

# Отчет по курсовой работе на тему: “StyleCLIP”

Deep Learning School при МФТИ  
1 семестр весна 2024  
Трофимов Антон  
12.07.2024

# Введение

В современном мире всё большую значимость приобретают модели искусственного интеллекта. Они используются в разнообразных сферах деятельности человека, помогая ему решать задачи, недоступные к решению алгоритмическим методом. Одной из таких задач является редактирование изображений по текстовому описанию.

## Выбор темы

Выбор данной темы обусловлен стремлением к более глубокому пониманию принципов редактирования и генерации изображений с помощью StyleGAN-ов.

## Цель работы

Реализация StyleCLIP на базе StyleGAN 3 для редактирования изображений с помощью текстовых описаний двумя способами:

- Optimization - оптимизация вектора в латентном пространстве stylegan
- Latent Mapper - обучение модели, преобразующей латентный вектор таким образом, чтобы он соответствовал текстовому описанию

Реализация редактирования реальных изображений

- С помощью StyleGAN3 Encoder
- С помощью оптимизации и зашумления латентного вектора

# Теоретическая часть

## Описание моделей

1. **StyleGAN3** (Style Generative Adversarial Network 3) - генеративная модель от NVidia, позволяющая создавать высококачественные изображения с контролем над стилями и содержимым изображения. StyleGAN состоит из двух моделей:
  - **Mapper** - модель, преобразующая вектор из латентного пространства Z в вектор w латентного пространства W, тем самым придавая будущему изображению стили
  - **Synthesis** - модель генератор, создающая из латентного вектора w конечное изображение, с учетом заданных стилей.
2. **CLIP** (Contrastive Language–Image Pre-Training) - модель от OpenAI, позволяющая сравнивать и устанавливать соответствие между текстовыми описаниями и изображениями. В данной работе используется для вычисления clip loss - потерь, связанных с несоответствием изображения и текста.
3. **Arcface** - модель, позволяющая определить схожесть между двумя изображениями с лицами людей. В данной работе используется в качестве регуляризации, для вычисления id loss - потерь, связанных с различиями между исходным и обработанным изображениями
4. **StyleGAN 3 Encoder** - модель, предоставляющая возможность преобразовать реальное изображение в вектор из латентного пространства W StyleGAN 3. Используется для изменения реальных изображений.
5. **VGG16** - модель для извлечения признаков из изображений. В данной работе используется для нахождения латентного вектора через оптимизацию среднего латентного вектора.

## Optimization

Оптимизационный подход в StyleCLIP предполагает изменение латентных векторов (w/z) StyleGAN таким образом, чтобы изображения, генерируемые моделью, соответствовали текстовым описаниям. Это достигается путём минимизации функции потерь, которая связывает изображения и текстовые описания через модель CLIP.

Основные шаги оптимизационного подхода

1. Генерация начального изображения:
  - a. Выбирается случайный латентный вектор  $z$  из латентного пространства StyleGAN, из которого генерируется style latent vector  $w_s \in W$
  - b. Генерируется начальное изображение  $I = G(w)$  с помощью StyleGAN на основе этого вектора.
2. Извлечение эмбеддингов:
  - a. Изображение  $I$  пропускается через модель CLIP для получения эмбеддинга  $E(image)$
  - b. Текстовое описание  $t$  пропускается через текстовую часть модели CLIP для получения текстового эмбеддинга  $E(text)$
3. Функция потерь складывается из нескольких основных составляющих
  - a. CLIP loss - косинусное расстояние между clip эмбеддингом изображения и текстовым эмбеддингом:

$$L_{clip}(G(w), t) = 1 - \cos(E_{img}, E_{txt})$$

- b. L2 лосс, не позволяющий оптимизированному вектору в значительной степени отклониться от стартового вектора оптимизации, данный лосс берётся с весовым коэффициентом  $\lambda_{L2}$

$$L_{L2} = \lambda_{L2} * \|w - w_s\|$$

- c. Регуляризация на ID с коэффициентом  $\lambda_{ID}$

$$L_{ID} = 1 - \langle R(G(w_s)), R(G(w)) \rangle$$

где  $R$  - предобученная модель Arcface

d. Итоговый лосс:

$$\text{i. } L = L_{clip} + \lambda_{L2} * L_{L2} + \lambda * L_{ID}$$

## Latent mapper

Этот подход предполагает обучение модели, которая преобразует латентные векторы в пространстве StyleGAN таким образом, чтобы полученные изображения соответствовали текстовым описаниям, заданным пользователем.

