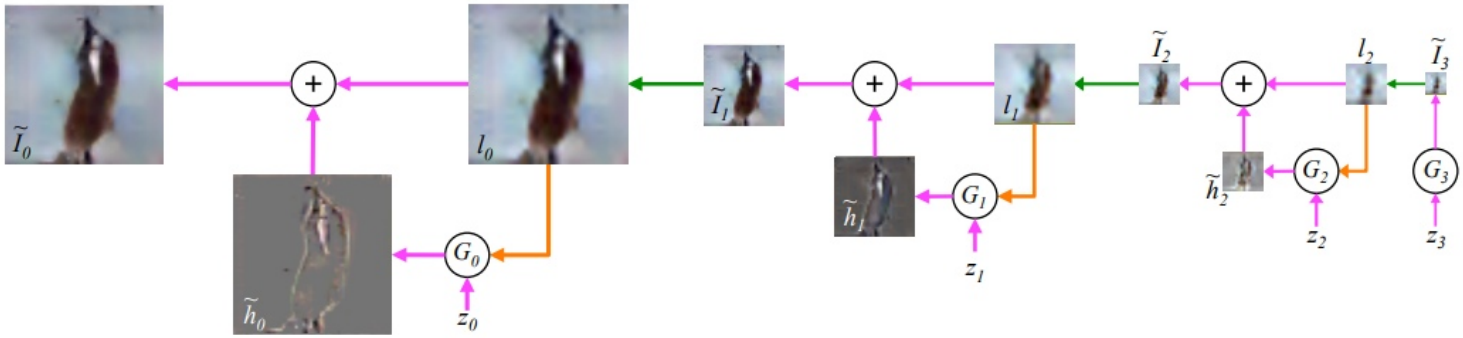


Рис.: Процедура семплинга для модели LAPGAN

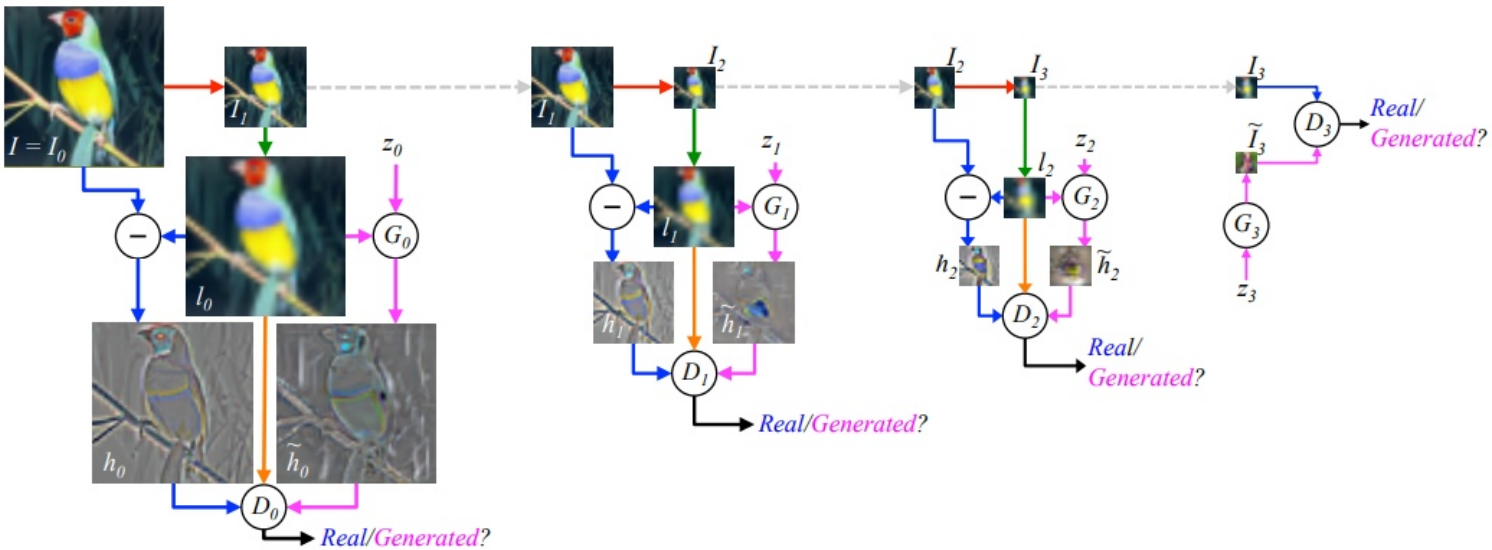


Рис.: Процедура обучения модели LAPGAN