ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК 630

**Отчет об исследовательском проекте на тему:**

# Замена прически с помощью генеративных моделей GANs

**Выполнил студент:**

группы БПМИ211, 3 курса Панфилов Борис Сергеевич

**Принял руководитель проекта:**

Мещанинов Вячеслав Павлович
Штатный преподаватель
Факультет компьютерных наук НИУ ВШЭ

**Соруководитель:**

Аланов Айбек
Штатный преподаватель
Факультет компьютерных наук НИУ ВШЭ

**Консультант:**

Михаил Кузнецов
Младший исследователь
AIRI

Москва 2024

# Содержание

# 1 Аннотация

Перенос прически является уникальной и чрезвычайно непростой задачей из-за сложных взаимосвязей в освещении, геометрии и частичной окклюзии. С каждым годом становиться все больше различных методов для ее решения, тем не менее идеального метода все еще не существует. Задачей данного проекта является создание нового метода на основе уже существующих решений.

# Ключевые слова

Генеративные модели, GANs, StyleGan, перенос прически.

# 2 Введение

Прогресс в области генерации изображений высокого разрешения с помощью GAN'ов (Generative Adversarial Networks) [1, 2, 3] позволил применить их для семантического редактирования лиц [4, 5]. Одной из наиболее сложных и интересных тем в этой области является перенос прически [6]. Суть этой задачи заключается в передаче атрибутов прически, таких как цвет, форма и структура, с эталонной фотографии на входное изображение, сохраняя при этом лицо и фон входного изображения. Понимание взаимодействия этих атрибутов является ключом к качественному решению проблемы. Эта задача имеет множество применений в реальной жизни, например, в программах для редактирования лиц, виртуальной реальности или компьютерных играх. Существующие подходы, которые решают эту задачу, можно разделить на два типа: оптимизационные [7, 8, 9, 10], где сначала ищется, а затем оптимизируется представление изображения в латентном пространстве генератора, и кодирующие [11, 6, 12], где весь процесс переноса прически осуществляется с помощью одного прямого прохода через нейронную сеть. В данной работе будет проведено количественное и качественное исследование уже существующих методов, а также будет усовершенствован лучший, на сегодняшний день метод.

# 3 Обзор литературы

## 3.1 Формализация задачи

Пусть есть 3 изображения - face, shape, color. Задача переноса прически заключается в том, чтобы перенести прическу с shape и color изображений на face изображение. При этом форма полученной прически должна совпадать с формой прически на shape изображении, а цвет - с цветом волос на color изображение. При переносе прически на face изображение, остальные места изображения, например фон или лицо меняться не должны. Будем называть final изображением, изображение получившееся после переноса прически на face изображение.

## 3.2 Латентные пространства

Отправной точкой для развития моделей для семантического редактирования лиц, в частности переноса прически, стало изобретение генеративной модели StyleGAN [2] и ее модернизированной версии StyleGAN2 [3]. В этих моделях используется одинаковый принцип генерации изображения: берется вектор в латентном пространстве $Z$, переводиться в латентное пространство $W$, а затем из этого представления при помощи нейронной сети генерируется изображение. Преобразование из Z в W призвано решить проблему сильной корреляции признаков в изображениях, генерируемых напрямую из Z. В векторах W разные аспекты изображения, такие как поза, прическа, освещение, менее зависимы друг от друга, что позволяет более тонко регулировать процесс генерации изображений. Таким образом для того, чтобы сгенерировать изображение при помощи модели StyleGAN2 нужно лишь найти подходящее латентное представление.

Оказалось, что в пространстве $W$ признаки коррелируют несильно, но оно недостаточно богатое, то есть оно не способно содержать в себе все множество изображений. В частности это проявляется в том, что при декодировании изображений не удается восстановить мелкие детали лица, такие как морщины или родинки. Поэтому начали использовать латентное пространство W+ [6, 8]. Его можно использовать так же, как и латентное пространство W, но оно в 18 раз шире.

Далее рассмотрим методы, которые уже стали классическими в этой подобласти науки, а также sota метод, над улучшением которого мы будем работать.

## 3.3 Barbershop [10]

Первый метод для переноса прически основан на пошаговой оптимизации векторов в латентном пространстве. Авторы придумали и использовали новое латентное пространство (FS) для смешивания изображений, которое лучше сохраняет детали и кодирует пространственную информацию. F составляющая отвечает за форму и текстуру волос, а S - за цвет и стиль.

Поиск смешанного изображения в данном методе состоит из 4 частей. На первом этапе ищеться маска сегментации для final изображения. Для этого используется предобученная нейронная сеть BiSeNet [13]. Она сегментирует изображение на 19 классов. Мы указываем какие классы хотим использовать из face, shape и color изображений. Далее выбранные классы накладываются друг на друга и получается final маска сегментации.

Во второй части метода для каждого изображения ищется латентное представление $w \in W+$, такое, чтобы изображение полученное из него было максимально близко к исходному изображению. В дальнейшем мы сможем получить из этого представления представление в FS.

На третьем этапе для всех трех изображений последовательно решается 2 подзадачи: сначала ищется латентное представление в пространстве F, такое, чтобы изображение полученное из него было максимально близко к исходному изображению. Затем, в силу того, что оптимизировать представления в пространсве F вычислительно сложно, ищется представление $w \in W+$, такое, чтобы изображение полученное из него не только было максимально близко к исходному изображению, но и было выравненно соответсвенно final маске сегментации. Далее два полученных представления смешиваются, тем самым для каждого изображения мы получаем $F_{align}$ - выравненное представление для структуры.

И, наконец, на последнем, четвертом этапе раздельно смешаваются $F_{align}$ и $S$ представления для получения итогового латентного представления, из которого и будет получено final изображение.

Новое латентное пространство и остальные архитектурные решения позволили показать данному методу более хорошие как качественные, так и количественные результаты в сравнении с предшествующими методами.

## 3.4 HairCLIP [12]

Во второй статье описана идея переноса прически не только из shape и color изображений, но и из текстового описания. Представленный метод основан уже не на пошаговой

оптимизации, а на использовании предобученых encoder'ов. Как следствие он работает гораздо быстрее, чем алгоритмы построенные на оптимизации, но показывает не столь высокое качество.

Работа этого метода проще для понимания, потому что он не оптимизирует представление в латентном пространстве. Весь процесс генерации нового изображения представляет из себя один проход через нейронную сеть. Тем не менее этот проход через нейросеть все еще можно воспринимать как несколько этапов. В начале мы получаем embedding'и для исходного изображения и целевой информации при помощи моделей E4E[14] и CLIP[15] соответственно. Далее embedding'и целевой информации примешиваются при помощи слоев модуляции в embedding'и исходного изображения. Эти embedding'и мы и считаем итоговыми.

Работоспособность такого примитивного на первый взгляд метода обусловлена тем, что модель обучали на большом количестве изображений и использовали для этого новые эффективные функции потерь.

Данный метод впервые показал пример метода, который способен переносить прическу без оптимизаций, что сделало его сильно быстрее предшествующих методов. Тем не менее это ускорение повлекло ухудшение количественных оценок модели.

## 3.5 StyleYourHair [9]

Следующий метод переноса прически - StyleYourHair - использует, как и Barbershop, оптимизацию представлений в латентном пространсте. Авторы уделяют особое внимание тому, что их алгоритм гораздо лучше справляется с ситуациями, в которых люди находяться в разных позах на face и shape изображениях. Кроме того, авторы вводят новую функцию потерь, помогающую сохранять детали прически при поиске выравненного латентного представления.

Процесс получения итогового латентного представления довольно похож на Barbershop и также состоит из 4 этапов оптимизации.

В первой части мы ищем для каждого из трех (face, shape, color) изображений латентное представление $w \in W+$, такое, чтобы изображение полученное из него было максимально близко к исходному изображению. Делается это аналогично методу Barbershop при помощи алгоритма II2S[16]. Далее на этом шаге при помощи оптимизации находиться представления соответствующие $w$ в $FS$ пространстве.

На втором этапе ищется выравненное латентное представление для shape и color изоб-

ражения, посредством оптимизиции ранее найденного в $W+$ латентного представления. При оптимизации используется несколько лоссов с различными весами. Один из лоссов является изобретением авторов и позволяет сохранять детали прически при повороте изображения.

В третьей части происходит дорисовка окклюзий. Может произойти такая ситуация, что у face изобажения волос больше, чем у shape изображения. В такой ситуации при переносе прически могут образоваться места, которые требуется дорисовать. Раньше это делегировалось самому StyleGan2, но, очевидно, он в этом не идеален, потому что его обучали просто генерировать реалистичные изображения, а не дорисовывать уже существующие. Поэтому авторы решили сделать целый модуль, в котором при помощи масок сегментации и кросс-энтропийного лосса обучается латентное представление для появившихся из под волос face изображения участков, которые нужно дорисовать.

В четвертой части смешиваются все полученные представления.

Новая функция потерь для сохранения структуры прически при повороте и остальные архитектурные решения позволили показать данному методу более хорошие как качественные, так и количественные результаты в сравнении с предшествующими методами, основанными на оптимизации.

## 3.6 CtrlHair [11]

Это вторая статья, в которой описывается метод переноса прически, основанный не на оптимизации, а на использовании предобученных encoder'ов. Изюминкой этого метода является то, что он позволяет не только смешивать входные изображения, но и изменять различные параметры исходного изображения, при помощи передвижения ползунков. Данный метод состоит из трех модулей(этапов): Encoding module, Interactive Editing module, Decoding module. Все они используют в себе предобученные модели.

Encoding модуль разделяет face изображение на 4 части: 3 латентных представления и маску сегментации. Для получения латентных представлений используются 3 предобученных encoder'а - $E_C, E_T, E_S$, каждый из которых является предобученной нейронной сетью SEAN [17]. Полученные представления отвечают за цвет, текстуру и форму соответственно. Все encoder'ы были обучены таким образом, чтобы embedding'и $E_C, E_T, E_S$ имели нормальное $\mathcal{N}(0, 1)$ распределение. Маска сегментации нужна, чтобы восстанавливать лицо и задний план изображения, ее мы получаем при помощи нейронной сети BiSeNet.

На втором этапе происходит различное видоизменение латентных представлений. Это может происходить посредством передвижения ползунков или домешивания других латент-

ных представлений.

Последний, decoding модуль используется для получения итогового изображения из модифицированных латентных представлений и маски сегментации. Для этого используются предобученный shape adaptor - нейронная сеть для inpainting'а на основе латентных представлений исходного изображения и decoder нейросети SEAN.

Данный метод привнес 2 новые идеи в область переноса причесок. Во-первых, он продемонстрировал эффективность использования SEAN в качестве кодировщика и генератора признаков. Во-вторых, в методе впервые использовалась постобработка изображений, которая способствовала значительному улучшению метрик.

# 4 Описание базовых методов

## 4.1 HairFlash

В нашей работе базовым является метод HairFlash.