Основные шаги Latent Mapper подхода

### 1. Обучение Latent Mapper:

Latent Mapper представляет собой нейронную сеть, которая принимает на вход исходный латентный вектор и текстовое описание и преобразует этот вектор так, чтобы результирующее изображение соответствовало текстовому описанию.

### 2. Архитектура

Архитектура latent mapper представлена ниже на изображении.

Известно, что разные слои StyleGAN отвечают за разные уровни детализации в сгенерированном изображении. Поэтому слои latent mapper делятся на три группы (coarse, medium, fine).

Обозначив латентный код входного изображения как  $w = (w_{coarse}, w_{medium}, w_{fine})$  маппер определяется следующим образом:

$$M_t(w) = (M^{coarse}_t(w_{coarse}), (M^{medium}_t(w_{medium}), (M^{fine}_t(w_{fine}))$$

### 3. Loss функция

- a. CLIP loss - косинусное расстояние между clip эмбеддингом изображения и текстовым эмбеддингом:

$$L_{clip}(G(w + M_t(w)), t) = 1 - \cos(E_{img}, E_{txt})$$

- b. L2 лосс, не позволяющий уйти далеко от стартового вектора, данный лосс берётся с весовым коэффициентом  $\lambda_{L2}$

$$L_{L2} = \lambda_{L2} * \|M_t(w)\|$$

- c. Регуляризация на ID с коэффициентом  $\lambda_{ID}$

$$L_{ID} = 1 - \langle R(G(w_s)), R(G(w)) \rangle$$

где R - предобученная модель Arcface

- d. Итоговый лосс:

$$L = L_{clip} + \lambda_{L2} * L_{L2} + \lambda * L_{ID}$$

## Global Directions

Ещё один подход, который не будет подробно рассмотрен в данной работе по причине недостатка времени на его реализацию. Однако упомянуть его в отчете будет не лишним.

Если оптимизация работает для одного единственного изображения и одного промпта, латентный маппер для  $n$  изображений и одного промпта, то Global directions работает для  $n$  изображений и  $m$  промптов.

Основная идея Global directions заключается в том, чтобы выделить в латентном пространстве stylegan-а направления, соответствующие определенным признакам, заданным заранее и научиться двигать латентный вектор к нужным признакам.

Таким образом, на старте задается  $m$  текстовых описаний, например, “sad”, “happy”, “surprised”, “old”, “woman” и так далее, а затем реализуется модель, которая находит направления для каждого из заданных описаний и после обучения, при получении латентного вектора она сдвигает его в нужном направлении с заданным коэффициентом.

# Практическая часть

## Optimization

Этот способ изменения изображений с помощью текстового описания включает следующие шаги:

1. Создается исходный вектор  $z$  латентного пространства  $Z$  модели StyleGAN. Достаточно сгенерировать вектор заданной размерности с нормальным распределением.
2. Вектор  $z$  пропускается через `stylegan.mapper` для получения style-вектора  $w$
3. Вектор  $w$  пропускается через `stylegan.synthesis` для генерации изображения
4. Сгенерированное изображение и векторы фиксируются, после чего начинается непосредственно оптимизация. В StyleGAN3 можно оптимизировать либо вектор  $z$ , либо вектор  $w$ . Оба варианта рассмотрены в работе в `optimization.ipynb`, в этой же статье будет рассмотрен вариант с оптимизацией  $w$ . Стоит отметить, что в StyleGAN2 было 3 латентных пространства:  $Z$ ,  $W$ ,  $W^+$  и  $W^+$  StyleGAN 2 соответствует  $W$  StyleGAN 3.
5. Задается текстовое описание изменений, которые необходимо применить к исходному изображению, после чего с помощью модели `clip` из него извлекаются признаки.
6. Вектор  $w$  клонируется в вектор  $w_{opt}$ , у которого включается расчет градиентов и который помещается в оптимизатор.
7. Циклично на каждом шаге из вектора  $w_{opt}$  генерируется изображение `image_opt` и считаются 4 лосса:
  - a. Clip Loss с помощью модели `clip`, в которую передается признаки, полученные в этой же модели из `image_opt` и признаки, полученные из текстового описания.
  - b. L2 Loss - считается MSE между  $w$  и  $w_{opt}$ , чтобы оптимизируемый вектор не слишком сильно отклонялся от исходного
  - c. ID Loss - получается при пропускании исходного и сгенерированного на итерации изображений через модель `arcface`

