Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Баранова Дарья Дмитриевна

РАЗРАБОТКА СЕРВИСА ДЛЯ ГЕНЕРАЦИИ РЕАЛИСТИЧНЫХ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ЭСКИЗА С ПОМОЩЬЮ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Выпускная квалификационная работа

студента образовательной программы магистратуры «Машинное обучение и высоконагруженные системы» по направлению подготовки *01.04.02 Прикладная математика и информатика*

Рецензент
ученая степень, ученое звание,
должность

Руководитель
ученая степень, ученое звание, должность

__Нарек Алвандян__
И.О. Фамилия

Москва, 2023 год

Аннотация

Работа содержит * глав, * страниц, * рисунков, * таблиц, * использованных источников, * приложений.

Оглавление

Введение

В настоящее время задача генерации изображений находится на острие развития науки. Множество задач сохраняют свою актуальность и требуют более тщательных исследований несмотря на недавно произошедшие прорывы в области и большое количество существующих нестандартных решений.

Данная работа посвящена задаче Sketch-to-Image – преобразованию заданного эскиза в изображение. Другими словами, необходимо сгенерировать реалистичное изображение на основе наброска (эскиза), сохраняя при этом определенную семантику и границы отдельных элементов. Эта задача играет важную роль в задачах синтеза и обработки изображений, таких как редактирование фотографий и раскрашивание. Синтез изображений на основе эскизов позволяет людям, не являющимся художниками, создавать реалистичные изображения без значительных художественных навыков или специальных знаний в области синтеза изображений. Разработанные решения могут использоваться в качестве мощных инструментов в индустрии художественного дизайна, или, к примеру, в качестве инструмента, помогающего в составлении реалистичного фоторобота подозреваемого.

Сложность задачи заключается в том, что элементы эскиза почти всегда будут сильно отличаться от границ соответствующих реалистичных зарисованных элементов. Поэтому необходимо контролировать генерацию, сохраняя баланс между реализмом и соответствием результата исходному эскизу. Для этого необходимо реализовать подход, который позволит регулировать при генерации уровень реализма и степень соответствия зарисовке. Дополнительной сложностью выступает необходимость генерировать детали, которые отсутствуют на эскизе, но необходимы для реалистичного изображения.

Разработанное решение должно быть прикладным, то есть являться комплексным законченным инструментом – продуктом, который можно использовать.

Таким образом, в качестве объекта исследования выступает готовое изображение, которое было получено в процессе генерации. А предметом исследования выступает процесс получения этого изображения, то есть процесс его генерации с помощью нейросетевой модели на основе заданного наброска.

Целью работы является создание системы генерации изображений на основе эскизов (или зарисовок), сохраняя при этом заданную семантику и задавая определенный уровень реализма.

Поставленная цель предполагает решение следующих задач:

– выполнить обзор существующих методов генерации изображений на основе эскизов;

– получить размеченные обучающие данные;

– разработать метод генерации изображения;

– экспериментально определить оптимальную архитектуру сети и оптимальную реализацию метода генерации;

– реализовать систему генерации в виде полноценного сервиса;

– выполнить тестирование системы.

Глава 1 Анализ существующих подходов

1.1 Существующие подходы генерации изображений по эскизу

Если рассматривать задачу в более общем виде, то можно её свести к Image-to-Image задаче, которая является ключевой во всей области компьютерного зрения. Большинство подходов к решению этой задачи основывается на нескольких базовых архитектурах, таких как GAN и Diffusion models, которые будут описываться далее. Задача Sketch-to-Image не является исключением, поэтому её существующие решения также основываются на указанных моделях с некоторыми модификациями в архитектуре. Рассмотри подробнее некоторые из них.

1.1.1 GAN

1.1.1.1 Описание подхода GAN

Generative Adversarial Networks (GANs) представляют собой подход к генерации данных с помощью нейронных сетей. При таком подходе используются две нейронные сети – генератор и дискриминатор. Первая сеть – генератор – генерирует необходимые данные, стараясь создать их максимально похожими на обучающие данные. Вторая сеть – дискриминатор – представляет из себя классификатор, который пытается различить между собой обучающие данные и данные, полученные из генератора.

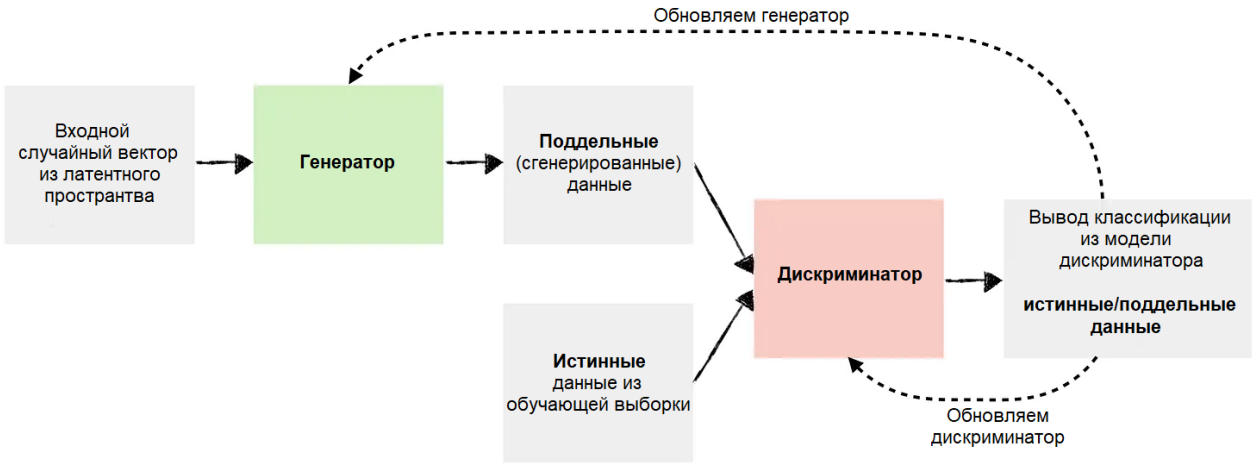

Рис. Стандартная архитектура GAN. Автор: https://solclover.com/

Весь подход состоит в том, чтобы провести обучение этой пары состязающихся между собой моделей таким образом, чтобы генератор пытался обмануть дискриминатор, создавая все более и более реалистичные данные, а дискриминатор сохранял возможность находить и отличать их от подлинных. Таким

образом, после обучения получим две готовые сети, которые можно использовать в различных задачах. Для задачи генерации данных (в частности, изображений) найдет применение обученный генератор, дискриминатор же будет не нужен. Однако он может быть полезен в других задачах компьютерного зрения, рассмотрение которых выходит за рамки данной работы.

1.1.1.2 Conditional Adversarial Networks (pix2pix)

Стандартную архитектуру GAN можно модифицировать, делая генерацию более управляемым процессом. В частности, для того, чтобы генерировать изображения определенного класса можно на вход моделям дополнительно подавать лэйбл или метку этого класса.

Такая модификация GAN называется Conditional Generative Adversarial Network (cGAN).