На основе разобранных выше статей нетрудно заметить, что для решения разных промежуточных задач при переносе прически есть 2 возможных подхода: с одной стороны можно оптимизировать представление в латентном пространстве, с другой можно использовать уже заранее обученные encoder'ы, чтобы получать нужные представления. У обоих подходов есть свои сильные и слабые стороны. Модели, основывающиеся на латентной оптимизации, показывают хорошее качество, но работают долго. Методы, построенные на предобученных энкодерах генерируют final изображение сильно быстрее, но не показывают столь хорошее качество.

Замысел авторов этого метода в том, чтобы заменить шаги, на которых использовалась оптимизация при помощи encoder'ов таким образом, чтобы качество не ухудшилось. За основу берется метод из статьи Barbershop, потому что в нем есть несколько понятных этапов, каждый из которых можно последовательно пробовать заменять энкодером и добиваться желаемого результата.
Данный метод состоит из 4 этапов:

1. Поиск представлений изображений в различных латентных пространствах и масок сегментаций.

2. Поиск F составляющей для final изображения.

3. Поиск S составляющей для final изображения.

4. Улучшение изображения посредством postprocessing'а.

Наглядное их преставление представлено на Рисунке 4.1.

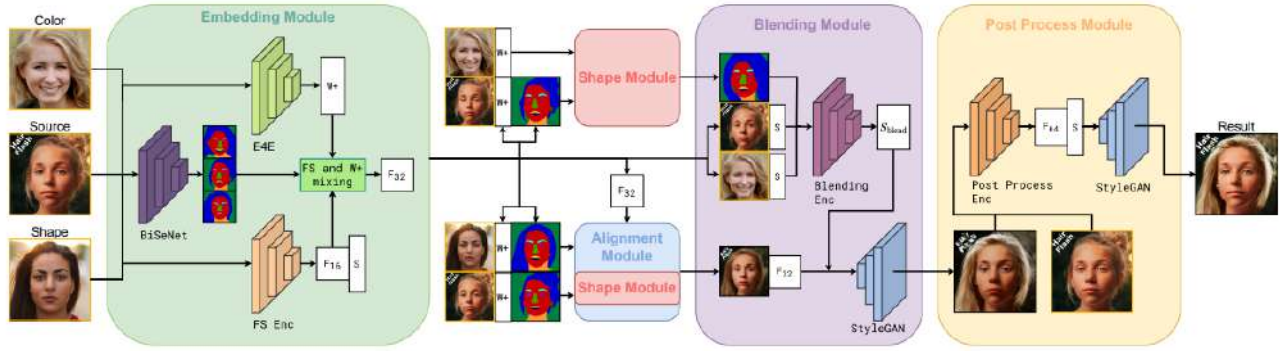



Рис. 4.1: Фреймворк FlashHair целиком.

### 4.1.1 Поиск представлений изображений в различных латентных пространствах и масок сегментаций.

На первом этапе мы хотим получить латентные представления для face, shape, color изображений. Для этого используются encoder в FS пространство[18] и, уже привычный нам, E4E encoder в W+ пространство. Проблема FS представления в том, что в нем сложно оптимизировать, так как объект очень большой. Поэтому поиск представления в W+ необходим. Итак, алгоритм следующий для каждого из изображений:

1. Применяем FS encoder и E4E:

$$F_{16}^{FSE},\ S = FS_{enc}(I)$$
$$W^{E4E} = E4E(I)$$

2. Так как хотим работать со структурой в $F_{32}$, переводим в это пространство полученные представления.

$$F_{32}^{FSE} = G_{4:6}(F_{16}^{FSE}, S)$$
$$F_{32}^{E4E} = G_{0:6}(W^{E4E}),$$

где $G_{x:y}$ - слои с x по y в генераторе StyleGan2.

3. Поиск масок сегментации:

$$M = BiSeNet(I)$$
$$H = DownSample_{32}(M = hair)$$

4. Смешиваем все, что получили на предыдущих шагах:

$$F_{32}^{mix} = \overline{H} \odot F_{32}^{FSE} + (1 - \alpha) * H \odot F_{32}^{FSE} + \alpha * H \odot F_{32}^{E4E}.$$

Авторы статьи брали $\alpha = 0.95$.

Наглядное представление алгоритма показано на Рисунке 4.2.

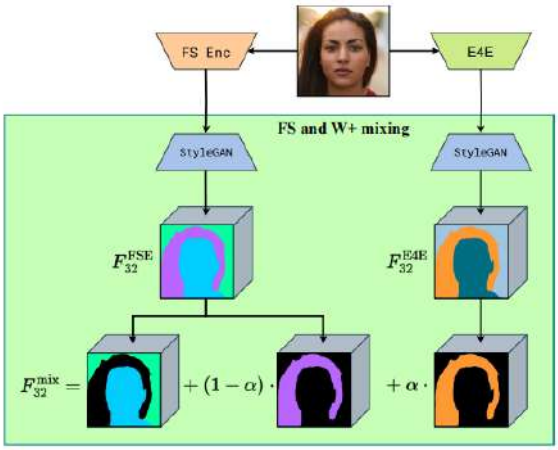



Рис. 4.2: Визуализация получения $F_{32}^{mix}$ из преставлений в $FS$ и $W+$.

Таким образом для каждого изображения мы получили:

- M - маску сегментации.
- H - маску сегментации для волос.
- Латентные представления $w^{E4E}, S, F_{32}^{mix}$

### 4.1.2 Поиск F составляющей для final изображения.

В этой части происходят действия только с face и shape изображениями, потому что color изображение не должно влиять на форму и структуру final изображения.
Данный модуль состоит из двух подчастей:

1. Сначала ищется маска сегментации с желаемой формой волос.

2. А затем по этой маске сегментации восстанавливается $F_{32}^{align}$ - структура итогового изображения.

**Поиск маски сегментации.**

Тут используется идея из метода CtrlHair с добавлением к ней RotateEncoder'a, который будет отвечать за поворот shape изображения.
В данной подчасти происходит следующее:

1. Поиск латентного представления, которое отвечает повернотому shape изображению.

$$w_{rotate} = Rotate_{Enc}(W_{face}^{E4E}, w_{shape}^{E4E}).$$

2. Поиск маски сегментации для полученного изображения.

$$M_{rotate} = BiSeNet(G(w_{rotate})),$$

где G - это генератор StyleGan2.

3. Поиск embedding'ов для частей масок сегментаций.

$$hair_{emb} = Shape_{Enc}^{hair}(M_{rotate}),$$
$$face_{emb} = Shape_{Enc}^{face}(M_{face}).$$

4. Получение маски сегментации с желаемой формой волос.

$$M_{align} = Shape_{Adaptor}(hair_{emb}, face_{emb}).$$

**Восстановление изображения по маске сегментации.**

SEAN - модель, которая может достать из изображения style векторы, а потом на их основе восстановить изображение по маске сегментации. В этой части авторы этим и пользуются, алгоритм следующий(проделываем его как для face, так и для shape изображения):

1. Получаем style векторы из изображения.

$$style\_codes = SEAN_{Enc}(I, M).$$

2. Генерируем выравненное изображение.

$$I^{inpaint} = SAEN_{Dec}(style\_codes, M_{align}).$$

3. Применяем E4E и первые 6 слоев генератора, чтобы получить представление выравненного изображения в F пространстве.

$$F_{32}^{inpaint} = G_{0:6}(E4E(I^{inpaint}))$$

F составляющую для final изображения получаем по следующей формуле:

$$\begin{aligned} F_{32}^{align} = {} & H_{align} \odot H_{shape} \odot F_{shape}^{mix} \quad + \\ & + H_{align} \odot \overline{H_{shape}} \odot F_{shape}^{inpaint} + \\ & + \overline{H_{align}} \odot \overline{H_{face}} \odot F_{face}^{mix} \quad + \\ & + \overline{H_{align}} \odot H_{face} \odot F_{face}^{inpaint}. \end{aligned}$$

Интуиция следующая: там где у final изображения волосы берем shape изображение, где нет - face. Где все плохо - inpaint'им. Иллюстрацию алгоритма выравнивания можно увидеть на Рисунке 4.3

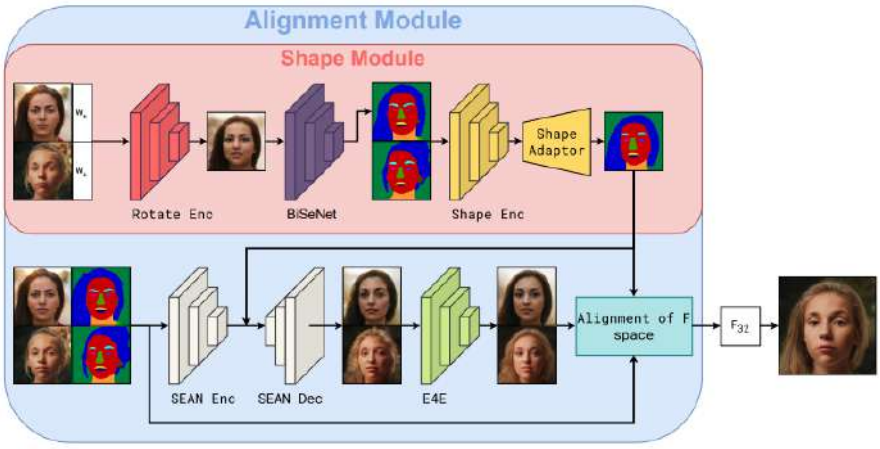



Рис. 4.3: Визуализация модуля выравнивания. Модуль принимает на вход, маски сегментаций изображений, $W+$ и $F$ представления с целью перенести желаему форму и структуру прически.

### 4.1.3 Поиск S составляющей для final изображения.

В силу того, что цвет и стиль мы хотим перенести с color изображения, shape изображение в этой части участвовать не будет. Авторы утверждают, что способ поиска распределения на S векторах, который использовался в методе Barbershop, устарел и предлагают способ на основе encoder'a.
Алгоритм следующий:

1. Получаем embedding'и нужных частей face и color изображений.

$$emb_{face} = CLIP_{Enc}(I_{face} \odot \overline{H_{align\_color}} \odot \overline{H_{face}} \odot \overline{H_{color}})$$
$$emb_{hair} = CLIP_{Enc}(I_{color} \odot H_{color})$$

2. Смешиваем полученные embedding'и предобученным encoder'ом.

$$S_{blend} = Blend_{Enc}(S_{face}, S_{color}, emb_{face}, emb_{hair})$$

Архитектура и способ обучения Blending encoder'a будут подробнее описаны в разделе 4.2.