- d. Loss - итоговый лосс, который является суммой вышеперечисленных лоссов взятых с коэффициентами(гиперпараметрами)  $\lambda_{L2}$ ,  $\lambda_{ID}$
- 8. Затем накопленная ошибка пропускается через вектор, меняя его значения в нужную сторону
- 9. Спустя порядка 100 итераций получаются неплохие результаты, для более стабильного результата достаточно провести ~300 итераций

## Latent Mapper

### Архитектура

Реализация модели Latent Mapper, в данном случае модель состоит из 3-ех групп слоев  $L_{coarse}$ ,  $L_{medium}$ ,  $L_{fine}$ , которые отвечают за соответствующие признаки в латентном пространстве. Каждая из групп слоев, в свою очередь, состоит из:

- PixelNorm для нормализации
- 4 полносвязных слоя с функциями активации leaky\_relu

Также, при создании маппера можно зафиксировать часть слоев, чтобы они не изменялись. Например, часто бывает полезно отключить изменение coarse слоев, чтобы сохранить основные черты лица исходного изображения.

### Обучение

1. Выбирается текстовое описание того, как должно будет измениться изображение, из него извлекаются признаки с помощью модели CLIP
2. Также нужен датасет с изображениями, но в данном случае в качестве него будут использоваться изображения, сгенерированные StyleGAN, в целях избежания необходимости инвертировать каждое изображение в латентное пространство.
3. На каждой эпохе генерируется батч изображений, латентные векторы в которых прогоняются через Latent Mapper с получением делт

4. Данные дельты, взятые с коэффициентом  $\lambda_{delta}$ , складываются с исходным вектором w и на основе этого нового вектора генерируется стилизованное изображение
5. Подобно варианту с оптимизацией, рассчитываются лоссы, градиент от которых распространяется back propagation-ом, и затем веса Latent Mapper обновляются.
6. Спустя порядка 20 эпох по 100 изображений в батче получаются первые результаты, а спустя 100 эпох результат довольно стабилен. В качестве оптимизатора выступает AdamW с начальным learning rate, а в качестве scheduler-a ReduceOnPlateau с factor=0.2, patience=10

## Редактирование реальных изображений

Сложность редактирования реального изображения с помощью StyleGAN-ов заключена в необходимости получения латентного вектора реального изображения, так как после получения вектора, изменение изображения ничем не отличается от редактирования сгенерированных изображений. Для решения этой задачи есть 2 подхода:

### StyleGAN3 Encoder

Суть данного метода достаточно проста: необходимо скачать предобученную модель энкодера для StyleGAN 3, которая преобразует реальное изображение в латентное пространство и отдает латентный вектор.

### Optimization

Метод очень схож с оптимизацией латентного вектора для редактирования изображения, однако в этом случае выбирается средний латентный вектор w и на каждой итерации оптимизации к нему подмешиваются шумы, после чего генерируется изображение и вычисляются лоссы:

- Distination loss - потери связанные с удаленностью исходного изображения относительно сгенерированного. Потери вычисляются с помощью VGG16 - модели, которая извлекает из обоих изображений их признаки, позволяя сравнить их между собой.

- Regularization loss - лосс для регуляризации шумов, чтобы вектор не слишком сильно отклонялся от среднего

На каждом шаге оптимизируется как сам вектор, так и шумы, добавляемые к нему.

### Сравнение подходов

Оригинал	StyleGAN 3 Encoder	Optimization
		

Как можно видеть, StyleGAN Encoder лучше и быстрее справляется со своей задачей. В оптимизационном подходе удается лишь найти приблизительные черты лица. Помимо этого, в данной работе не удалось использовать латентный вектор, полученный с помощью оптимизации, для редактирования изображений с помощью латентного маппера, так как на выходе получается шум. Это связано с тем, что его значения выходят из диапазон  $[-1;1]$ , однако при нормализации вектора, как во время оптимизации, так и после ее завершения, восстановленное изображение приобретает черты, которых в ненормализованном векторе не было.