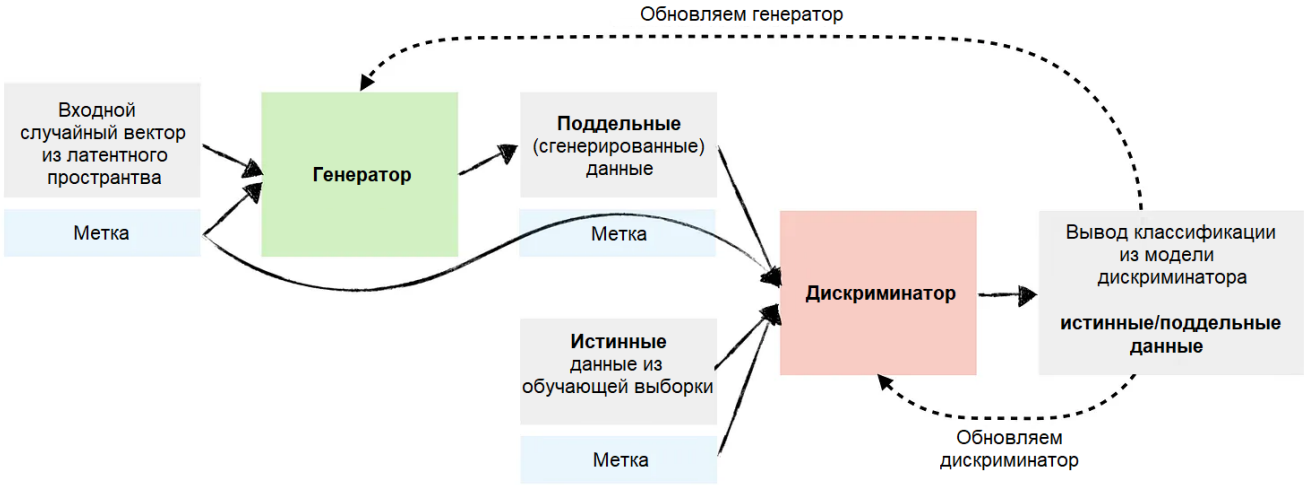

Рис. Архитектура условного (conditional) GAN. Автор: https://solclover.com/

Входной меткой может выступать что угодно, в частности, это может быть текстовое описание или исходное изображение, которое затем нужно модифицировать. Для задачи генерации реалистичного изображения по наброску входной меткой как раз будет выступать этот набросок.

В работе [https://arxiv.org/abs/1611.07004] исследователями использовалась такая архитектура cGAN. При этом, входным изображением не всегда выступал именно набросок, так как задача в исследовании ставилась иначе – создать «переводчик» изображений из одной области в другую. В качестве области может

выступать RGB-изображение, карта границ, карта семантических меток и т. д. Из-за этого входным изображением выступало соответствующее подмножество изображений, содержащих объект в соответствующей области.

Такой подход может частично решать поставленную в работе задачу. Для этого исходной областью будет выступать карта границ объекта, а целевой областью – RBG-изображение (см. рисунок 1).

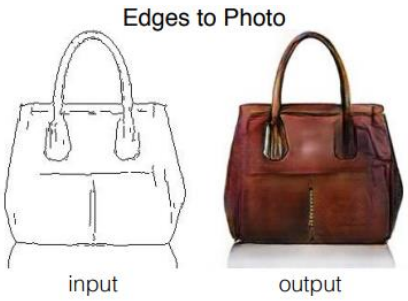

Рис. 1 Результат работы pix2pix модели при переводе изображения из карты границ в RGB-изображение

Недостатком данного решения является сильное ограничение на соответствие границ объекта на сгенерированном изображении граничным линиям эскиза. Такое ограничение сильно портит качество генерации в задаче генерации по эскизу, так как границы эскиза зачастую условны и редко совпадают с очертаниями соответствующего реалистичного объекта.

1.1.1.3 Contextual GAN

Указанный недостаток был также описан и исправлен в рамках другой более новой работы [https://arxiv.org/pdf/1711.08972.pdf]. По словам авторов, её главное преимущество заключается именно в отсутствии упомянутого строгого ограничения на пиксельное соответствие для границ.

Новый подход заключается в том, что теперь генератор будет пытаться восстанавливать изображение, как схожим образом это происходит в задаче Inpainting. Другими словами, авторы взглянули на задачу с новой стороны – необходимо не сгенерировать изображение, а восстановить его, представив эскиз в качестве поврежденного исходного изображения. В традиционной Inpainting задаче поврежденная часть входного изображения дополняется с использованием

окружающего содержимого изображения в качестве контекста. Однако в нашей задаче контекстом будет выступать эскиз.

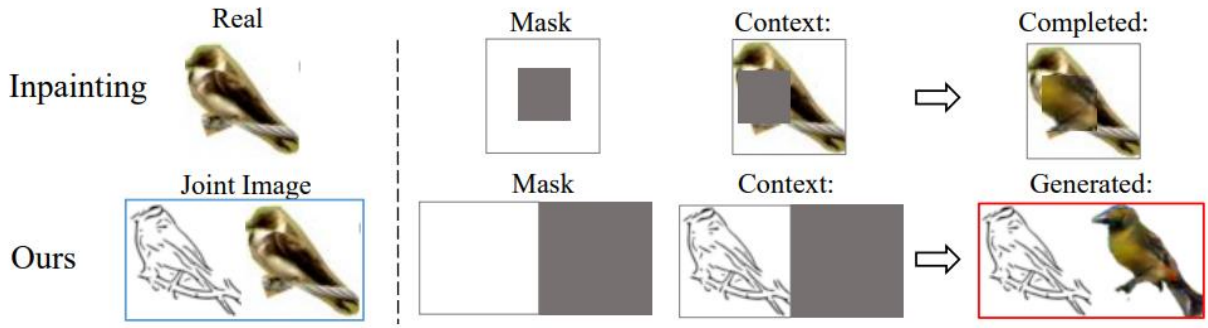

Рис. 2 Иллюстрация подхода Contextual GAN. Эскиз используется в качестве контекста

Для иллюстрации приведен рисунок 2. При дорисовке поврежденной части изображения птицы в верхнем ряду немаскированная часть птицы являются контекстуальной информацией. По аналогии, теперь «испорченной» частью будет выступать все изображение, которое необходимо сгенерировать, а «контекст» будет предоставляется входным эскизом (нижний ряд рисунка 2). При этом входным изображением теперь будет выступать объединенная пара – скетч слева и поврежденная часть изображения справа.

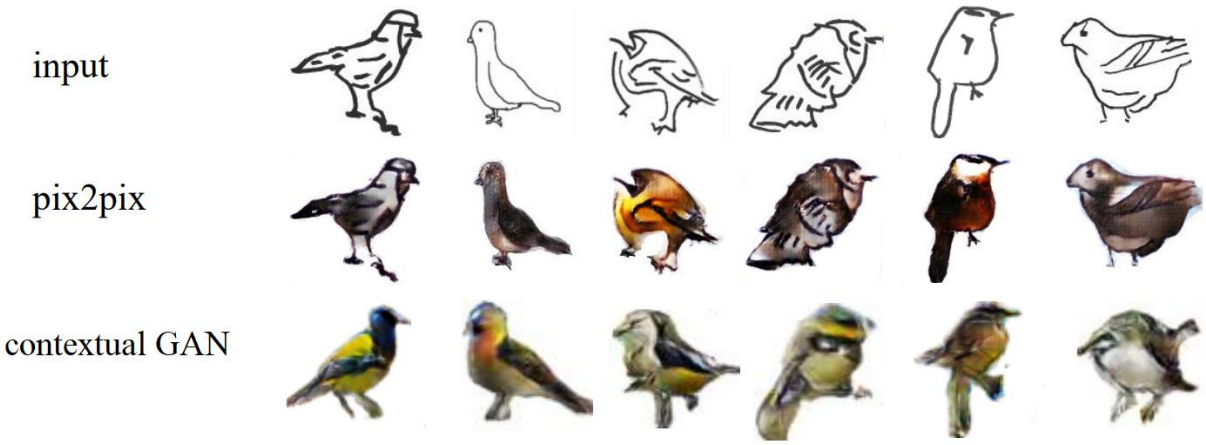

Рис. 3 Сравнение результатов работы Contextual GAN и Pix2Pix на примере генерации птиц

Сравнивая полученные результаты с результатами работы pix2pix модели, которая была описана ранее, авторы статьи приходят к выводу, что pix2pix модель имеет тенденцию строго следовать очертаниям входного эскиза, в то время как Contextual GAN сохраняет устойчивость к искаженным наброскам, так как, по

мнению авторов, больше заточена под воспроизведение изученного распределения данных.