### 4.1.4 Улучшение изображения посредством postprocessing'a.

Final изображение получаем путем применения генератора StyleGan2 к полученным представлениям:

$$I_{blend} = StyleGan2(F_{32}^{align}, S_{blend})$$

Далее на заключительном шаге происходит postprocessing. Как показал опыт метода CtlrHair, пост обработка может улучшить метрики, а значит сделать получающиеся изображения более реалистичными. Для улучшения $I_{blend}$ изображения обучается 3 новые модели: $FS_{Enc}$ - по картинке возвращает $F_{64}$ и S, $Fused_{FEnc}$ - модель для смешивания представлений в F пространстве, $Fused_{SEnc}$ - модель для смешивания представлений в S пространстве.
Используя их, получаем следующий алгоритм postprocessing'a:

1. Находим представления $I_{blend}$ и $I_{face}$.

$$F_{64}^{blend}, S^{blend} = FS_{Enc}(I_{blend})$$
$$F_{64}^{blend}, S^{blend} = FS_{Enc}(I_{blend})$$

2. Смешиваем полученные представления.

$$F_{final} = Fused_{FEnc}(F_{64}^{blend}, F_{64}^{face})$$
$$S_{final} = Fused_{SEnc}(S^{blend}, S^{face})$$

3. И наконец получаем улучшенное final изображение.

$$I_{final} = StyleGan2(F_{64}^{final}, S_{final})$$

Наглядное представление алгоритма показано на Рисунке 4.4.

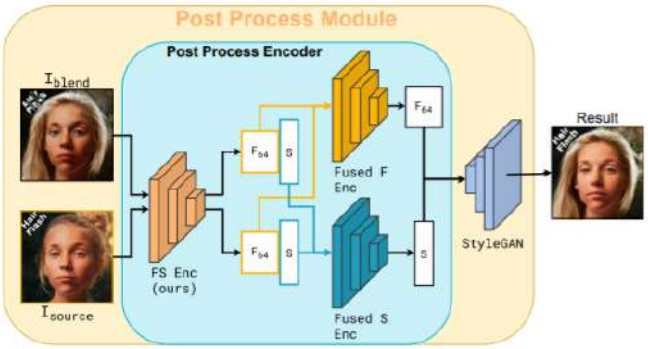



Рис. 4.4: Визуализация модуля постобработки. Модуль принимает на вход получившееся на предыдущем шаге и face изображения, целью модуля является улучшить качество final изображения.

## 4.2 Blending модуль.

Рассмотрим подробнее как устроена и как обучалась модель, использующаяся в модуле blending'a, потому что далее мы будем заниматься именно ее улучшением.

Для обучения был собран датасет из 15000 троек изображений из датасета FFHQ. Изображения выбирались таким образом, чтобы у color изображения были волосы, чтобы было откуда переностить цвет и стиль. Для каждой тройки были проделаны все шаги метода до модуля blending'a. Таким образом каждая единици в тренировочной выборке состаяла из $I_{face}$, $S_{face}$, $F_{face}^{align}$ color изображения, выравненного под face изображение, $I_{color}$, $S_{color}$. Наша цель - обучить модель модифицировать $S_{face}$ таким образом, чтобы при генерации получалось изображение с цветом волос как у $I_{color}$ с данным $F_{face}^{align}$ тензором.

В качестве архитектуры авторами была выбрана нейронная сеть, показанная на рисунке 4.5. На вход в encoder подаются S векторы и embedding'и модели CLIP для частей

face и shape изображений. Сама сеть состоит из полносвязных слоев, модулей модуляции и активаций. Каждый модуль модуляции также содержит в себе полносвязные слои.

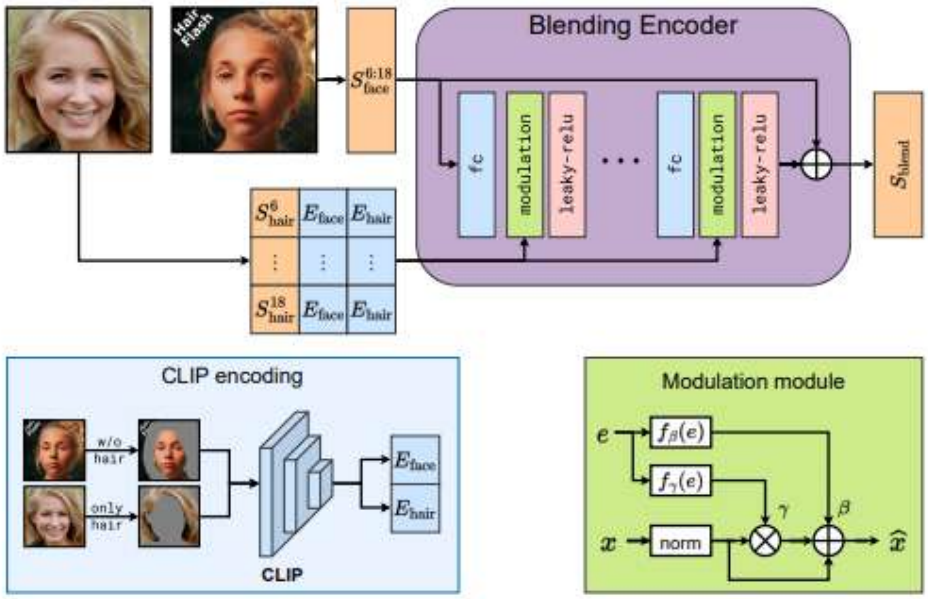



Рис. 4.5: Диаграмма модуля blending'a. $f_\beta(e)$ и $f_\gamma(e)$ - это нейронные сети со следующей архитектурой: Linear(1536, 1024) -> LayerNorm(1024) -> LeakyReLU(0.01) -> Linear(1024, 512).

Для обучения сети отобранный датасет пропускался через нейронную сеть в течении 1000 эпох следующим образом:

1. Сначала подготавливались необходимые маски сегментаций.

$$H_{color} = (BiSeNet(I_{color}) = hair),$$
$$H_{face} = (BiSeNet(I_{face}) = hair),$$
$$H_{align\_color} = (BiSeNet(G(F_{face}^{align})) = hair),$$
$$M_{target} = \overline{H_{face}} \odot \overline{H_{align\_color}} \odot \overline{H_{color}}.$$

2. Далее использовался сам blending encoder.

$$emb_{face} = CLIP_{Enc}(I_{face} \odot M_{target}),$$
$$emb_{hair} = CLIP_{Enc}(I_{color} \odot H_{color}),$$
$$S_{blend} = Blend_{Enc}(S_{face}, S_{color}, emb_{face}, emb_{hair}),$$
$$I_{blend} = StyleGan2(F_{face}^{align}, S_{blend}).$$

3. Затем, для того, чтобы модель обучалась использовались функция потерь следующего вида:

$$\mathcal{L}_{clip}(I_1, I_2, M_1, M_2) = 1 - CosSim_{clip}(I_1 \odot M_1, I_2 \odot M_2),$$

где $CosSim_{clip}$ - это косинусное расстояние между embedding'ами clip'а для частей полученных изображений. Итоговый лосс для обучения получался следующим образом:

$$\begin{aligned}
\mathcal{L} &= \lambda_{color}\mathcal{L}_{color} + \lambda_{face}\mathcal{L}_{face}, \\
\mathcal{L}_{color} &= \mathcal{L}_{clip}(I_{blend}, I_{color}, H_{align}, H_{color}), \\
\mathcal{L}_{face} &= \mathcal{L}_{clip}(I_{blend}, I_{face}, M_{target}, M_{target}).
\end{aligned}$$

Авторы берут $\lambda_{color} = \lambda_{face} = 1$ и используею оптимайзер Adam с learning rate $= 1 \times 10^{-4}$ и weight decay $1 \times 10^{-6}$

# 5 Протокол тестирования и визуализации.

Зафиксируем протокол тестирования и визуализации, которого мы будем придерживаться при тестировании методов для переноса прически. Используется два подхода - качественный и количественный. В обоих подходах используются изображения из датасета Celeba-HQ.

В количественном подходе подсчитываются метрики $FID$ и $FID_{clip}$. Они измеряют расстояние между распределениями реальных изображений и сгенерированных моделью изображений на основе внутренних представлений глубокой сверточной сети, обученной на классификацию изображений. Далее я буду рассматривать не только задачу, в которой face, shape и color изображения разные, но и другие комбинации. Все варианты задач представлены в таблице 5.1

В качественном подходе мы будем глазами отсматривать полученные изображения и оценивать работу моделей. Изображения всегда будут конкатенироваться рядом друг с другом, тем самым образуя столбцы. В первых трех столбцах всегда будут места для Face, Shape и Color изображений.

| Название | Face | Shape | Color |
|---|---|---|---|
| full | face | shape | color |
| color | face | face | color |
| shape | face | shape | face |
| both | face | shape | shape |
| rec | face | face | face |

Таблица 5.1: Варианты задач для подсчета метрик. face, shape, color - это 3 различных изображения. Face, Shape, Color указывают на место, на которое мы берем face, shape или color изображение.

.

# 6 Сравнение базового метода с другими.

Чтобы убедиться, что HairFlash и правда наилучший метод, мы сравнили его с предшествующими методами, запустив каждый из них собственноручно.

## 6.1 Выбор методов для сравнения.

Для сравнения мы выбрали методы Barbershop и HairClip.

Barbershop попал в этот список, потому что:

1. Он является первым методом, на котором удалось получить хорошие результаты.
2. Он показывает наилучшие результаты среди методов по метрике FID.
3. Последующие статьи сравнивают свои методы с ним.

Hairclip в свою очередь я решил взять потому, что

1. Он способен принимать на вход в качестве целевой информации не только изображения, но и текст, а это может пригодиться в будущем в качестве бейзлайна, если моя модель будет уметь работать с аналогичной входной информацией.
2. Этот метод работает гораздо быстрее, чем Barbershop, а значит с ним можно будет сравнивать скорость работы моего алгоритма.

## 6.2 Фиксация параметров моделей.

Для воспроизведения получившихся результатов зафиксируем параметры, с которыми были запущены модели при получении результатов.

Конфигурационные параметры метода Barbershop:

| Parameter | Value |
|---|---|
| sign | 'realistic' |
| smooth | 5 |
| size | 1024 |
| channel_multiplier | 2 |
| latent | 512 |
| n_mlp | 8 |
| opt_name | 'adam' |
| learning_rate | 0.01 |
| lr_schedule | 'fixed' |
| percept_lambda | 1.0 |
| l2_lambda | 1.0 |
| p_norm_lambda | 0.001 |
| l_F_lambda | 0.1 |
| W_steps | 1100 |
| FS_steps | 250 |
| ce_lambda | 1.0 |
| style_lambda | 4e4 |
| align_steps1 | 140 |
| align_steps2 | 100 |
| face_lambda | 1.0 |
| hair_lambda | 1.0 |
| blend_steps | 400 |

Конфигурационные параметры метода HairClip:

| Parameter | Value |
|---|---|
| stylegan_size | 1024 |
| test_batch_size | 1 |

Конфигурационные параметры метода HairFlash:

| Parameter | Value |
|---|---|
| mixing | 0.95 |
| smooth | 5 |
| size | 1024 |
| channel_multiplier | 2 |
| latent | 512 |
| n_mlp | 8 |
| batch_size | 3 |

## 6.3 Результаты запуска методов.

Ниже приведены результаты качественного и количественного сравнения методов. Для начала сравним получающиеся изображения. В четвертом столбце - результат метода HairClip, в пятом и шестом - Barbershop и HairFlash соответственно.


Анализ:

1. В большинстве случаев все 3 метода с поставленной задачей справляются.

2. HairFlash справился лучше во всех случаях, кроме 3-го. В третьем случае у Barbershop'а лучше получилось перенести как кудри с shape изображения, так и цвет с color изображения.