# Результаты

## Optimization

### Prompt

Очень важной составляющей работы со stylegan-ами, как выяснилось ниже в эксперименте, является непосредственно текстовое описание. Часто возникают ситуации, когда одна и та же картинка, обработанная со схожими промптами дает совершенно разный результат. Вот яркий пример: Предположим, что мы хотим добавить человеку бороду. Ниже таблица с результатами, полученными с одинаковыми гиперпараметрами

Попытки отрастить бороду				
Оригинальное изображение	“beard”	“Big beard”	“lush beard”	
				

Только в последнем варианте “lush beard” удалось найти направление для оптимизации вектора.

### Гиперпараметры

Основными гиперпараметрами в обоих подходах являются  $\lambda_{L2}$ ,  $\lambda_{ID}$ , learning\_rate, n\_steps/n\_epochs. Прежде чем обсуждать их значения в обоих случаях, стоит отметить, что эмпирически сложилось ощущение, что для каждого текстового описания следует подбирать уникальные параметры, так как описания могут различаться. Однако, если попытаться

структурировать эмпирический опыт, то можно выдвинуть следующую гипотезу:

Гипотеза о коэффициентах регуляризации

Для начала разобьем множество промптов на 3 условных группы по сложности:

Сложность промпта	Описание	Примеры
Простая	Изменения, затрагивающие незначительную часть изображения и не меняющие основные черты исходного изображения	Эмоции, добавление простых элементов, таких как: очки, сигарета, сережки и т.д. Возрастные изменения, макияж.
Нормальная	Изменения значительной части элементов исходного изображения	Взаимодействие с волосяным покровом(прически. Борода и т.д.). Взаимодействие с цветами элементов.
Сложная	Изменения, перерабатывающие большую часть элементов исходного изображения	“шрек”, “аватар”, “джокер”, “gigachad”, “джин”, персонажи мультфильмов

Гипотеза состояла в следующем:

Чем проще промпт, тем выше можно выставлять коэффициенты регуляризации, например, для простых промптов коэффициенты лямбд могут находиться в диапазоне [0.05; 0.3] для лучшей регуляризации и схожести стилизованных изображений с исходными. Для средних необходимо снижать коэффициенты регуляризации для достижения результата до ~[0.1;0.01] и для сложных промптом коэффициенты регуляризации должны снизиться до [0.01; 0.001].

Однако, после проведения серии экспериментов, заключающихся в оптимизации латентного вектора для каждой группы сложности промптов с разными коэффициентами (были взяты промпты “sad”, “beard”, “joker”), выяснилось, что данная гипотеза не верна. Если в “sad” можно наблюдать соответствие гипотезе, то уже в “beard” она рушится, т.к. На промпте “beard” даже с нулевыми коэффициентами регуляризации оптимизатор не смог подобрать вектор. В то же время на промпте “lush beard” оптимизатор справился с задачей даже при  $\lambda_{L2} = 1$ ;  $\lambda_{ID} = 1$ , что свидетельствует о том, что под каждую задачу необходимо эмпирически подбирать в первую очередь соответствующий промпт, а уже затем гиперпараметры.

Ход эксперимента:

Простой промпт

Текстовое описание “sad” steps = 300 lr= 0.01		
Оригинальное изображение 	Шум с большим lr=0.1 	$\lambda_{L2}=0.3 \lambda_{ID}=0.3$ грусть почти не проявляется 
$\lambda_{L2}=0.1 \lambda_{ID}=0.1$ схожесть высокая, грусть видна 	$\lambda_{L2}=0.01 \lambda_{ID}=0.01$ грусть более ярко выражена, схожесть ниже 	$\lambda_{L2}=0.001 \lambda_{ID}=0.001$ более грустным чем с предыдущими параметрами сделать не удается 

Видно, что наиболее качественными для простого промпта изображения получились при  $\lambda_{L2} \in [0.1, 0.001]$