1.1.2 Diffusion model

1.1.2.1 Описание подхода diffusion model

Основная идея диффузионной модели состоит в том, что с помощью итеративного прямого диффузионного процесса поданное на вход изображение «разрушается», смешиваясь с шумом до тех пор, пока оно целиком не станет состоять из белого шума. После этого происходит обратный диффузионный процесс – процесс восстановления изображения или его «расшумление», который также происходит итеративно. [https://arxiv.org/abs/1503.03585] Таким образом, имея правильно обученную сеть, мы можем подавать ей на вход изображение, состоящее полностью из белого шума, и получать на выходе сгенерированное изображение. Рассмотрим подробнее прямой и обратный диффузионные процессы.

Прямой процесс заключается в том, что исходное изображение x_0 постепенно зашумляется, проходя определенное количество итераций t={0,..,T}, пока на последней итерации T изображение x_T не начнет содержать только белый шум, то есть пока не будет полностью состоять из данных нормального распределения N(0, I). Таким образом, обозначим как $q(x_t|x_{t-1})$ функцию зашумления изображения x_{t-1} до состояния x_t.

При обратном процессе изображение, наоборот, расшумляется, проходя итерации от x_T до x_0. На шаге i модель получает изображение x_i на вход, и возвращает его в менее зашумленном состоянии x_{i-1}, таким образом постепенно генерируя его. Итеративность процесса генерации позволяет добиться лучшего качества, а также отслеживать и контролировать генерацию на каждом шаге. Обозначим $p(x_{t-1}|x_t)$ как функцию расшумления изображения, то есть перевода из состояния x_t в состояние x_{t-1}.

Согласно введенным обозначениям, будем иметь:

$$q(x_t|x_{t-1}) = N(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I),$$

где:

$N(\cdot)$ – нормальное распределение,

$\sqrt{1-\beta_t}x_{t-1}$ – матожидание,

β_tI – дисперсия,

β_t – коэффициент, отвечающий за постепенное изменение шума в изображении, $\beta_t \in [0,1]$.

Также можно свернуть все итеративные переходы, соединив все преобразования в одну итерацию и избавившись от промежуточных вычислений:

$$q(x_t|x_0) = N(x_t, \sqrt{\bar{a}_t}x_0, (1-\bar{a}_t)I),$$

где:

$\alpha_t = 1 - \beta_t$,

$\bar{a}_t = \prod_{s=1}^{t} a_s$.

Теперь подробнее опишем обратный процесс.

$$p(x_{t-1}|x_t) = N(x_{t-1}, \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)),$$

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\epsilon(x_t,t)),$$

где:

$\epsilon(x_t,t)$ – шум, $\epsilon \sim N(0,I)$

$\Sigma_\theta(x_t,t)$ считается зафиксированной величиной и может быть установлена как β_t (в дальнейших работах этот параметр будет также учиться моделью).

Для реализации описанного процесса необходимо создать сеть, которая будет предсказывать шум, который присутствует на изображении [https://arxiv.org/abs/2006.11239]. После этого на каждой итерации необходимо вычитать предсказанный шум из входного изображения, делая его таким образом менее зашумленным. Функция потерь L_t будет являться стандартной среднеквадратичной ошибкой:

$$L_t = \|\epsilon - \epsilon(x_t,t)\|^2$$

Можно выразить x_{t-1} через x_t:

$$x_{t-1} = \begin{cases} \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\epsilon(x_t,t)\right) + \sqrt{\beta_t}\epsilon, & t > 1 \\ \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\epsilon(x_t,t)\right), & t = 1 \end{cases}$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Рис.6 Алгоритмы для процедуры обучения и выборки для диффузионной модели

Нужно заметить, что количество шума на каждой итерации может и будет отличаться, так как это позволяет достичь лучшего качества. При линейном характере изменения доли шума на изображении наблюдается несколько проблем, а именно:

1. На последних итерациях процесса, когда изображение приближается к тому, чтобы стать белым шумом и утратить последние элементы содержательной информации, визуально изменения уже не кажутся значительными (см рисунок). Можно сказать, что последняя треть или даже половина итераций уже выглядит как полностью зашумленное изображение.
2. При этом, на первых итерациях зашумления ключевая содержательная информация слишком быстро утрачивается.

Рис.7 Зашумление при линейном (вверху) и косинусном (внизу) подходе соответственно на каждой итерации t от 0 до T. В последней четверти линейного графика изображения представляют собой почти чистый шум, тогда как косинусный график добавляет шум медленнее.

Исходя из этого, в качестве альтернативы был предложен [https://arxiv.org/abs/2102.09672] косинусный характер изменения доли шума на изображении.

Архитектура сети представляет собой U-Net. На вход сети также будет подаваться номер итерации, представленный особым образом. Это важная деталь, так как, как было указано выше, в зависимости от номера итерации на изображении будет присутствовать разная доля шума.

1.1.2.2 Stable diffusion model

Stable Diffusion model является усовершенствованием классической diffusion модели. Она является latent diffusion model (LDM), что означает, что модель работает не с изображениями напрямую, а с их закодированными представлениями в латентном пространстве.

Stable Diffusion состоит из 3 частей: вариационного автоэнкодера (VAE), U-Net и дополнительного текстового энкодера, так как генерирует изображение на основе не только изображения, но и текстового описания.

Кодировщик VAE сжимает изображение из пространства пикселей в скрытое пространство меньшего размера, сохраняя основную семантику изображения и отбрасывая наименее фундаментальную информацию. Это позволяет увеличить скорость работы сети, так как размер данных уменьшился. Декодировщик VAE генерирует выходное изображение, преобразуя представление обратно в пространство пикселей.

Для эмбединга текста используется предварительно обученная сеть CLIP [https://arxiv.org/abs/2103.00020].

1.1.2.3 Classifier Guidance for Latent Diffusion Models

Одним из существующих подходов, помогающим делать генерацию данных с помощью диффузионных моделей более управляемой, является использование вспомогательного «руководящего» классификатора в процессе генерации (так называемый метод Classifier Guidance). Подход состоит в том, что при генерации помимо диффузионной модели также используется дополнительная модель-классификатор, loss которой может показывать, насколько генерируемое изображение действительно соответствует входным данным (например, текстовому описанию или эскизу, заданным на входе). При таком подходе производная полученной ошибки классификатора будет использоваться для обновления изображения после каждого шага генерации.

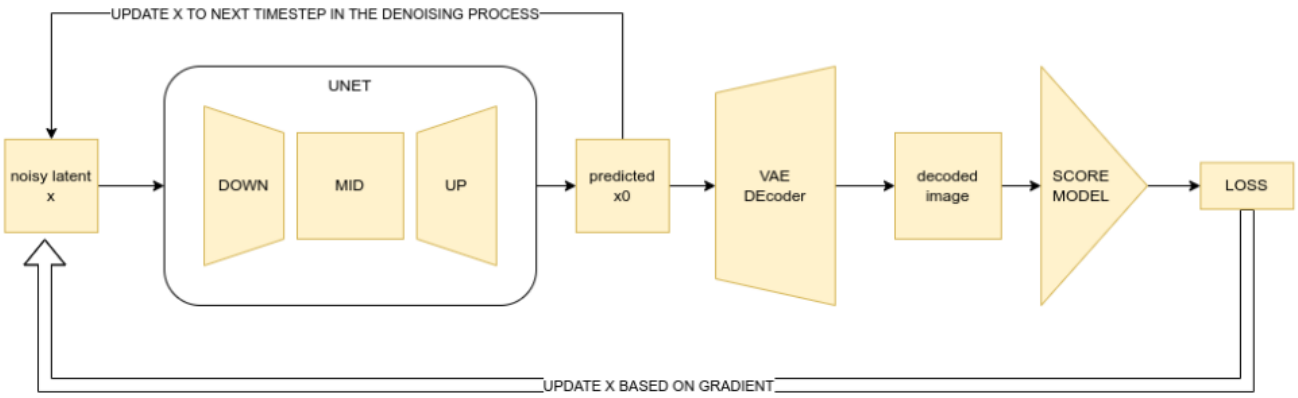

Рис. Стандартная реализация Classifier Guidance для латентных диффузионных моделей [https://wandb.ai/johnowhitaker/midu-guidance/reports/Mid-U-Guidance-Fast-Classifier-Guidance-for-Latent-Diffusion-Models--VmlldzozMjg0NzA1]

Для реализации такого обучения требуются большие вычислительные мощности и большое время работы, так как предполагается, что обратных проброс градиентов будет являться сложной и дорогостоящей операцией (требуется вычисление градиентов в модели классификации, затем декодере VAE (в случае моделей скрытой диффузии) и самой диффузионной U-Net на каждом этапе генерации).