3. Есть случаи, где видны недочеты в работе предшествующих методов. Например HairClip не смог переменсти форму в 3 и 5 случаях, а у Barbershop'а появился непонятный артефакт на волосах в 4 случае.

Количественное сравнение работы методов представлено в таблице 6.1. Релаизация и веса для методов Barbershop и HairClip была взята из их официальных репозиториев. Изначальный репозиторий с кодом метода HairFlash я получил от научного руководителя, так как метод еще находился на стадии разработки и не был в открытом доступе. В этом репозитории уже были предобученные чекпоинты для всех модулей, которыми я и воспользовался для получения метрик.

| | FID | | | | $FID_{CLIP}$ | | | | Time(s) |
|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | full | color | shape | both | V100 |
| Barbershop | <u>15.94</u> | <u>24.52</u> | **20.54** | <u>24.08</u> | <u>7.07</u> | <u>8.12</u> | **3.89** | <u>6.76</u> | 645 |
| HairClip | 34.95 | 40.68 | 40.08 | 42.92 | 12.20 | 13.32 | 10.94 | 13.44 | **0.36** |
| HairFlash | **13.84** | **21.28** | <u>24.04</u> | **23.31** | **5.46** | **3.53** | <u>5.40</u> | **6.46** | <u>1.97</u> |

Таблица 6.1: Сравнение метрик для Barbershop, HairClip и HairFlash. Жирным выделен лучший результат, подчеркнутый - второй после лучшего.

Видно, что HairFlash является лучшим методом по большинству метрик, лишь иногда Barbershop обходит его. По времени работы видно насколько Barbershop медленнее, чем остальные 2 метода. Таким образом HairFlash действительно находит золотую середину между качеством и скоростью работы.

# 7 Исследование базового метода

## 7.1 Обучение модуля блендинга

Первым шагом я обучил модуль блендинга самостоятельно. В репозитории уже был написан скрипт для обучения, но он был написан в однопоточной манере, то есть его можно было выполнять максимум на 1 GPU, что делало обучение чрезвычайно долгим. Я дополнил имеющийся скрипт кодом, позволяющим запускать обучение сразу на нескольких GPU, чтобы ускорить обучение модуля и проведение экспериментов. В силу того, что датасет, на котором учили модель изначально довольно большой (15000 троек картинок) и проводить эксперименты на нем долго, я обучил бейзлайн на меньшем наборе данных (3000 троек картинок). Полученные результаты для baseline модели представлены в таблице 7.1

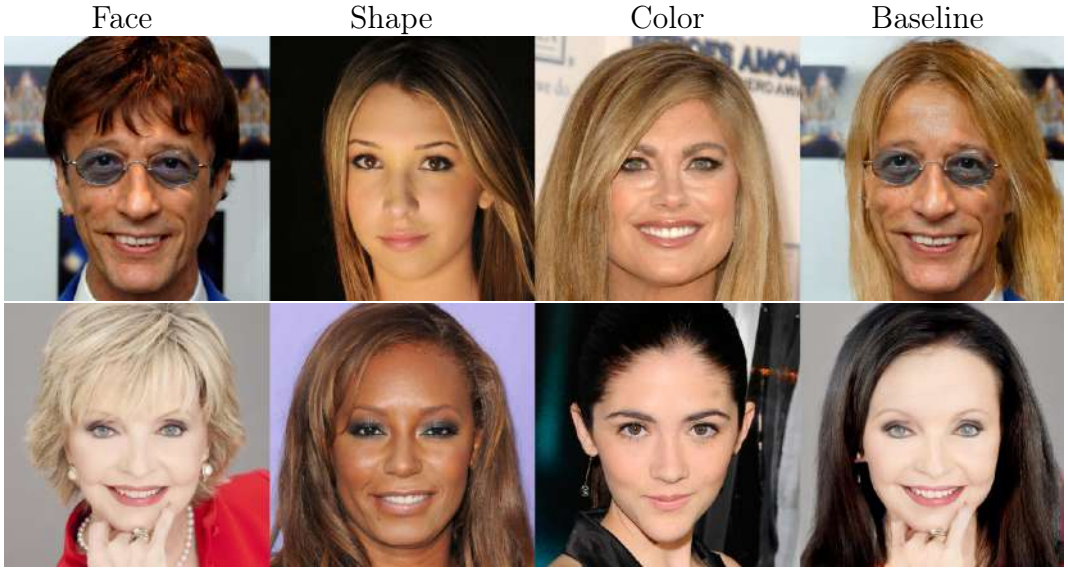
| | $FID$ | $FID_{CLIP}$ |
|---|---|---|
| full | 15.08 | 5.80 |
| color | 21.77 | 3.56 |
| shape | 25.33 | 5.78 |
| both | 24.44 | 6.66 |
| reconstruct | 9.97 | 1.63 |

Таблица 7.1: Результаты полученные, при помощи моего чекпоинта. Можно заметить, что, используя мой чекпоинт, качество получается хуже, что оправдано тем, что я обучал модуль на меньшем наборе данных и без использования двух этапного обучения с lpips.

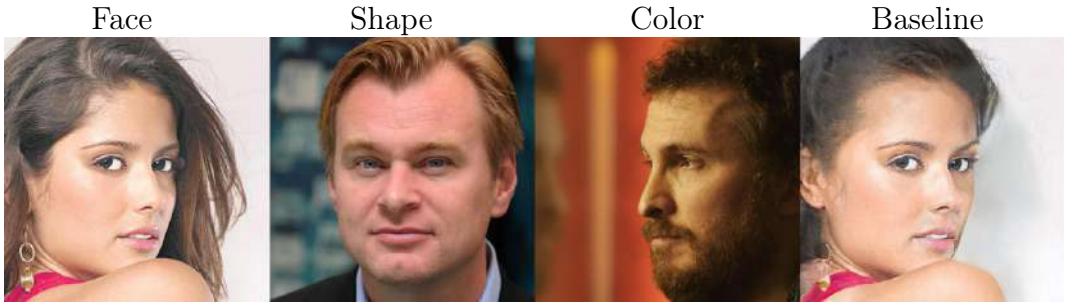
## 7.2 Качественный анализ метода.

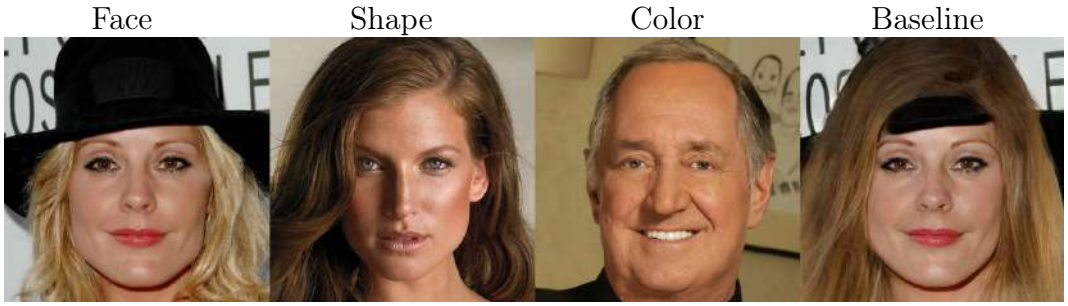
Посмотрим на случаи, когда метод справляется хорошо и плохо. Нашей дальнейшей целью будет улучшить работу метода в тех случаях, где он справляется плохо, при этом не испортив его качество на уже хорошо получающихся примерах.

Примеры, на которых метод отрабатывает идеально.




Примеры, с которыми метод справляется плохо.






Минусы:

- В первом случае модель совершенно не смогла передать текстуру волос.
- Во втором алгоритм не справился с тем, чтобы убрать головной убор с final изображения.

# 8 Эксперименты

## 8.1 Уменьшение времени работы метода

Сейчас в модуль блендинга подается 4 объекта - S векторы исходного изображения, S векторы color изображения, embedding-и не волос исходного изображения и embedding-и волос color изображения. Для того, чтобы получить embedding-и не волос исходного изображения используется бинарная маска сегментации, в которой единицами отмечена область, в которой нет волос у исходного изображения, нет волос у изображения, получающегося после выравнивания color изображения, и нет волос у color изображения 4.2.

Гипотеза заключается в том, что смотреть на выравненное color изображения может быть лишним. То есть если мы не будем использовать выравненное color изображение, то качество не ухудшится, а метод начнет работать быстрее, так как нам не нужно будет выравнивать color изображение. Формально теперь будем искать маску не волос face изображения следующим образом:

$$H_{color} = (BiSeNet(I_{color}) = hair),$$
$$H_{face} = (BiSeNet(I_{face}) = hair),$$
$$M_{target} = \overline{H_{face}} \odot \overline{H_{color}}.$$

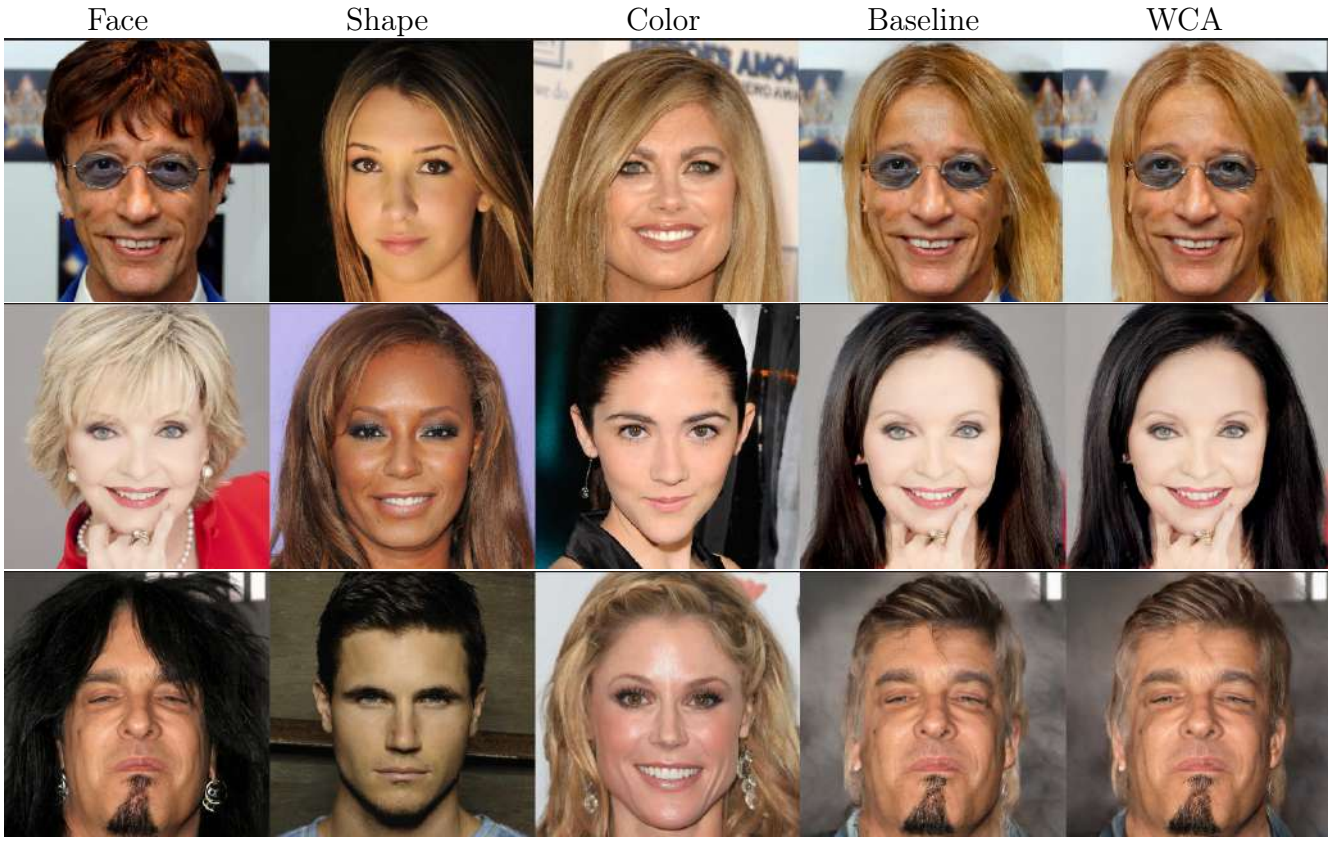
В ходе проверки первой гипотезы мы обучили модуль блендинга без использования маски сегментации, покрывающей волосы выравненного color изображения. Это ускорило

