Санкт–Петербургский государственный университет

Черногорский Федор Евгеньевич

Выпускная квалификационная работа

Применение style-transfer моделей для генерации искусственных данных в задаче генерации 3D-моделей персонажей по скетч рисункам

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2018 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Современное программирование»

Научный руководитель:
профессор факультета Математики и компьютерных наук, д.ф.-м.н. Александр Сергеевич Куликов

Рецензент:
разработчик,
ООО «Яндекс Беспилотные технологии»
Степан Максимович Конев

Санкт-Петербург
2022 г.

Содержание

Введение

В наши дни трудно представить себе человека, который не пользуется какими-либо электронными устройствами. Несмотря на это, нам важны не устройства сами по себе, а скорее контент, доступ к которому они предоставляют. Этот контент может принимать различные формы, однако наиболее часто он представляется в виде визуальной информации. Большую часть подобного рода информации можно отнести к тем или иным разновидностям компьютерной графики.

Производство графики представляет собой комплексный процесс. В первую очередь на свет появляются концепты и наброски, представляющие собой первое видение будущего произведения или его части. По существующим концепт-артам создаются 3D модели. 3D-художник вручную переносит содержание двумерного изображения в трехмерное пространство. На построенную модель накладываются текстуры. В зависимости от формы и содержания конечного продукта в пайплайн может добавиться труд аниматоров, скульпторов, риггеров и других специалистов.

Создание трехмерной компьютерной графики является длительным и дорогим процессом, включающем в себя труд множества людей.

В настоящее время существуют различные разновидности программного обеспечения, стремящиеся облегчить труд цифровых художников. К ним относятся пакеты для 3D моделирования, скульптинга, текстурирования и другие. Большинство из них можно настроить с помощью различных плагинов и сделать процесс работы с 3D более удобным.

В связи с быстрым развитием методов, основанных на анализе данных, логично попытаться применить их к задачам, решаемым в процессе создания компьютерной графики. В частности, попытаться внедрить нейронные сети в пайплайн создания 3D модели или анимации для того, чтобы освободить художника от рутинных задач.

Однако данный подход встречается с некоторыми трудностями на своем пути к реализации: для обучения моделей, основанных на анализе данных, требуется большое количество этих самых данных. К сожалению, для решения задач, связанных с обработкой рисунков с человекоподобными персонажами

существует мало размеченных наборов данных. Это оправдано, так как для получения разметки по рисункам в большинстве случаев необходим труд профессионалов. Например в задаче реконструкции позы персонажа по рисунку для каждого элемента данных необходимо провести ручную реконструкцию позы, что, в свою очередь, является довольно трудоемким процессом, который подвластен не каждому.

Не стоит забывать, что при рисовании художники не используют точные математические измерения для ортографической или перспективной проекции, вместо этого полагаясь на свой опыт и эмпирические правила [16, 36]. В связи с этим художники часто искажают пропорции конечностей и используют нелинейную перспективу [32]. Кости персонажей часто выходят за пределы своей нормальной длины на рисунках из-за неточностей рисунка или художественной лицензии [36, 16, 34]. Все это не позволяет использовать искусственные данные в необработанном формате, сгенерированные как есть.

В данной работе мы хотим преодолеть ограничения, накладываемые на применение нейронных сетей в работе с реалистичными изображениями, путем создания искусственных данных на основе существующих 3D-моделей [2]. Эти искусственные данные должны, быть похожи на реальные рисунки, чтобы затем на них имело смысл проводить обучение нейронных сетей для решения различных задач, требующих размеченных датасетов. Для создания датасета предлагается воспользоваться современными моделями переноса стиля между изображениями [14]. Перенос стиля позволит нам использовать обширные базы с 3D моделями для создания правдоподобных набросков с соответствующими им разметками, полученными непосредственно из 3D моделей.

Предполагается исследовать применимость подобного подхода к генерации искусственных данных в задаче реконструкции 2D скелета персонажа по рисунку. В связи с тем, что персонажи на рисунках часто имеют нереалистичные или сильно искаженные пропорции также мы хотим сравнить методы переноса стиля с применением пространственных аугментаций.