Один из способов обойти это ограничение был описан в работе [https://arxiv.org/pdf/2207.12598.pdf]. Исследователи используют в качестве альтернативы две диффузионные модели – conditional и unconditional, и используют их вместе в процессе генерации. В работе утверждается, что такой подход дает схожие результаты, что и Classifier Guidance, однако требует сильно меньше вычислений и ресурсов. В качестве гиперпараметра будет выступать вес (пропорция) смешивания описанных моделей. Утверждается, что задав гиперпараметр оптимально можно достигнуть того же качества, что и с помощью дополнительной модели классификатора.

Также, две диффузионные модели можно совместить в одну, задав безусловную диффузию как модель, обусловленную нулевым токеном $\emptyset$. Именно такой подход и был использован в работе экспериментально. Обучение такой скомбинированной модели тогда будет происходить следующим образом. Гиперпараметром будет задаваться p_{uncond} – вероятность подать на вход нулевой

токен вместо определенного условия c для conditional модели. Случайным образом с вероятностью p_{uncond} будет задаваться вход на каждой итерации обучения.

При генерации шум будет вычисляться как комбинация выходов conditional и unconditional моделей:

$$\tilde{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) = (1 + w)\epsilon_\theta(\mathbf{z}_\lambda, \mathbf{c}) - w\epsilon_\theta(\mathbf{z}_\lambda)$$

1.1.2.4 Diffusion-based framework that generates images from Sketches and Strokes (DiSS) [https://arxiv.org/pdf/2208.12675.pdf]

В данной работе поставленная задача генерации фотореалистичных изображений на основе эскиза решается с помощью описанной выше диффузионной модели. В работе описан подход, при котором на вход подается эскиз, а также цветовая карта, которая является изображением такого же формата, что и изображение эскиза, но состоит из трех каналов и содержит в себе цветовую информацию, то есть своеобразную карту, где какой цвет должен располагаться на итоговом изображении. Пример приведен на рисунке ниже.

Рис. Пример входных и выходных данных в работе DiSS.

При этом стандартная диффузионная модель имеет несколько модификаций:

1. На вход модели (U-Net) подаются одновременно представление изображения эскиза в латентном пространстве и представление цветовой карты.
2. За основу метода обучения взят описанный выше подход, использующий комбинацию из conditional и unconditional моделей. Он был немного изменен под особенности конкретной задачи. Обучение состояло из двух этапов. Первый – обучение при полных данных, когда на вход подавались как признаки

эскиза в латентном пространстве c_{sketch}, так и признаки цветовой карты c_{stroke}. На втором этапе какое-то из условий убиралось (заменялось на пустой токен $\emptyset$), задавая своеобразную unconditional модель. Тогда комбинация выходов приобретала следующий вид:

$$\begin{aligned}\hat{\epsilon}_\theta(x_t, t, \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}}) &= \hat{\epsilon}_\theta(x_t, t, \emptyset, \emptyset) \\ &+ s_{\text{sketch}}(\hat{\epsilon}_\theta(x_t, t, \mathbf{c}_{\text{sketch}}, \emptyset) - \hat{\epsilon}_\theta(x_t, t, \emptyset, \emptyset)) \\ &+ s_{\text{stroke}}(\hat{\epsilon}_\theta(x_t, t, \emptyset, \mathbf{c}_{\text{stroke}}) - \hat{\epsilon}_\theta(x_t, t, \emptyset, \emptyset)).\end{aligned}$$

С помощью этой линейной комбинации контролируется степень соответствия генерируемого изображения каждому из условий (условию близости к эскизу и условию близости к карте цветов)

3. Степень реализма достигается путем использования метода итеративного уточнения скрытых переменных (iterative latent variable refinement), описанного в работе [https://arxiv.org/pdf/2108.02938.pdf]. Метод заключается в том, чтобы на каждом шаге генерации изображения x_{t-1} из x_t вычислять итоговое изображение x'_{t-1} как линейную комбинацию x_{t-1}, $\mathcal{L}_N(x_{t-1})$ и $\mathcal{L}_N(c_{comb_{t-1}})$, где $\mathbf{c}_{\text{comb}t-1} \sim q(\mathbf{c}_{\text{comb}t-1}|\mathbf{c}_{\text{comb}_0})$ с $\mathbf{c}_{\text{comb}_0} = \mathbf{c}_{\text{comb}}$, при этом c_{comb} – это сумма изображения эскиза и цветовой карты, а $\mathcal{L}_N$ представляет собой линейную операцию фильтрации низких частот (linear low pass filtering operation), которая заключается в том, чтобы понизить размерность изображения до размерности N и затем увеличить размерность изображения до исходного. Коэффициент N будет закладывать степень схожести итогового изображения с входными данными, определяя «насколько много частот входного изображения» будет учитываться при генерации итогового.

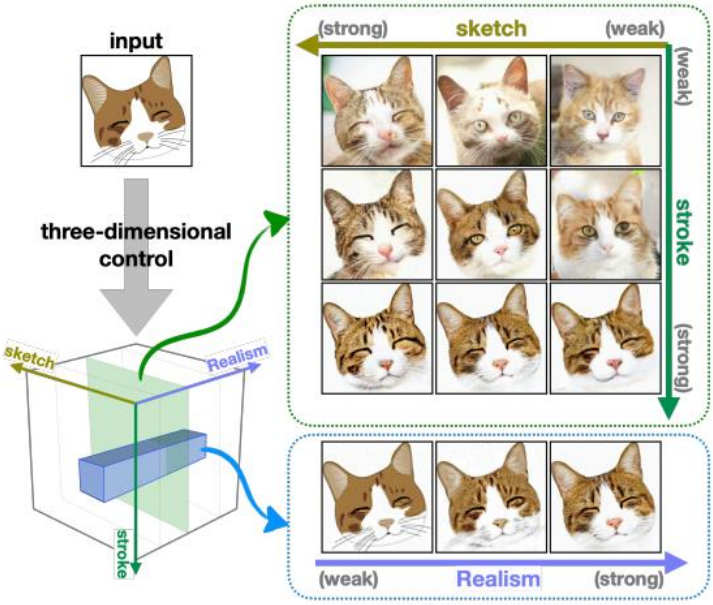

Рис.4 Пример полученных результатов работы модели DiSS при разных соотношениях соответствия эскиз-цвет-реализм

1.1.2.5 MLP Latent Guidance Predictor [https://arxiv.org/pdf/2211.13752v1.pdf]

В данной работе использовалась stable diffusion model, в которой также присутствовала модификация авторов. Основная идея подхода в том, что к готовой диффузионной модели вида text-to-image добавляется вспомогательная MLP сеть, которая будет выступать помощником в процессе генерации изображения.

Метод позволяет сгенерировать изображение на основе эскиза и текстового описания. В результате могут получиться разные результаты для одного эскиза – разница будет формироваться на основе различного текстового описания.