метод на 20 процентов, метрики при этом сильно не поменялись. Подробные количественные результаты эксперимента представлены в таблице 8.1

| | FID | | | | | $\text{FID}_{\text{CLIP}}$ | | | | | Time(s) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | rec | full | color | shape | both | rec | V100 |
| Baseline | 15.08 | 21.77 | 25.33 | 24.44 | **9.79** | **5.80** | 3.56 | **5.78** | **6.66** | **1.63** | 1.97 |
| WithoutColorAlignment | **15.03** | **21.78** | **25.28** | **24.03** | 9.90 | 5.84 | **3.50** | 6.20 | 6.98 | 1.83 | **1.63** |

Таблица 8.1: Сравнение метрик бейзлайна и экспериментальной модели. Жирным выделен лучший результат.

Сравним на нескольких примерах baseline с моделью, использовавшейся в эксперименте, чтобы удостовериться, что метод по-прежнему хорошо справляется:

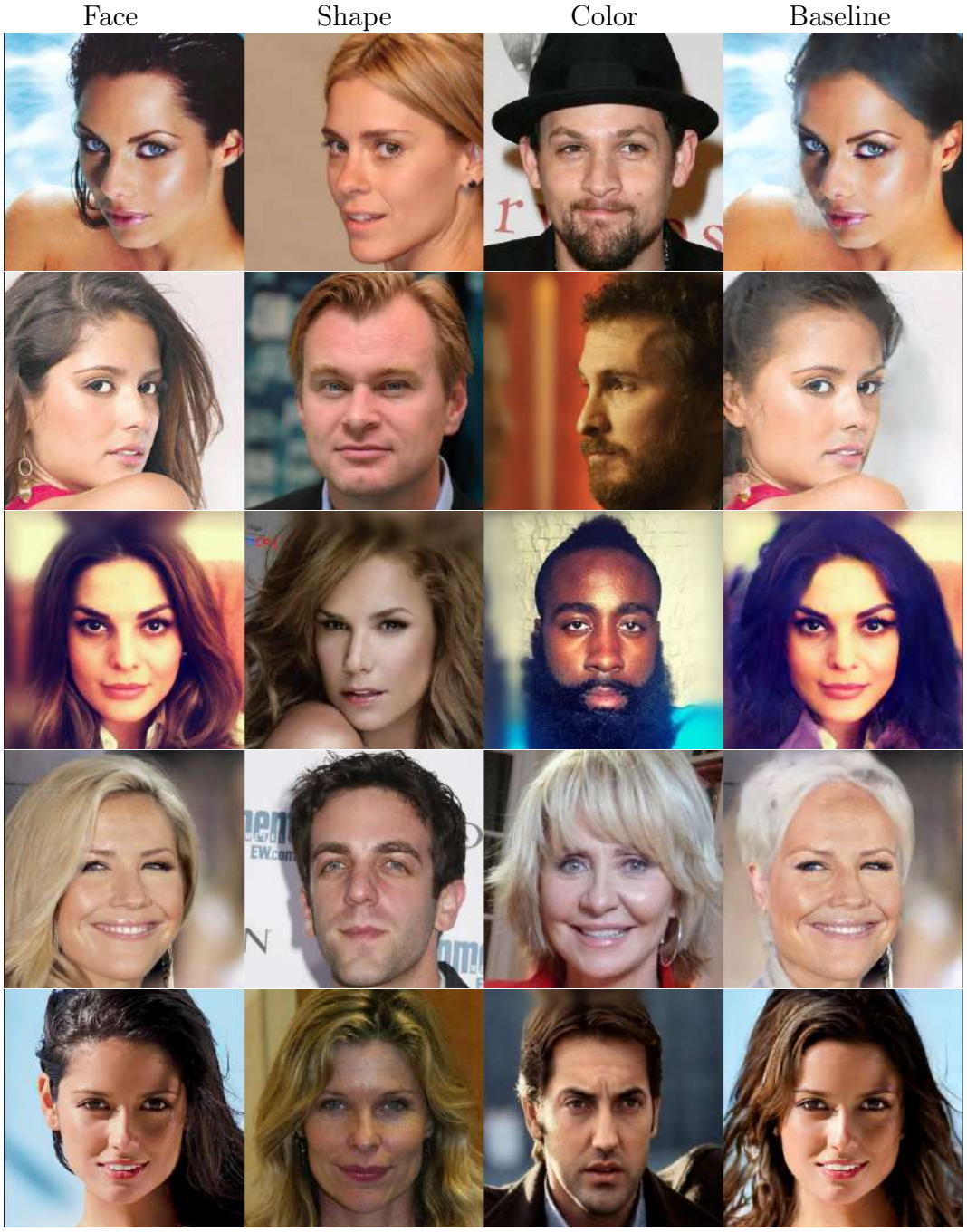



Анализ:

- В первом примере видим, что прическа перенеслась практически идентично baseline'у, но пробор стал более четким, что делает final изображение более реалистичным.
- Во втором случае опять же получился практически такой, же результат как и у baseline'а. Единственное отиличе в том, что волосы стали немного темнее.
- В третьем примере волосы получились немного светлее, что ближе к цвету color изображения. Но глобально все так же, как и в baseline'е.

Итог: данный эксперимент подтвердил гипотезу, что использование маски сегментации выравненного color изображения излишне. А значит можно ускорить метод убрав этот шаг.

## 8.2 Улучшение текстуры волос




На примерах наглядно видно, что прическа у получившихся изображений похожа по форме на shape изображение, по цвету на color изображение, но вот выглядят волосы по другому. Они размыты, не видно четких локонов. Складывается ощущение, как будто взяли нужную форму и закрасили краской. Как будто если бы модель знала при раскрашивании о том, как идут локоны у shape изображения, то она бы раскрасила их лучше. Далее рас-

смотрим различные архитектурные изменения, при помощи которых мы пытались научить модель реалистичнее раскрашивать получающуюся прическу. На пятом изображении, кроме непрорисовки локонов в правом верхнем углу, получились довольно неестественные волосы в левом нижнем углу, текстуру этих волос тоже хотелось бы улучшить.

Поскольку информация о локанах содержится в shape изображении, нужно было понять как мы хотим добавить ифомацию о shape изображении в blending модуль. Если посмотреть на вход encoder'а на рисунке 8.1, то можно заметить, что есть два места, в которые можно добавить информацию о shape изображении.

1. С одной стороны можно добавить эту информацию к S векторам source изображения. И уже к полученному представлению примешивать информацию о цвете волос.

2. С другой стороны можно примешивать эту информацию к S векторам source изображения вместе с информацией о цвете волос.

Кроме того, мы решили попробовать 2 варианта представления информации о shape изображении: в виде S векторов shape изображения и в виде embedding'ов волос shape изображения. Количественные результаты экспериментов представленны в таблице 8.2