## Промпт средней сложности

Текстовое описание “lush beard” steps = 300 lr= 0.01		
Оригинальное изображение 	$\lambda_{L2} = 0.3 \quad \lambda_{ID} = 0.3$ 	$\lambda_{L2} = 0.5 \quad \lambda_{ID} = 0.5$ 
$\lambda_{L2} = 0.01 \quad \lambda_{ID} = 0.01$ 	$\lambda_{L2} = 0.001 \quad \lambda_{ID} = 0.001$ 	$\lambda_{L2} = 1 \quad \lambda_{ID} = 1$ 

В данном эксперименте все результаты достаточно похожи, однако наилучшими субъективно оказались  $\lambda_{L2} = 0.3 \quad \lambda_{ID} = 0.3$  и  $\lambda_{L2} = 0.001 \quad \lambda_{ID} = 0.001$ , что, как уже говорилось выше, опровергает исходную гипотезу.

## Алгоритм оптимизации

Исходя из вышеописанной провальной гипотезы, можно сделать вывод, что промпт, является важнейшим компонентом успеха в оптимизации. Поэтому можно описать такой алгоритм оптимизации:

1. Задать низкие гиперпараметры  $\lambda_{L2} = 0.001$ ;  $\lambda_{ID} = 0.001$  для того, чтобы регуляризационные лоссы не заглушали собой clip loss и выбрать промпт.
2. Провести оптимизацию и посмотреть, насколько результат соответствует желаемому. Обычно общий лосс в такой конфигурации должен прийти к значению меньшему 0.65 для удовлетворительного результата.
3. При неудовлетворительном результате изменить промпт на синонимичный. Например, если мы хотим сделать человека на исходном изображении лысым, то стоит протестировать следующие промпты “bald”, “hairless”, “completely bald”, “a bald as an egg”, “completely bald person” и так далее. Таким образом, необходимо повторять оптимизацию до момента, пока не появится желаемый результат изменения, или максимально близкий к нему.
4. Необходимо зафиксировать успешный промпт и начать увеличивать коэффициенты регуляризации для увеличения схожести с исходным изображением.

Пример:

Облысение		
Оригинал	“bald”	“hairless”
		
“completely bald”	“as bald as an egg”	“completely bald person”
		

Видно, что “completely bald” максимально близки к желаемому результату из прочей выборки.

Результат с повышенными до 0.5 коэффициентами регуляризации



Видно, что внешность стала ближе к оригиналу, как минимум, глаза уже не такие накрашенные, как было с меньшими коэффициентами.

Итоговые гиперпараметры для оптимизатора

Параметр	Значение
N steps	300
Learning rate	0.01
$\lambda_{L2}$	[0.001;0.7]
$\lambda_{ID}$	[0.001;0.7]
Scheduler StepLR step size	50
Scheduler StepLR gamma	0.5

Сравнение W-оптимизации и Z-оптимизации



В случае с простым промптом “smile” для W-оптимизации мужчина помимо улыбки ещё и состарился, в случае Z-оптимизации изменил национальность. Но для более сложных промптов Z-оптимизация работает хуже.

## Latent Mapper

С латентным маппером сложнее ставить эксперименты из-за длительности обучения, в частности, обучение одного маппера занимает порядка часа, в том время, как в оптимизационном методе ~5 минут.

Однако, качество, которое он показывает при правильной настройке - достаточно высокое.

За счет возможности фиксации определенной группы признаков (coarse/medium/fine) можно более тонко настраивать меняющиеся аспекты, а за счет  $\lambda_{\text{delta}}$  регулировать силу этих изменений.

## Отличия Latent Mapper от Optimization

Плюсы:

- Большая свобода настройки обучения из-за большего количества гиперпараметров
- Более высокое качество
- Меньшая зависимость от конкретного промтта
- Быстрая обработка изображений после обучения
- Лучше справляется со сложными промптами
- Возможность отключить изменение определенной группы признаков (coarse, medium, fine)
- Возможность, за счет изменения  $\lambda_{\text{delta}}$  регулировать силу изменений
- Возможность редактировать множество изображений, на одном обученном маппере

Минусы:

- Длительность обучения (в 15-20 раз дольше оптимизации)