Рис.5 Примеры различной генерации при одном эскизе и разном текстовом описании

Можно сказать, что MLP в данном случае будет выступать в роли Classifier Guidance. Его задача заключается в учете границ эскиза, который предварительно переведен в латентное пространство, поэтому для его реализации не потребуется

раскодировать изображение из пространства признаков. Добавленный MLP обучается предсказывать истинный эскиз по переданным ему активационным слоям диффузионной модели. На выходе требуется, чтобы перцептрон выдавал представление максимально похожее на представление эскиза. Формально, для входного тензора изображения w и контекста c на итерации t обозначим соединенные (сконкатенированные) активации выбранных внутренних слоев сети $\{l_1, \ldots, l_n\}$ как:

$$F(w|c,t) = [l_1(w|c,t), \ldots, l_n(w|c,t)]$$

Тогда входное измерение MLP представляет собой сумму количества каналов выбранных слоев. На выходе ожидается, что перцептрон вернет изображение аналогичного размера, которое получается на выходе диффузионной сети после очередной итерации.

Также перцептрон должен уметь работать с входными данными с любой итерации диффузионной модели, поэтому дополнительно ему на вход подается номер итерации t.

После того, как перцептрон будет обучен, сеть будет считаться обученной полностью и готовой для генерации. Для того, чтобы на итерации t, получив входное изображение x_{t-1}, сгенерировать изображение x_t, необходимо, чтобы диффузионная сеть предсказала шум $\epsilon(x_t, t)$, который присутствует на изображении. Во время этого предсказания у модели должны считаться и сохраняться активации $F(w|c,t)$ выбранных внутренних слоев, которые используются MLP. На их основе перцептрон генерирует соответствующие латентное представление эскиза E˜, вычисляется мера похожести полученного эскиза E˜ на истинный e: L(E˜, E(e)) = ||E˜ − E(e)||^2, а затем считается соответствующий антиградиент $-\nabla_{x_t}\mathrm{L}$. Тогда новое изображение на выходе будет рассчитываться как:

$$\widetilde{x_{t-1}} - 1 = x_{t-1} - \alpha * \nabla_{x_t}\mathrm{L}$$

Где:

α – коэффициент, который задает степень похожести итогового изображения на эскиз. Эмпирически было выяснено, что лучше себя показывает нормированный коэффициент по формуле:

$$\alpha = \frac{\|x_t - x_{t-1}\|_2}{\|\nabla_{x_t}\mathrm{L}\|_2}\,\beta$$

В этом случае коэффициентом выступает уже β, который будет иметь порядок $O(1)$.

Таким образом, диффузионная модель будет стараться соблюдать указанные границы эскиза при генерации. Исходной моделью до модификации выступает предобученная latent diffusion model (Stable Diffusion).

1.2 Вывод

В этой главе было проведено исследование нескольких существующих подходов для решения задачи генерации реалистичного изображения на основе скетча. На основе сравнения результатов можно сделать ряд выводов. Во-первых, можно заметить, что качество изображений, полученных диффузионными моделями, сильно превышает качество изображений, сгенерированных моделями GAN архитектуры. Поэтому будущее решение будет основываться на архитектуре диффузионной модели. Во-вторых, решение 1.1.2.5 представляется более предпочтительным по сравнению с решением 1.1.2.4, так как кажется более простым и интересным подходом. При этом, это самое новое исследование, у него до сих пор не опубликован исходный код и недоступны веса обученной модели. В работе описаны хорошие полученные результаты, а также решение удовлетворяет сформулированным требованиям, а именно – существует механизм контроля степени соответствия эскиз-реализм. Обучение модели занимает достаточно мало времени, так как используется предобученная модель диффузии, и необходимо обучить только вспомогательную MLP сеть.

Таким образом, решение предлагается строить на основе архитектуры, описанной в исследовании [https://arxiv.org/pdf/2211.13752v1.pdf].

Глава 2 Разработка методов и проектирование системы

В данной главе будет описана подробнее предлагаемая архитектура будущей системы.

2.1 Проектирование системы в рамках создания телеграм бота

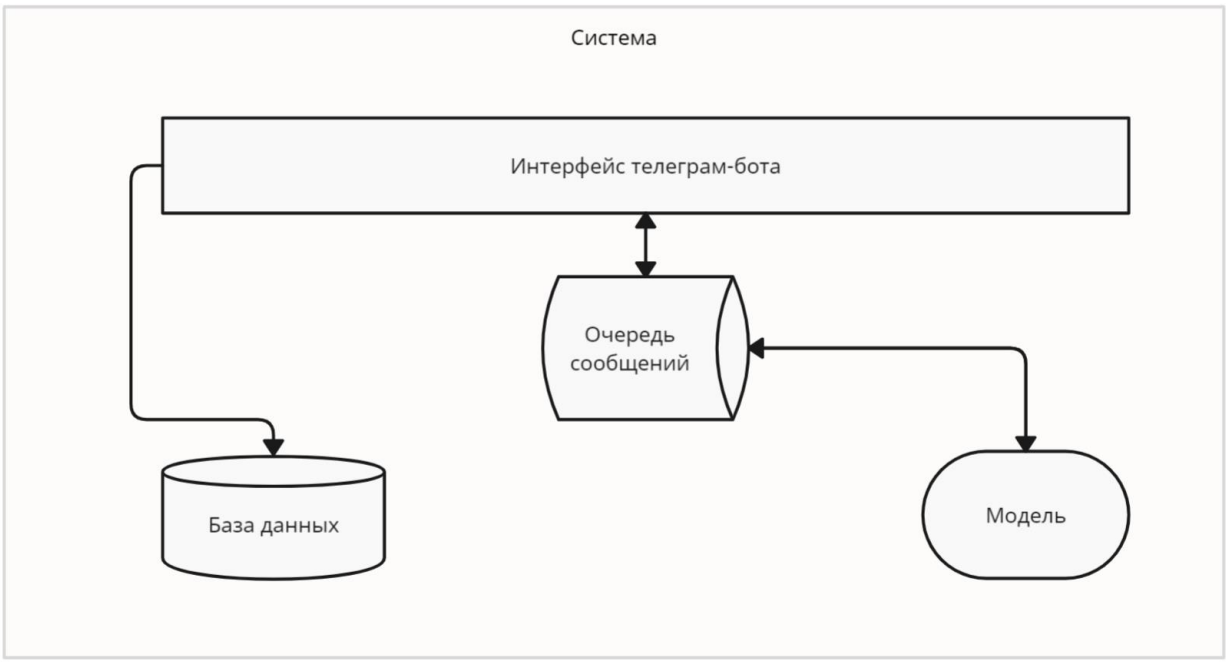

Рис. Схема компонент системы

Система будет состоять из нескольких больших компонентов, взаимодействующих между собой. Основным компонентом будет выступать интерфейс бота, представляющий собой своеобразную «обертку» для взаимодействия с клиентами. При общении с ботом все запросы пользователей в первую очередь будут попадать именно в этот компонент. Он будет выполнять функции авторизации пользователей, регистрирование задач, формирование ответов. В качестве вспомогательного компонента будет выступать база данных, которая будет хранить необходимую дополнительную информацию, к примеру всех зарегистрированных в системе пользователей, историю общения с ними, историю и детали их запросов. Для реализации взаимодействия с моделью между ботом и моделью в качестве связующего звена будет выступать очередь сообщений, представленная на схеме отдельным компонентом. Все задачи на генерацию данных моделью будут помещаться в нее и выполняться моделью по мере работы. Такая реализация необходима, для возможности оптимально распределять нагрузку, и легко масштабировать систему при необходимости.

Глава 3 Программная реализация системы генерации и проведение экспериментов

В данной главе будет описан подробнее процесс реализации описанного метода для генерации изображения по переданному скетчу, проведенные эксперименты с моделью, а также детали реализации всей системы в целом.

3.1 Используемая модель диффузии

В качестве основной диффузионной модели, на которой будет строиться генерация изображений, была выбрана Stable Diffusion v1-5 [https://huggingface.co/runwayml/stable-diffusion-v1-5]. Это latent diffusion модель, в которой используется предварительно обученный кодировщик текста (CLIP ViT-L/14).