Постановка задачи

Целью данной работы является поиск метода генерации искусственных данных, подражающих наброскам реальных художников. Получение правдоподобных набросков предполагается осуществлять с помощью нейросетевых моделей переноса стиля [14]. Помимо этого предполагается оценить полезность использования сгенерированных данных как аугментации при решении задачи реконструкции 2D позы по изображениям. Для достижения поставленной цели требуется решить следующие задачи.

- Изучить существующие подходы для генерации правдоподобных набросков.
- Собрать датасет отрендеренных 3D моделей с сохранением информации о скелете.
- Рассмотреть популярные методы переноса стиля и применить их к отрендеренным изображениям для имитации стиля набросков.
- Рабочие методы из предыдущей задачи использовать для создания датасета правдоподобных скетчей.
- Оценить качество полученных датасетов на задаче определения 2D позы.

1. Обзор предметной области

В исследованной литературе встречается применение моделей переноса стиля в качестве аугментации данных при обучении моделей компьютерного зрения [29]. В нашей работе мы изучим применимость подобного подхода для решения задачи генерации искусственных правдоподобных набросков, а также проверим применимость подобного подхода в качестве аугментации в задаче определения 2D позы персонажа по рисунку.

Задача, поставленная в процессе работы, не является широко известной, однако уже предпринимались попытки ее решения. В статье [30] авторы реализовали подход для очистки набросков от загрязнений, то есть решили задачу, обратную нашей. Эта статья является продолжением их исследования [31], в котором они решали данную задачу с помощью обычной сверточной сети [24]. Сеть представляла из себя несколько сверточных слоев, которые сначала сжимали изображение, а затем разжимали его, то есть сеть являлась своего рода автокодировщиком. В статье [30] авторы улучшили модель, добавив в нее состязательные аугментации. Модель увеличили, добавив в нее сеть-дискриминатор, которая училась различать реальные рисунки и те, которые сгенерировала модель - автокодировщик.

В той же статье был предложен способ для генерации карандашных набросков по очищеному изображению. Поменяв местами входные и выходные данные авторы добились генерации правдоподобных рисунков. Пример работы данного алгоритма можно увидеть на рис. 1.

Однако подход, описанный в статье [30] оказался невоспроизводим, так как для обучения модели необходимо наличие размеченных данных, которые не были выложены в открытый доступ авторами статьи. При этом во время обучения автокодировщику передается пара из наброска и уже очищенного изображения. К сожалению таких данных нет в открытом доступе или в достаточном количестве.

Основной задачей, решаемой в данной работе является задача переноса стиля между изображениями [14]. В стандартной постановке задачи мы имеем два изображения, одно из которых содержит контент, который мы хотим сохранить, а второе изображение является источником стиля, который