## Гиперпараметры

Параметр	Диапазон значений	Подробности
Num epochs	[100;300]	Этого достаточно, первые результаты уже на 60 эпохе
Batch size	[20;100]	Чем сложнее промпт, тем больше стоит ставить батч
Learning rate	[0.1;0.5]	При низких значениях идет больше, а с ReduceOnPlateau 0.5 показывает неплохие результаты
$\lambda_{L2}$	[0.01;0.3]	В отличие от оптимизационного метода низкие значения не нужны
$\lambda_{ID}$	[0.005;0.05]	ID лямбду лучше выставлять чуть ниже L2
$\lambda_{delta}$	[0.1;1]	Коэффициент, на который умножается дельта, полученная из маппера
ReduceOnPlateau factor	[0.2;0.5]	Обычно 0.2 достаточно
ReduceOnPlateau patient	[5;20]	В зависимости это кол-ва эпох
Фиксация слоев маппера (coarse, fine, medium)	true/false	В зависимости от промпта

Разница результатов при обучении с разными дельтами



Благодаря  $\lambda_{delta}$  удается редактировать мощность изменений. Обычно 0.5 достаточно, а при  $\lambda_{delta} = 1$  для сложных промптов часто получаются зашумленные изображения.

Работа с реальными изображениями

В целом, качество становится чуть ниже только на этапе преобразования реального изображения в латентное пространство, далее всё работает, как обычно. Пример:

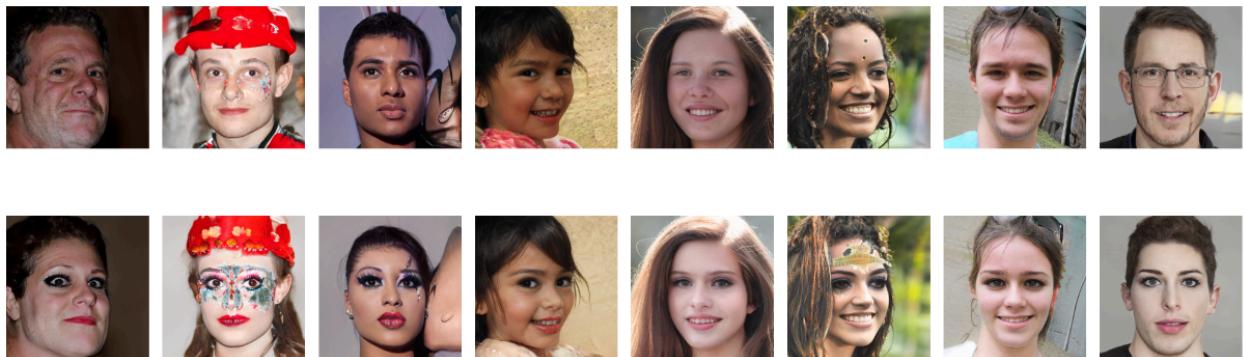


После энкодера на восстановленном изображении пропали погоны и изменилось выражение лица, но далее щетина нарисовалась вполне неплохо.

То же касается и оптимизационного подхода

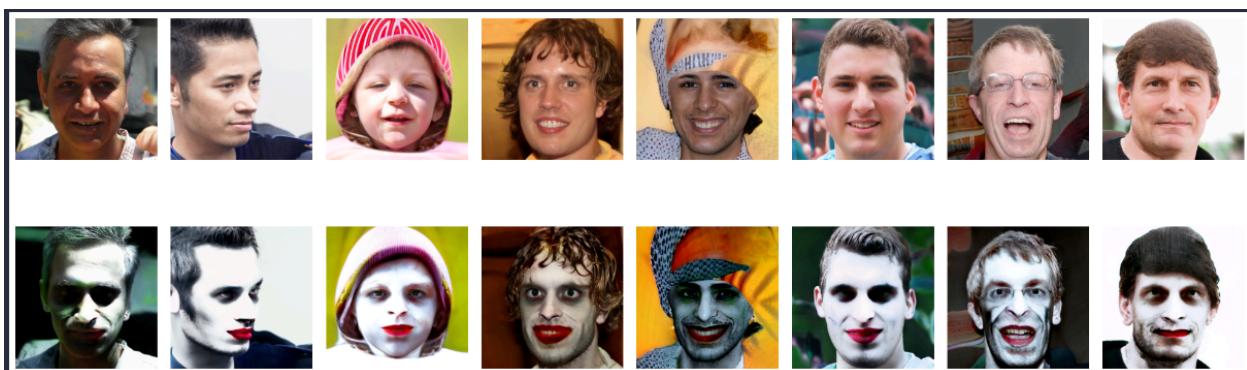
Примеры промптов

“makeup”