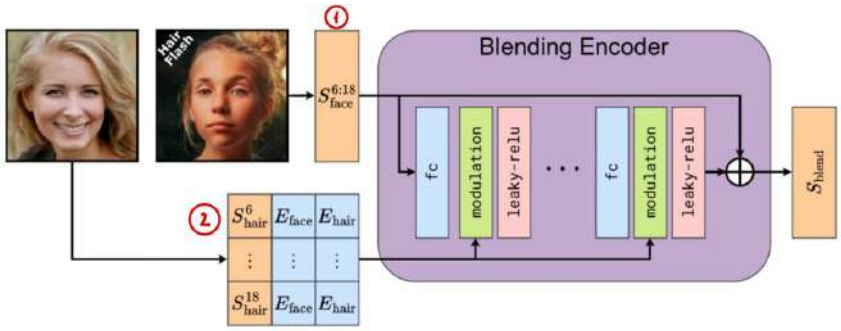


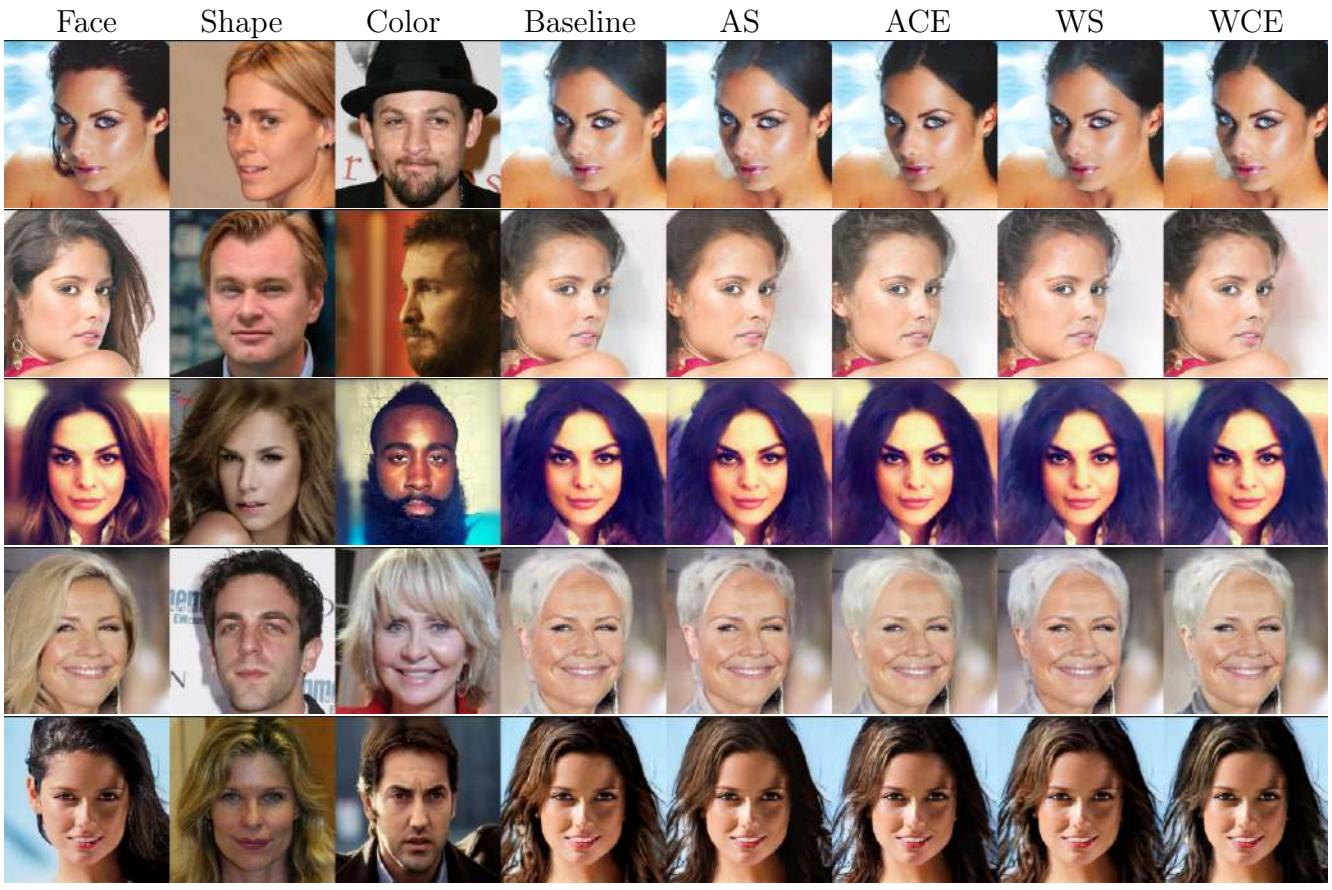

Рис. 8.1: Архитектура модели, использующейся для блендинга. Цифрами обозначены места, в которые мы пробовали добавлять информацию о shape изображении.

Нетрудно заметить, что лучше остальных показывает себя модель WithClipEmbeds - та, в которой мы присоединяли embedding'и волос, полученные из shape изображения, к S векторам face изображения, и к полученному представлению примешивали информацию о цвете волос.

| | FID | | | | | $\text{FID}_{\text{CLIP}}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | rec | full | color | shape | both | rec |
| Baseline | 15.08 | 21.77 | 25.33 | 24.44 | **9.79** | **5.80** | 3.56 | **5.78** | 6.66 | **1.63** |
| AdmixS(AS) | 15.19 | 22.05 | 25.35 | **24.02** | 10.18 | 6.28 | 3.85 | 6.17 | 7.03 | 1.81 |
| AdmixClipEmbeds(ACE) | 15.21 | 22.10 | 25.65 | 24.11 | 10.07 | 6.27 | 3.94 | 6.32 | 6.96 | 1.86 |
| WithS(WS) | 15.50 | 22.09 | 26.04 | 24.58 | 10.13 | 6.14 | 3.88 | 6.25 | 7.04 | 1.82 |
| WithClipEmbeds(WCE) | **14.82** | **21.66** | **25.26** | 24.22 | 10.08 | **5.80** | **3.51** | 6.19 | **6.55** | 1.72 |

Таблица 8.2: Сравнение метрик бейзлайна и моделей построенных для проведения экспериментов. AdmixS - модель, в которой мы примешиваем S векторы shape изображения, AdmixClipEmbeds - модель, в которой мы примешиваем embedding'и волос shape изображения, WithS - модель, в которой мы присоединили S векторы shape изображения к S векторам face изображения и уже к полученному представлению примешиваем информацию о цвете волос, WithClipEmbeds - модель, в которой мы присоединили embedding'и волос полученные из shape изображения к S векторам face изображения и опять же к полученному представлению примешиваем информацию о цвете волос. Жирным выделен лучший результат, подчеркнутый - второй после лучшего.

Проанализируем примеры работы этих моделей и проанализируем изменения. Названия сокращены до первых букв:




Анализ:

- Получившиеся изображения подтверждают получившиеся метрики - видно, что изображения полученные при помощи модели WithClipEmbeds стабильно получаются лучше, поэтому далее проанализируем, что стало лучше в сравнении с baseline'ом именно в

результатах этой модели.

- Видны улучшения в первом, втором и пятом случаях. В первом начал виднеться пробор, но все еще плохо раскрашиваются локоны. Во втором волосы стали гораздо менее размытыми, но текстура все еще не идеальна. В пятом случае также, как и в первом начал виднеться пробор, но текстура, как в левом нижнем углу, так и в правом верхнем еще далека от идеала.

- Во втором и третьем случаях улучшений не наблюдается.

Вывод: добавление информации о shape изображении может улучшить способность модели раскрашивать локоны. Модель WithClipEmbeds приблизила нас к тому, чтобы поправить часть случаев и улучшила метрики, поэтому теперь новые улучшения будем применять к ней.

Далее для того, чтобы еще больше помочь модели улавливать текстуру волос и реалистичнее раскрашивать локоны, мы решили попробовать добавить лосс на shape изображение. В экспериментах далее мы рассмотрели 2 различных лосса - MSE и CosSim.
Для MSE подсчитывался следующий лосс:

$$
\begin{aligned}
\mathcal{L} &= \lambda_{color}\mathcal{L}_{color} + \lambda_{face}\mathcal{L}_{face} + \lambda_{shape}\mathcal{L}_{shape}, \\
\mathcal{L}_{color} &= \mathcal{L}_{clip}(I_{blend}, I_{color}, H_{align}, H_{color}), \\
\mathcal{L}_{face} &= \mathcal{L}_{clip}(I_{blend}, I_{face}, M_{target}, M_{target}), \\
I^{gray}_{blend} &= Grayscale(I_{bland}) \\
I^{gray}_{face} &= Grayscale(I_{face}) \\
\mathcal{L}_{shape} &= MSE_{clip}(I^{gray}_{blend}, I^{gray}_{face}, M_{target}, M_{target})
\end{aligned}
$$

А для CosSim shape лосс считался аналогично остальным - при помощи косинусного расстояния:

$$\mathcal{L} = \lambda_{color}\mathcal{L}_{color} + \lambda_{face}\mathcal{L}_{face} + \lambda_{shape}\mathcal{L}_{shape},$$
$$\mathcal{L}_{color} = \mathcal{L}_{clip}(I_{blend}, I_{color}, H_{align}, H_{color}),$$
$$\mathcal{L}_{face} = \mathcal{L}_{clip}(I_{blend}, I_{face}, M_{target}, M_{target}),$$
$$I^{gray}_{blend} = Grayscale(I_{bland})$$
$$I^{gray}_{face} = Grayscale(I_{face})$$
$$\mathcal{L}_{shape} = \mathcal{L}_{clip}(I^{gray}_{blend}, I^{gray}_{face}, M_{target}, M_{target}).$$

В силу того, что мы не хотим, чтобы цвет волос shape изображения протек в исходное изображение, лоссы мы считали между изображениями преобразованными в grayscale.

Кроме того, появилась гипотеза, что когда мы ищем embedding'и shape изображения, стоит искать embedding'и не напрямую изображения, а его grayscale версии. Обосновывается эта гипотеза тем, что для улавливания текстуры волос, например локонов, не нужен цвет волос. Соответственно из черно-белой картинки мы сможем доставать ровно ту информацию, которую нам нужно.

Посмотрим на количественные результаты экспериментов:

| | FID | | | | | $\text{FID}_{\text{CLIP}}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | rec | full | color | shape | both | rec |
| Baseline | 15.08 | 21.77 | 25.33 | 24.44 | **9.79** | **5.80** | 3.56 | **5.78** | 6.66 | **1.63** |
| WithClipEmbeds | **14.82** | **21.66** | **25.26** | **24.22** | 10.08 | **5.80** | **3.51** | 6.19 | **6.55** | 1.72 |
| CosSimColor(CSC) | 16.23 | 23.19 | 26.88 | 25.93 | 10.40 | 7.22 | 4.21 | 7.64 | 8.04 | 1.93 |
| CosSimGray(CSG) | 15.63 | 22.40 | 25.90 | 24.90 | 10.30 | 7.68 | 4.26 | 8.15 | 8.18 | 1.96 |
| MseColor(MC) | 15.00 | 21.89 | 25.58 | 24.32 | 10.12 | 6.55 | 3.88 | 6.90 | 7.29 | 1.91 |
| MseGray(MG) | 15.39 | 22.15 | 25.74 | 25.09 | 10.08 | 6.40 | 3.83 | 6.72 | 7.33 | 1.80 |

Таблица 8.3: Сравнение метрик бейзлайна и экспериментальных моделей. CosSimColor - модель, в которой использовался лосс CosSim и на вход подавались embedding'и волос цветного изображения, CosSimGray - лосс CosSim и на вход подавались embedding'и волос изображения преобразованного в grayscale, MseColor - MSE лосс и на вход подавались embedding'и волос цветного изображения, MseGray - MSE лосс и на вход подавались embedding'и волос изображения преобразованного в grayscale. Жирным выделен лучший результат, подчеркнутый - второй после лучшего.

По метрикам явно видно, что добавление лоссов только ухудшает качество модели. Сравним на наших примерах baseline, лучшую модель без применения лоссов - WithClipEmbeds и все модели с лоссами. Некоторые названия сокращены до первых букв, порядок совпадает с порядком моделей в таблице 8.3


Анализ:

- Видно, что лучше прическа и, в частности, текстура не начала переноситься ни в одном случае.
- В четвертом случае даже видны странные спецэффекты.

Вывод: добавление вышеописанных лоссов не помогает улучшить текстуру final изображения при переносе прически.

После неудачи с проверенными выше лоссами мы решили, что проблема в том, что мы подаем в лосс grayscale изображения и это может недостаточно точно указывать модели на текстуру локонов. Поэтому далее мы решили попробовать подавать в лосс не grayscale изображения, а дополнительно к этому применять к ним фильтр Собеля. Данный фильтр предназначен для выделения границ, а значит он сможет помочь модели понимать форму локонов на shape изображении. Пример работы фильтра Собеля изображен на рисунке 8.3

Далее было проведено 2 эксперимента, в которых все лоссы остались прежними, кроме $\mathcal{L}_{shape}$. В первом эксперименте для его вычисления использовалась, уже привычная, косинусная похожесть: $\mathcal{L}_{shape} = \mathcal{L}_{clip}(I^{sobel}_{blend}, I^{sobel}_{face}, M_{target}, M_{target})$. Тем не менее по результатам первого эксперимента стало понятно, что модель плохо справляемся с переносом цвета, несмотря на значительные улучшения в прорисовке локонов. Мы предположили, что это может происходить из-за того, что при нахождении embedding'ов clip'ом используются grayscale картинки,

Рис. 8.2: Исходное изображение.

Рис. 8.3: После применения фильтра Собеля.