3.2 Системные требования

Для локальной работы модель требует достаточно много видеопамяти. В процессе локального запуска на собственном оборудовании (NVIDIA GeForce RTX 3070, 8GB) модели не хватило памяти. Поэтому был арендован специальный сервер с более современными видеокартами, обладающими большим объемом памяти. Можно сказать, что минимальные требования для модели составляют около 12GB видеопамяти для запуска и генерации.

Таким образом, работа с моделью осуществлялась на машине с 1 видеокартой RTX 3090 24 GB VRAM.

3.3 Обучающая выборка

Во всех экспериментах использовался датасет Imagenet [https://image-net.org/static_files/papers/imagenet_cvpr09.pdf] с именами классов в качестве текстового описания. Соответствующие изображения эскизов были сгенерированы с помощью модели предсказания краев из [https://arxiv.org/abs/2108.07009], а затем дискретизированы с порогом 0,5.

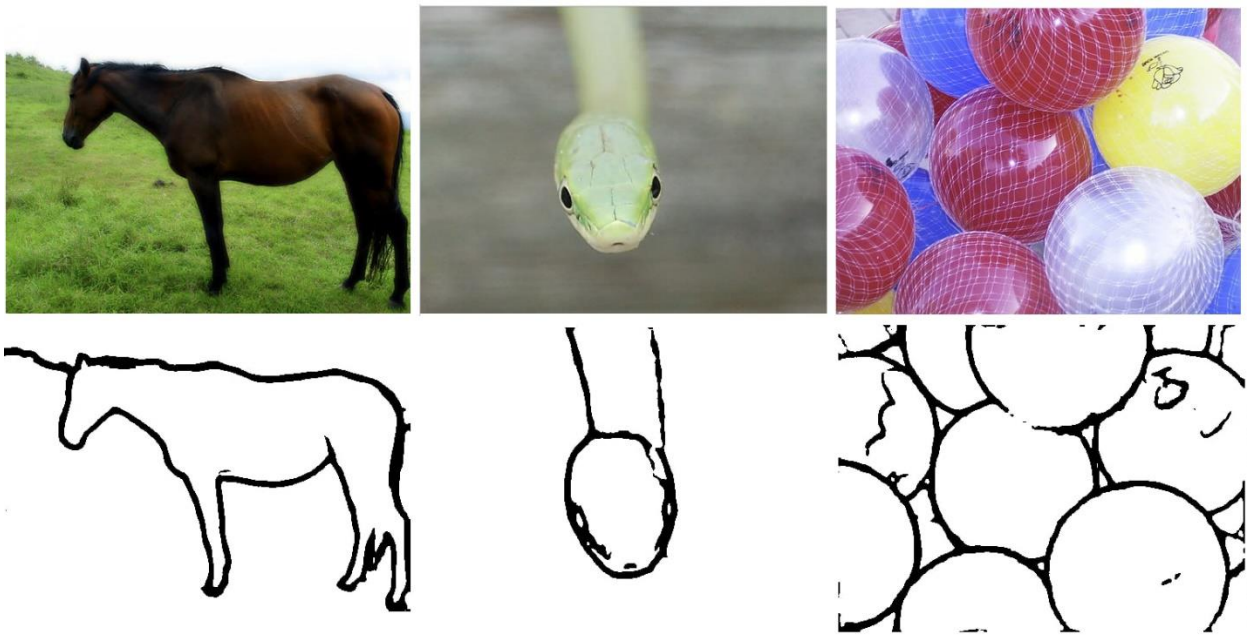

Рис. Примеры из обучающей выборки. Сверху оригинальное изображение из датасета Imagenet, внизу – соответствующее обработанное изображение-эскиз

Обучение проходило на подвыборке из 40 тысяч изображений разных классов (300 классов, по 100-200 изображений на каждый класс). В процессе обучения на вход модели подавалось оригинальное изображение, зашумленное до итерации t, а также текстовое описание, которое представляло собой название класса, к которому принадлежит изображение в датасете Imagenet.

3.4 Построение baseline решения

Baseline решением будет считать то, которое описано в статье [] и в первой главе работы. Решение строилось только на основе опубликованной статьи, так как ни код, ни модель авторы пока не опубликовали.

3.4.1 Построение и обучение MLP

Описанная MLP модель представляет собой guidance модель, которая будет «направлять» (регулировать/контролировать) процесс генерации диффузионной модели. На вход MLP принимает активации с 9 слоев диффузионной модели. Это 3 слоя с input блока модели (2-ой, 4-ый и 8-ой слои), также 3 слоя с middle блока

модели (0-ой, 1-ый и 2-ой слои) и 3 слоя с output блока (также 2-ой, 4-ый и 8-ой слои). Эксперименты с использованием различных слоев, отличных от перечисленных, не принесли значимых улучшений, поэтому в дальнейшем будет описана только перечисленная комбинация.

Так как описанные слои имеют различную размерность, то их необходимо отмасштабировать. Все слои были увеличены до размера референса с помощью билинейной интерполяции и затем объединены вместе вдоль размерности своих каналов. Таким образом, имеем область размера 64*64, представленную в нескольких каналах, собранных с разных слоев диффузионной модели. Дополнительно на вход MLP подается временная метка, которая хранит в себе номер итерации генерации изображения, а также positional encoding, который представляется как sin(2* pi * t * 2^(-l)), где l = 0,…,9. Таким образом размерность входных данных MLP составляет 9280 канала для 64*64 изображения. На выходе от модели ожидается тензор, совпадающий по размерности с латентным представлением скетч-изображения (4 канала для 64*64 изображения).

Обучение MLP проводилось в течение часа на A-100 видеокарте на описанном ранее наборе данных. После работа сети была протестирована на нескольких изображениях (см результаты на рисунке ниже).