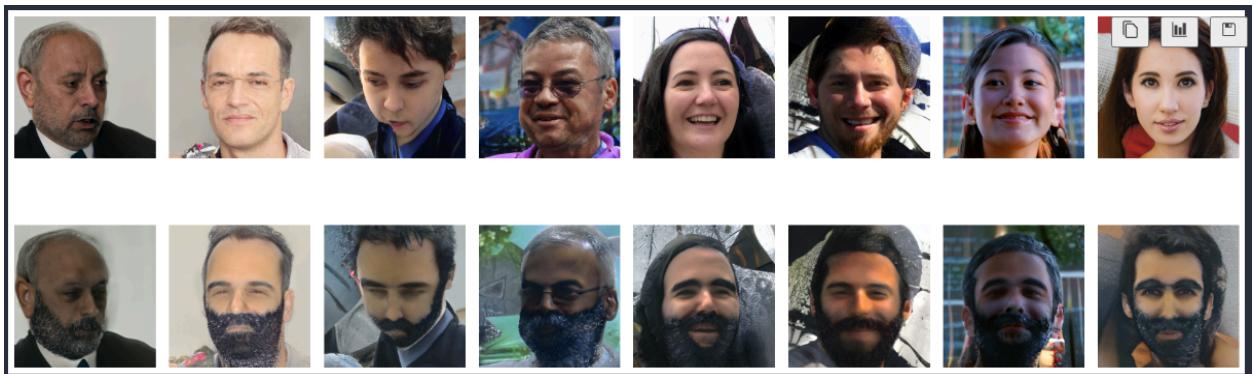
Видно, что латентный маппер неплохо стилизует изображения, на второй картинке слева превратив артефакты генерации в блёстки. Однако также он считает, что макияж - женская прерогатива и меняет пол мужчинам.

“joker”



Маппер также неплохо справляется с промптами высокой сложности, учитывая тени и делая изображение более зловещим, как на первой слева картинке

“big beard”



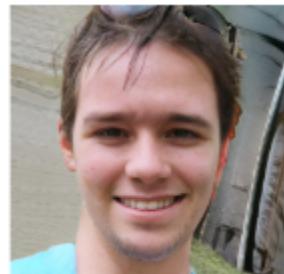
Борода выросла у всех, а вместе с ней у некоторых ещё и густые брови

## Забавные факты

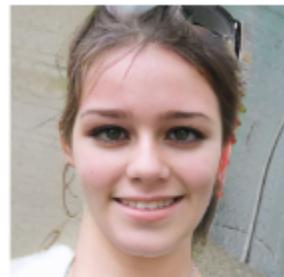
У StyleGAN есть проблемы с цветами, при любом промпте, где в словах присутствует цвет - красится всё подряд, а не только исходная деталь, особенно это видно в оптимизационном методе. Так, задав в промпте “blue hair”, на выходе можно спокойно получить аватара



Также, периодически возникают некоторые расистские веяния. В частности, при запросе “big beard”, помимо самой бороды появлялись пышные брови, внешность склонялась к арабской и... барабанная дробь... добавлялись красные злые глаза. Или, при попытке изменить прическу на ирокез, с промптом “mohawk” человек вместо этого превращался в индейца.



Также, при использовании промта, характерного для определенного пола - модель часто меняет пол исходных людей. Например, промпт “makeup” меняет пол на женский, а промпт “brutal”, напротив, на мужской.



## Вывод

### Optimization

В оптимизации результат сильно зависит от промпта и не всегда оптимизатору удается найти в латентном пространстве верное направление для изменения вектора. Поэтому необходимо подбирать несколько промптов с нулевой регуляризацией и затем, при получении удовлетворительного результата, увеличивать регуляризацию для повышения схожести.

### Latent Mapper

Маппер намного лучше справляется с задачей и является предпочтительным, он способен справляться со сложными промптами и показывать хороший результат.

Также, в будущем хочется попробовать Global Directions и сравнить качество с latent mapper-ом. Погрузиться в современные диффузионные сети, с несравненным качеством.

На этом всё, спасибо за внимание!