однако clip под это не заточен. Поэтому во втором случае для shape лосса мы решили отойти от идеи использования embedding'ов clip'а и попробовали напрямую считать MSE между получающимися после применения фильтра собеля изображениями: $\mathcal{L}_{shape} = MSE(I^{sobel}_{blend}, I^{sobel}_{face})$. В таблице 8.4 представлены количественные результаты проведения экспериментов. По ним видно, что у WithClipEmbeds и SobelCosSim получаются примерно одинаковые метрики.

| | FID | | | | | $\text{FID}_{\text{CLIP}}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | rec | full | color | shape | both | rec |
| Baseline | 15.08 | 21.77 | 25.33 | 24.44 | **9.79** | **5.80** | 3.56 | 5.78 | 6.66 | 1.63 |
| WithClipEmbeds | **14.82** | **21.66** | 25.26 | **24.22** | 10.08 | **5.80** | **3.51** | 6.19 | **6.55** | 1.72 |
| SobelCosSim | 14.95 | 21.73 | **25.25** | 24.35 | 10.40 | 5.92 | 3.55 | **5.70** | 6.59 | 1.93 |
| SobelMSE | 16.43 | 24.10 | 26.42 | 25.68 | 11.75 | 6.05 | 3.93 | 7.19 | 7.01 | 2.12 |

Таблица 8.4: Сравнение метрик бейзлайна и экспериментальных моделей. SobelCosSim - модель из первого эксперимента, в котором использовался фильтр Собеля и косинусное расстояние, SobelMSE - модель из второго эксперимена, в котором использовался фильтр Собеля и MSE. Жирным выделен лучший результат, подчеркнутый - второй после лучшего.

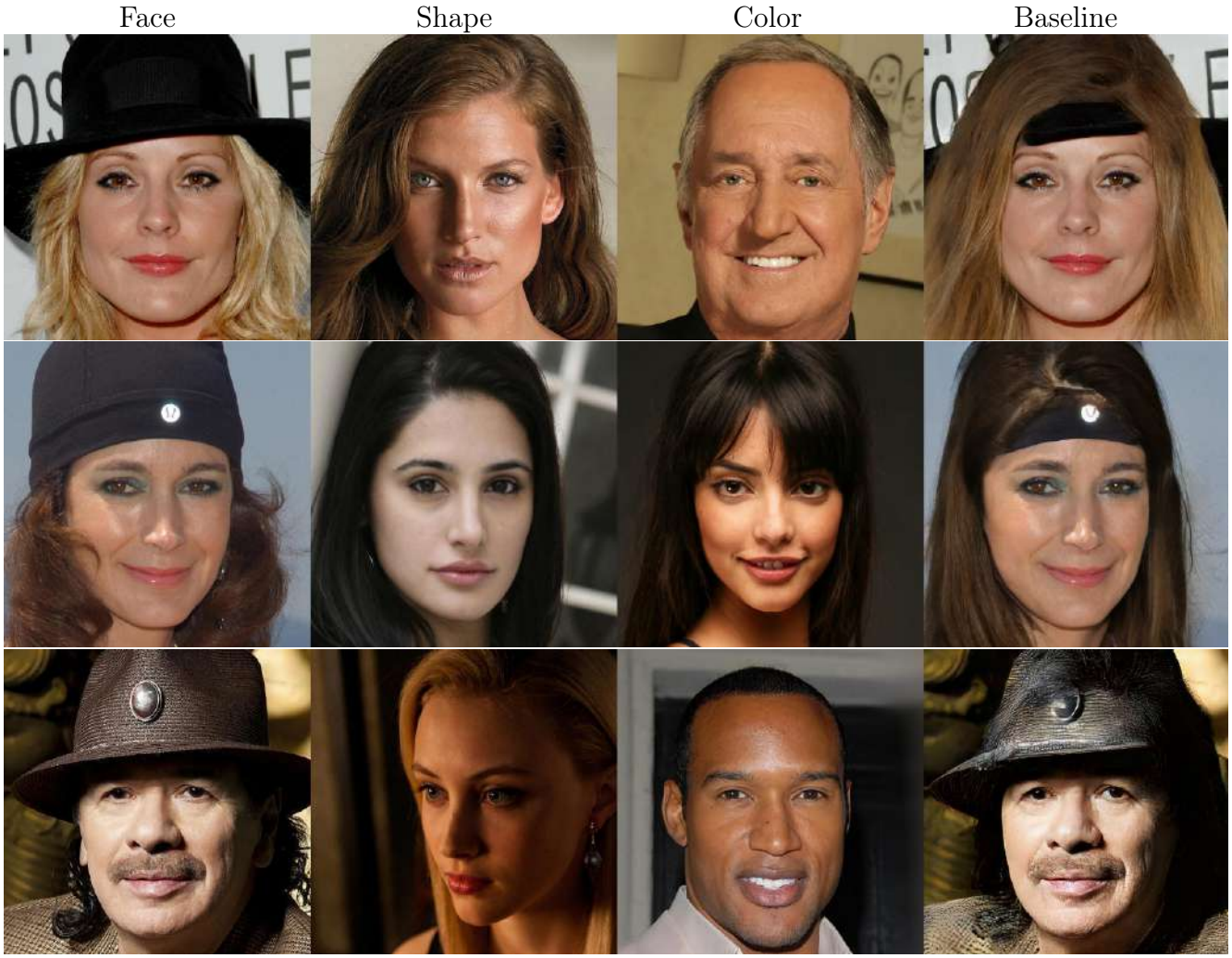
Проведем качестсвенный анализ экспериментов.

Анализ:

- Заметим, что обе новые модели гораздо лучше справляются с раскрашиванием локонов. Во всех пяти случаях видны заметные улучшения в текстуре волос.
- У всех моделей, кроме SobelCS есть большой недостаток, он хорошо виден на втором примере. У всех моделей, кроме SobelCS остался след в том месте, где были волосы у face изображения, а вот SobelCS справилась с тем, чтобы идеально обработать это место.
- Несмотря на значительные улучшения в прорисовке локонов у моделей SobelCS и SobelMSE, видно, что в некторых случаях, например в 3 и 4, они плохо справляются с переносом цвета color изображения.

Вывод: использование подходящих лоссов действительно приводит к лучшему переносу текстуры с shape изображения. SobelCS - лучшая из описанных на данный момент моделей.
**Итог:** в ходе экспериментов мы поняли, что модель SobelCS дает наилучший результат, поэтому в окончательном варианте будем использовать ее.

## 8.3 Обработка головных уборов




На фотографиях у людей зачастую есть головные уборы, либо очки на волосах. В большинстве случаев это мешает переносу прически, как мы можем наблюдать на изображениях выше. В связи с этим следующая гипотеза заключается в том, чтобы попробовать предобрабатывать изображения, снимая с них головные уборы. Если более формально, то я хочу попробовать реализовать это следующим образом. Во-первых для каждой картинки сделать предобработку еще до этапа получения эмбеддингов. Предобработка будет выполняться в 2 шага. На первом мы сегментируем изображение и удаляем участок, отвечающий головному убору, если таковой есть. На втором, если нам удалось убрать головной убор на первом шаге - дорисовываем изображение, то есть решаем задачу inpainting'а. Далее делаем все шаги так же как и ранее.

Для решения проблемы с головными уборами я воспользовался нейронной сетью HairMapper [19], которая способна убрать волосы с изображения, то есть сделать человека лысым. Наглядный пример работы данной модели изображен на рисунке 8.5:

Рис. 8.4: Первая картинка

Рис. 8.5: Вторая картинка

В ходе проведения эксперимента был реализован следующий алгоритм:

1. Снимаем с исходного изображения волосы и головной убор используя HairMapper.

2. Запускаем HairFlash, подставив вместо source изображения получившееся.

Для начала удостоверимся, что описанная выше последовательность действий не испортила метод:

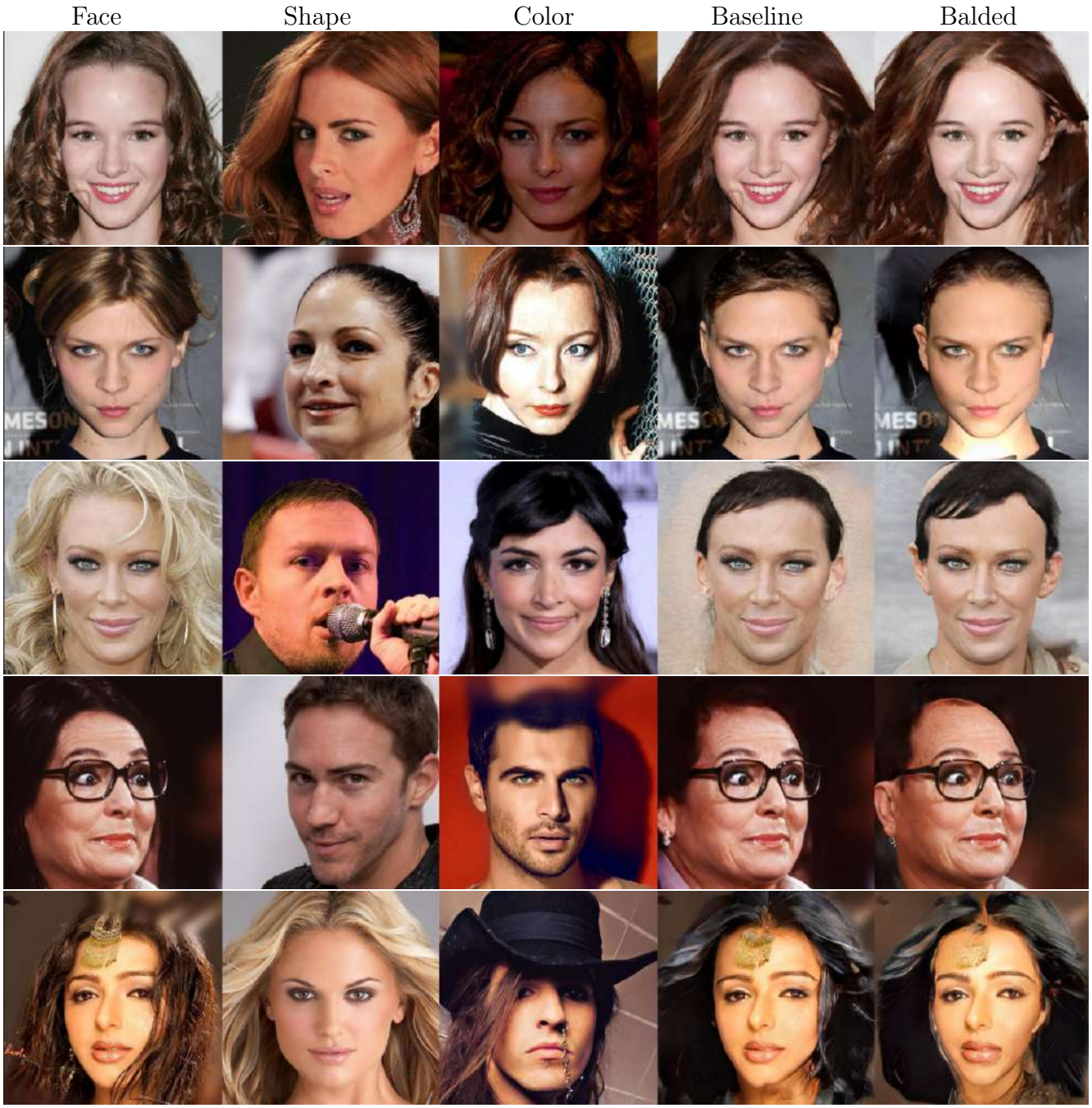

Как можно заметить, модели и правда удается снимать головные уборы, сохраняя при этом качество генерации.