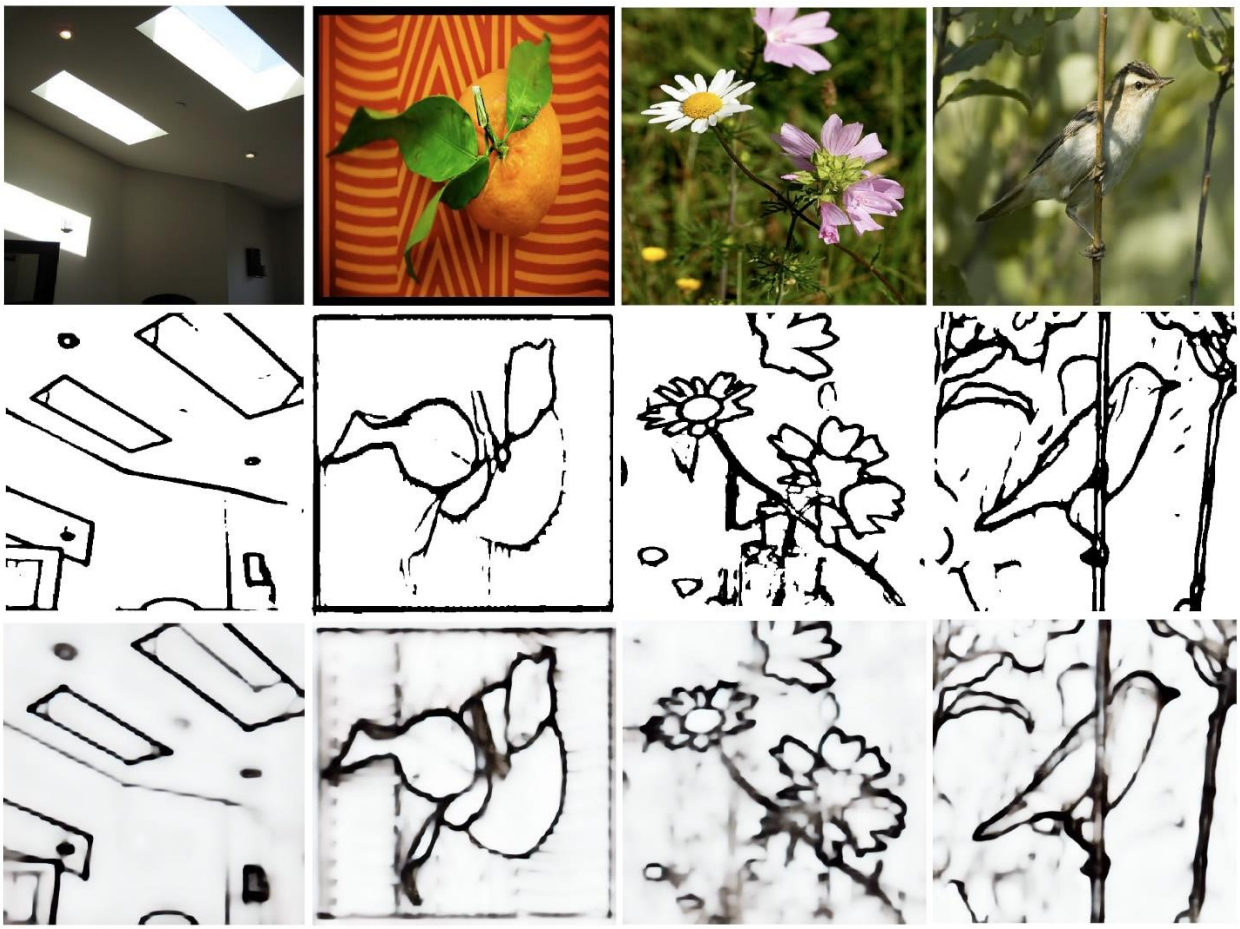

Рис. Исходное изображение (верхний ряд), соответствующий полученный по нему скетч из обучающего датасета (средний ряд) и скетч, полученный от MLP (нижний ряд)

Можно сказать, что получено достаточно хорошее качество работы MLP. В некоторых случаях предсказанный скетч оказывается даже лучше, чем скетч из обучающей выборки (например, для изображения мандарина по центру – скетч из датасета обрезает верхнюю границу фрукта, в то время как MLP выделяет это границу, что является более корректным поведением).

3.4.2 Результаты

Для тестирования полученной модели использовались как скетчи из датасета, так и несколько тестовых изображений из датасета SketchyCOCO [https://sysu-imsl.github.io/EdgeGAN/index.html].

Были перебраны различные комбинации параметров реалистичности betta и S (доля итераций, на которых будет использоваться вспомогательная guidance модель). В результате чего были обнаружены некоторые описанные ниже закономерности.

Для большого значения betta и S=t (вспомогательная guidance модель применяется на всех итерациях генерации) генерируемое изображение получается полностью соответствующим скетчу и теряет всякий реализм. Некоторые примеры представлены на рисунках ниже.

Рис. Примеры генерации при разных параметрах betta и S

Скетч, на основе которого проходила генерация, представлен на рисунке в первой колонке (черно-белый). Полученные скетчи представлены в остальных колонках. Можно заметить, как при увеличении betta становятся более явными линии границ эскиза на итоговом изображении. При этом при уменьшении betta границы скетча все менее и менее заметны, но изображение становится более наполненным. Появляется фон и детали, изображение становится более реалистичным.

Однако можно заметить, что несмотря на наличие на изображении четких границ скетча, основной генерируемый контент изображения все равно не совпадает с этими границами. Рассмотрим эту проблему подробнее на примере велосипеда. Соответствующие сэмплы представлены в нижнем ряду рисунка. От диффузионной модели требовалось сгенерировать фотографию красного велосипеда. Можно видеть, что на итоговых генерациях действительно присутствуют красные «пятна»,

однако они не совпадают с видимыми границами скетча. При значениях betta меньше 1.7 заметно, что красный велосипед генерируется в другой части изображения, но при этом его границы как будто размываются. Словно сеть начинает рисовать велосипед независимо от скетча, а потом на уже достаточно больших итерациях, когда основной контент изображения уже заложен, начинает прорисовываться скетч. Однако его границы уже сильно далеки от основных элементов изображения и модель уже не способна это исправить.

Можно также рассмотреть генерацию при заданном значении betta = 1.9 и различном значении S. Примеры генерации представлены на рисунке ниже.

Рис. Примеры генерации при разном параметре S

Видно, что при S <= 0.6 * t теряется видимость скетча. При этом независимо от значения S также наблюдается несогласованность положения видимого красного велосипеда на изображении с видимыми границами скетча. При S = 0.5 * t скетч уже становится совсем не заметен и не вносит никакого вклада в итоговое изображение. Конечно, это граничное значение может меняться при изменении betta (при увеличении betta скетч перестанет быть заметен уже не при S=0.5 * t, а при меньшем значении). Однако приведенные цифры все еще достаточно интересны, так как baseline изначально проектировался с оптимальными параметрами betta = 1.6 и S = 0.5 * t, как это было описано в исследовании.

3.5 Проведение экспериментов

3.5.1 Изменение стартовой точки

На основе описанной в предыдущем подпункте проблемы можно выдвинуть гипотезу, объясняющую возникновение этой проблемы. Заметим, что генерация основной сцены изображения происходит на самых начальных итерациях. Чем

больше итерация, тем больше прорабатываются детали и меньше закладывается основной контент. Одновременно с этим заметим, что на самых начальных итерациях MLP хуже всего способен выявлять границы будущего скетча, так как большая часть изображения состоит пока преимущественно из шума (или только из него).

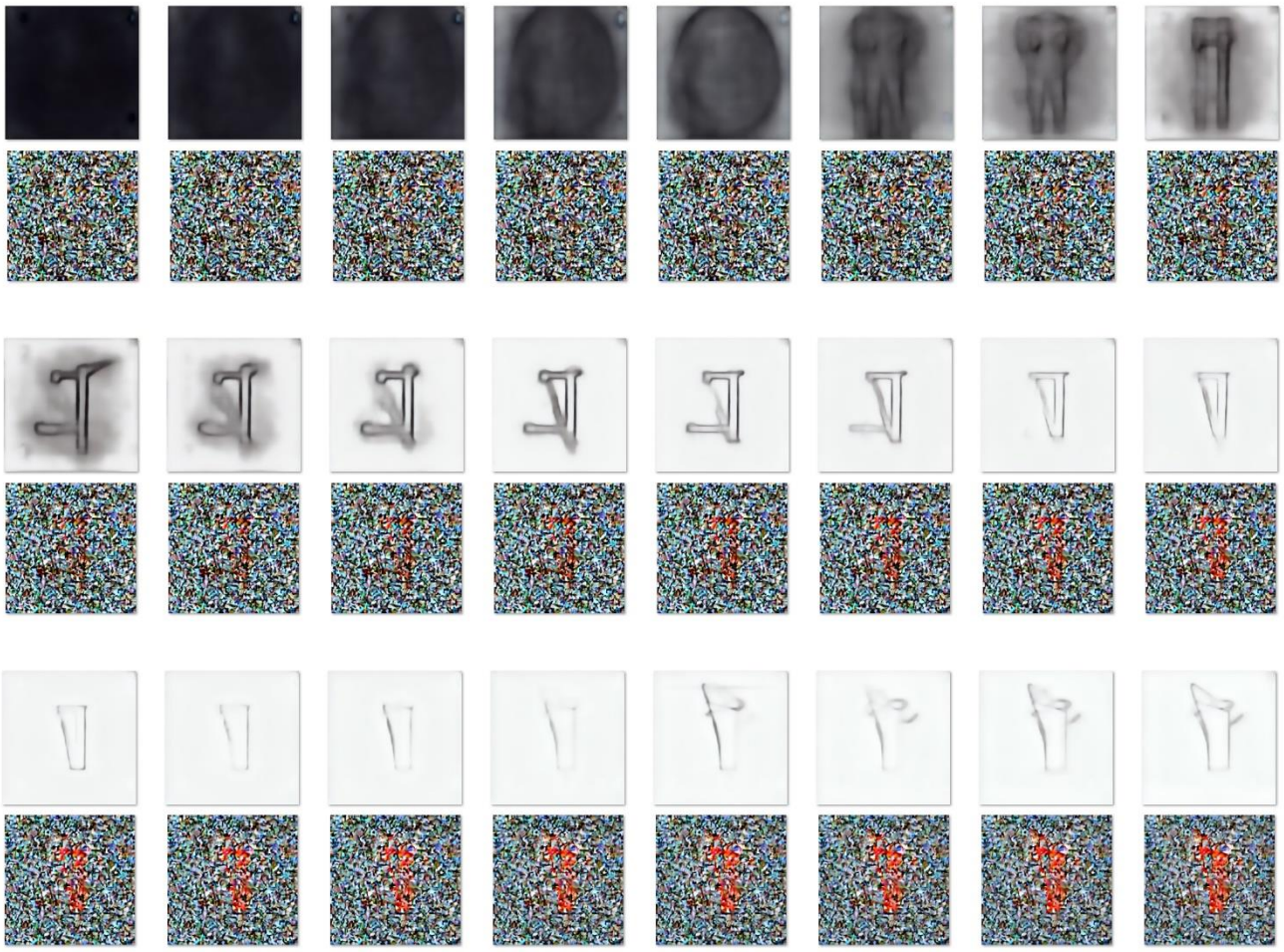

Рис. Первые 24 итерации генерации изображения. Слева-направо, сверху-вниз порядок увеличения итераций. Нижний ряд – генерируемое изображение, верхний ряд – полученный с помощью MLP скетч на соответствующей итерации

Таким образом, проблема заключается в том, что MLP хуже всего работает тогда, когда он больше всего нужен. При этом дополнительное дообучение MLP на начальных итерациях не дает никаких значимых результатов.

В качестве решения был использован самый простой метод – добавление на первой итерации к шуму, подающемуся на вход диффузионной модели, предварительно смешанный с шумом входной скетч в некотором соотношении. Результаты представлены на рисунке ниже.

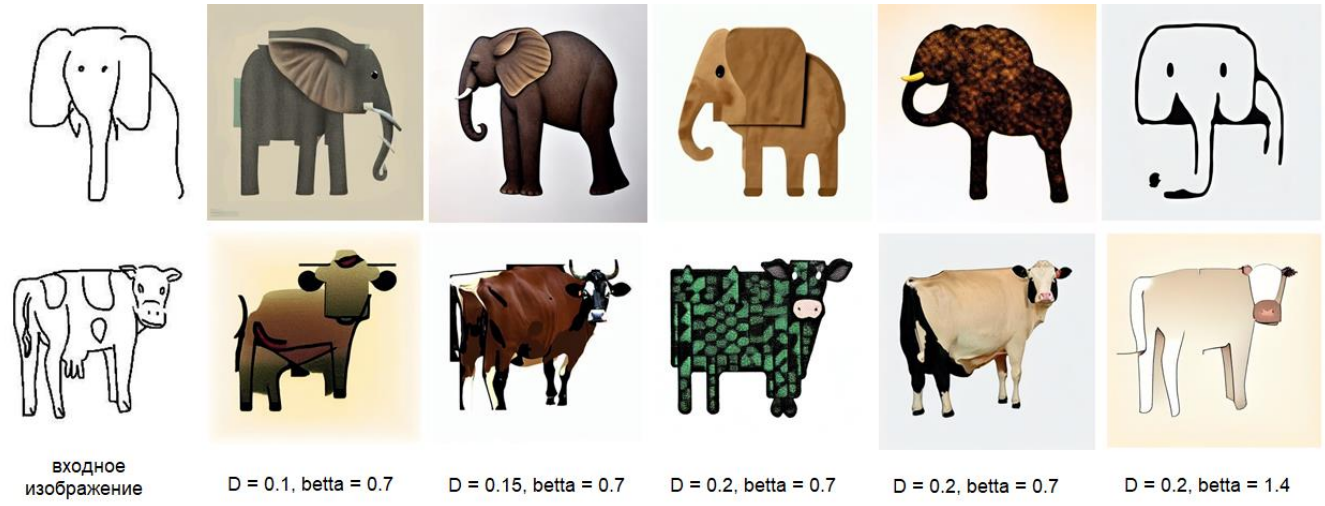

Рис. Примеры генерации при разном параметре D и betta

Введем дополнительный параметр D, отвечающий за смешивание шума и входного скетча на первой итерации:

$$x_t = (1 - D) * noize + D * sketch$$

Тогда получим, что чем больше параметр D, тем сильнее должно быть влияние скетча на итоговое изображение. Примеры, приведенные выше, получены при разных D и betta, однако можно заметить, что при увеличении значения D фон изображения становится белым (аналогично фону скетча). Этот недостаток был описан также в исследовании [https://arxiv.org/pdf/2211.13752v1.pdf] при сравнении работы с моделью SDEdit [Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, JunYan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. arXiv preprint arXiv:2108.01073, 2021], которая работает схожим образом – в определенных пропорциях смешивает скетч и шум, а затем подает на вход диффузионной модели.

Заметим, что полученные результаты сильно превосходят предыдущие, когда генерация строилась только на MLP. Таким образом, мы улучшили результат baseline модели.

3.5.2 Отказ от MLP

Предположим, что полученное неплохое качество обусловлено только инициализацией, в то время как использование MLP только портит модель. Попробуем исключить перцептрон и использовать для соответствия скетчу только добавление его к шуму на первой итерации. Полученные результаты представлены ниже.

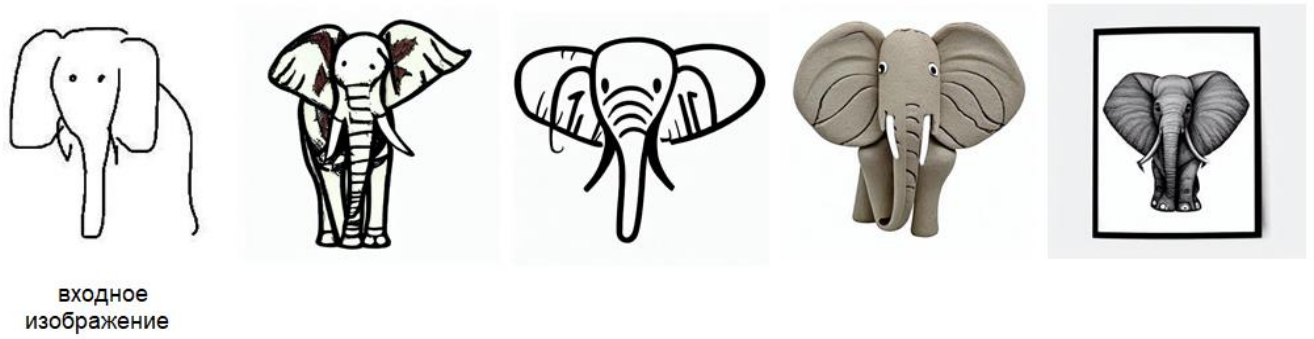

Рис. Примеры генерации без использования MLP

Видим, что продолжает преимущественно белый фон. При этом сами границы генерируемых объектов сильно отдалились от границ скетча. Больше всего это заметно по ушам слона, которые теперь стали расходиться сильно шире и становиться более круглыми, выходя за границы скетча, где они были более квадратной формы.

Таким образом, использование MLP в качестве guidance predictor позволяло лучше следить за следованием границам, чем при отсутствии перцептрона, когда от границ остается только намек, заложенный на самой первой итерации и деформированный на всех последующих.