Однако, количественные метрики и время генерации ухудшились, как мы можем видеть в таблице 8.5

| | FID | | | | | $\text{FID}_{\text{CLIP}}$ | | | | | Time(s) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | rec | full | color | shape | both | rec | V100 |
| Baseline | **15.08** | **21.77** | **25.33** | **24.44** | **9.79** | **5.80** | **3.56** | **5.78** | **6.66** | **1.63** | **1.97** |
| WithBalding | 18.79 | 23.32 | 27.02 | 26.91 | 10.77 | 6.24 | 4.12 | 7.07 | 7.59 | 1.91 | 3.91 |

Таблица 8.5: Сравнение метрик бейзлайна и экспериментальной модели. Жирным выделен лучший результат.

Рассмотрим несколько неудачных примеров, чтобы понять, почему метрики ухудшились.




Анализ:

- В первых двух случаях видны артефакты: в первом на волосах справа около уха, а во втором на шее. Известно, что артефакты довольно сильно влияют на метрики.

- В третьем и четвертом случаях мы можем быть свидетелями ситуации, когда модель не справилась с тем, чтобы закрасить всю лысину. Это сделало получившиеся картинки не очень красивыми и нереалистичными.
- В пятом примере HairMapper плохо справился с дорисовкой лица, после того как мы убрали волосы. Из-за этого пострадало и конечное изображение.

Вывод: снимать головные уборы, конечно, полезная идея, потому что подобные изображения иногда встречаются. Но как мы можем заметить это довольно сильно повлияло на качество и время генерации в худшую сторону. Поэтому от этой идей пользоваться в итоговой модели не будем.

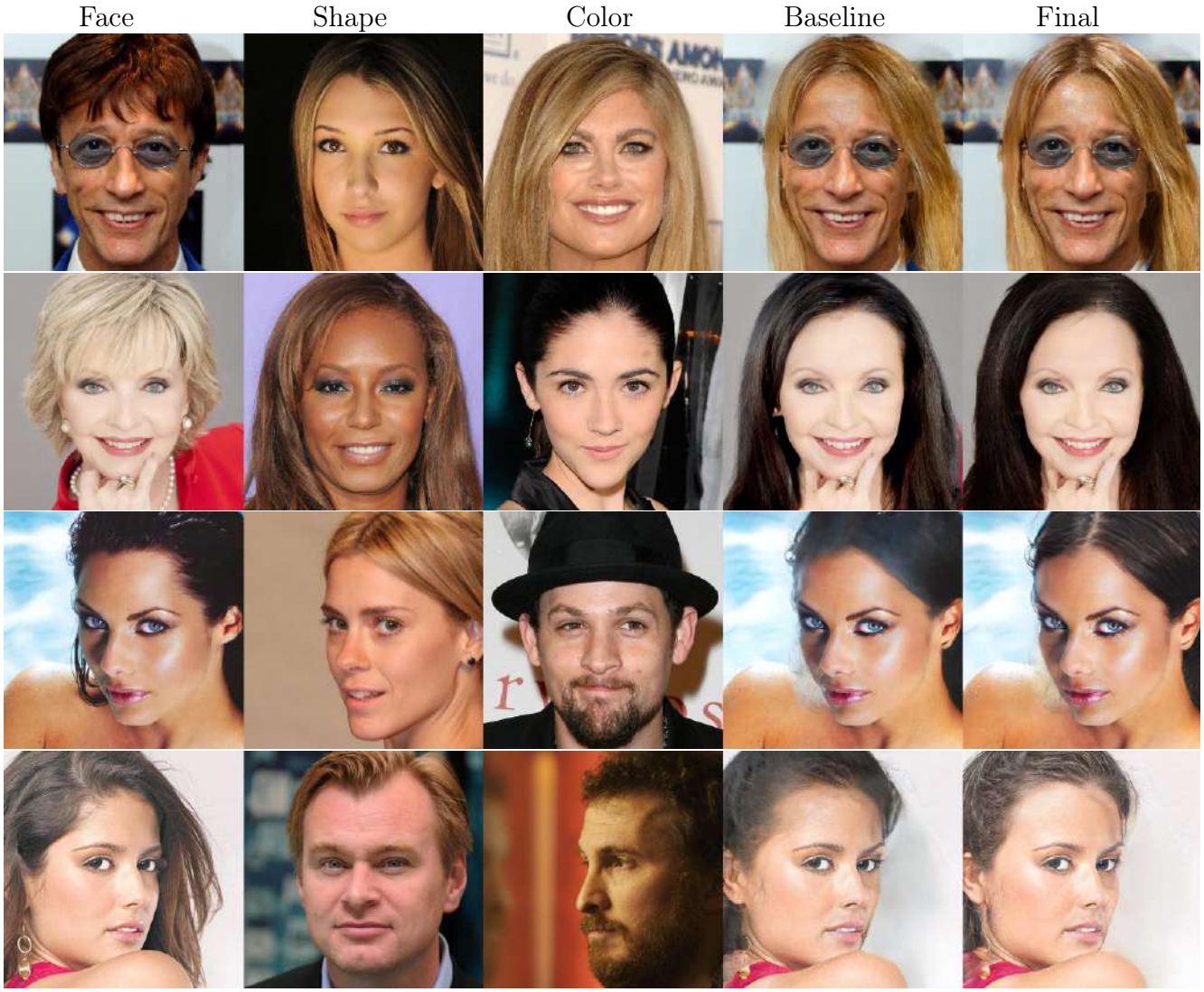
# 9 Итоговая модель

В качестве итоговой(Final) модели я соединил базовый метод и результаты первых двух гипотез. То есть теперь метод будет быстрее за счет того, что не потребуется искать выравненное color изображения. И в то же время будет лучше переносить текстуру волос за счет добавления embedding'ов волос shape изображения во вход blending encoder'a и добавления нового лосса, основанного на косинусном расстоянии и фильтре Собеля. Проведем качественное и количественное сравнение с baseline'ом, чтобы удостовериться в том, что нам удалось улучшить метод.
Количественное сравнение представлено в таблице 9.1. Видно, что большинство метрик улучшилось.

| | FID | | | | | $FID_{CLIP}$ | | | | | Time(s) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | full | color | shape | both | rec | full | color | shape | both | rec | V100 |
| Baseline | 15.08 | 21.77 | 25.33 | 24.44 | **9.79** | **5.80** | 3.56 | **5.78** | 6.66 | **1.63** | 1.97 |
| Final | **14.99** | **21.75** | **25.26** | **24.21** | 10.15 | 5.89 | **3.52** | 5.87 | **6.62** | 1.75 | **1.63** |

Таблица 9.1: Сравнение метрик baseline'a и финальной модели. Жирным выделен лучший результат.

Теперь посмотрим на качественное сравнение, чтобы убедиться, что наша модель не стала работать хуже на тех примерах, на которых она уже работала хорошо, и научилась переносить текстуру на сложных примерах.




Анализ:

- На первых двух изображениях видно, что наша модель не стала хуже и по прежнему хорошо отрабатывает там, где и ранее показывала хороший результат.
- На 3 и 4 изображениях видно, что наша модель начала лучше генерировать текстуру волос на получающихся изображениях.

**Заключение.** В результате проделанной работы:

1. Был переписан в многопоточной манере скрипт для тренировки модуля блендинга.
2. Был проведен ряд экспериментов, тестирующих гипотезы по улучшению модуля blending'a.
3. Был улучшен метод для переноса прически HairFlash. Теперь он более качественно переносит текстуру волос и работает на 20% быстрее.

# Список литературы


[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, и Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

[2] Tero Karras, Samuli Laine, Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision и pattern recognition, pages 4401–4410, 2019.

[3] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, и др.

[4] Wentao Jiang, Si Liu, Chen Gao, Jie Cao, Ran He, Jiashi Feng, и др.

[5] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, и Matthias Zwicker. Faceshop: Deep sketch-based face image editing. arXiv preprint arXiv:1804.0897 2018.

[6] Zhentao Tan, Menglei Chai, Dongdong Chen, Jing Liao, Qi Chu, Lu Yuan, Sergey Tulyakov, и Nenghai Yu. Michigan: multi-input-conditioned hair image generation for portrait editing. arXiv preprint arXiv:2010.16417, 2020.

[7] Sasikarn Khwanmuang, Pakkapon Phongthawee, Patsorn Sangkloy, Supasorn Suwajanakorn. Stylegan salon: Multi-view latent optimization for pose-invariant hairstyle transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision и Pattern Recognition, pages 8609-8618, 2023.

[8] Rohit Saha, Brendan Duke, Florian Shkurti, Graham W Taylor, Parham Aarabi. Loho: Latent optimization of hairstyles via orthogonalization. In Proceedings of the IEEE/CVF Conference on Computer Vision и Pattern Recognition, pages 1984–1993, 2021.

[9] Taewoo Kim, Chaeyeon Chung, Yoonseo Kim, Sunghyun Park, Kangyeol Kim, и Jaegul Choo. Style your hair: Latent optimization for pose-invariant hairstyle transfer via local-style-aware hair alignment. In European Conference on Computer Vision, pages 188–203. Springer, 2022.

[10] Peihao Zhu, Rameen Abdal, John Femiani, и Peter Wonka. Barbershop: Gan-based image compositing using segmentation masks. arXiv preprint arXiv:2106.01505, 2021.

[11] Xuyang Guo, Meina Kan, Tianle Chen, и Shiguang Shan. Gan with multivariate disentangling for controllable hair editing. In European Conference on Computer Vision, pages 655–670. Springer, 2022.

[12] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Zhen tao Tan, Lu Yuan, Weiming Zhang, и др.

[13] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, и Nong Sang. 2018. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. Lecture Notes in Computer Science (2018), 334–349.

[14] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, и Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. ACM Transactions on Graphics (TOG), 40:1 – 14, 2021.

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, и Ilya Sutskever. Learning transferable visual models from natural language supervision. In ICML, 2021.

[16] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, и Peter Wonka. 2020b. Improved StyleGAN Embedding: Where are the Good Latents? arXiv:2012.09036 [cs.CV].

[17] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive nor- malization. In Proceedings of the IEEE/CVF conference on computer vision и pattern recognition, pages 2337–2346, 2019.

[18] Xu Yao, Alasdair Newson, Yann Gousseau, Pierre Hel lier. A style-based gan encoder for high fidelity reconstruc tion of images и videos. European conference on computer vision, 2022.

[19] Yiqian Wu, Yong-Liang Yang и Xiaogang Jin. “HairMapper: Removing Hair From Portraits Using GANs”. В: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Июнь 2022, с. 4227—4236.