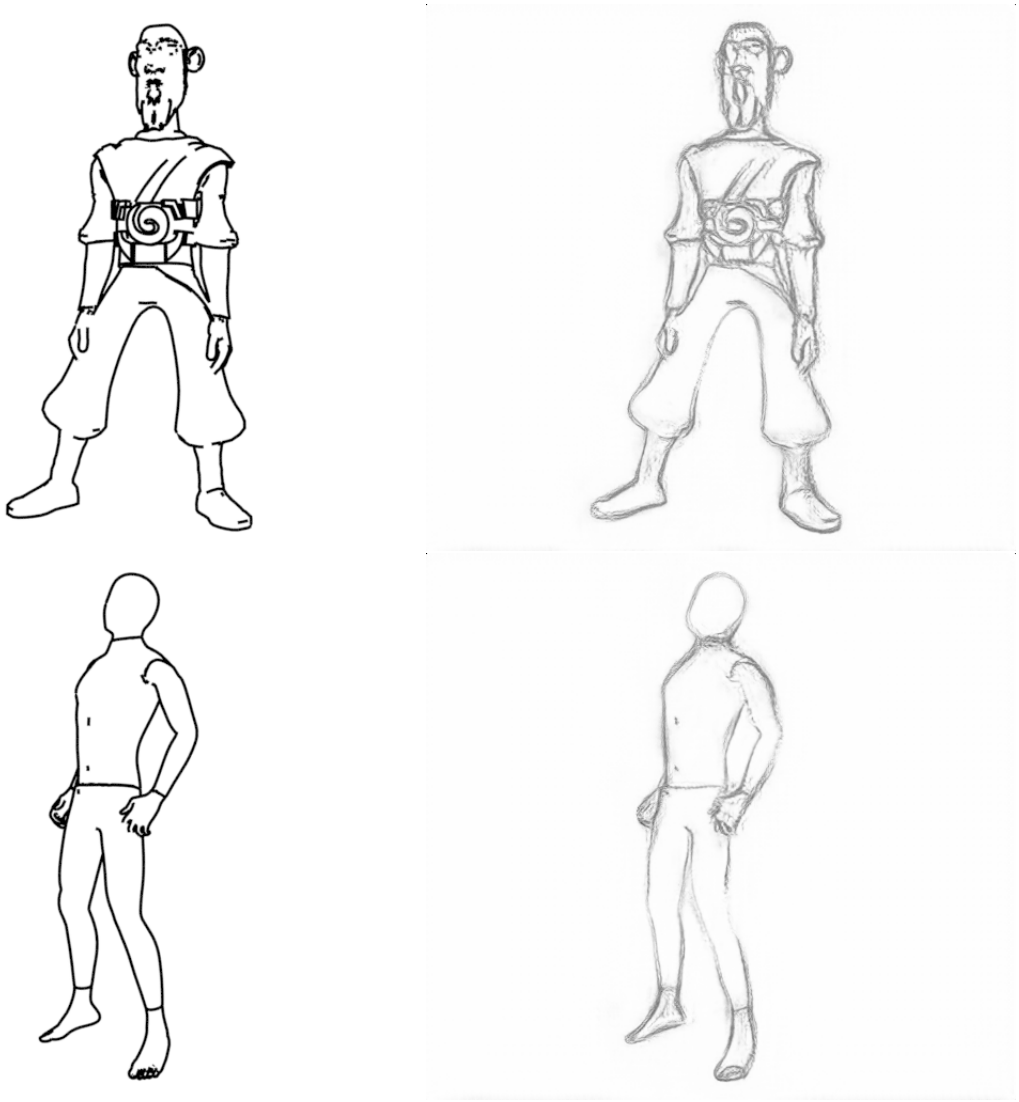

Рис. 1: Пример работы алгоритма из статьи [30, Simo-Serra]

мы хотим скопировать и перенести на изображение с контентом. Существуют различные подходы к решению данной задачи [37]. В процессе работы мы сконцентрировали свое внимание на решениях, которые используют нейронные сети и глубокое обучение для реализации переноса стиля между изображениями.

Большинство современных методов переноса стиля используют в своей архитектуре генеративно состязательные сети [15]. Данная разновидность нейронных сетей состоит из двух компонент: генератора и дискриминатора. Сеть-дискриминатор в процессе обучения старается научиться различать между собой реальные примеры данных и те, которые были сгенерированы

сетью-генероатором. Генератор же, в свою очередь, старается обмануть дискриминатор, для этого он обучается генерировать элементы данных наиболее похожие на то, что было представлено в обучающей выборке.

Помимо генеративно состязательных сетей в данной работе мы использовали модели, основанные на специальных Adaptive Instance Normalization слоях [17], а также на механизме визуального внимания [12]. Подробнее об этих подходах можно прочитать в соответствующих разделах 3.4, 3.5.

2. Описание исходных данных

В процессе подготовки данной работы было проведено исследование существующих наборов данных скетчей. Было обнаружено несколько датасетов для работы с набросками [13, 28], однако ни один из них не содержал в себе набросков человекоподобных персонажей.

В нашей работе решено было использовать методы переноса стиля для получения правдоподобных набросков. Для осуществления переноса стиля необходимо иметь два подмножества изображений: контентные и стилевые. К контентным относятся изображения, содержащие в себе персонажа. При этом контентному изображению должен быть сопоставлен набор данных, полученных из трехмерной модели, который необходим для обучения модели машинного обучения на искусственных данных. К стилевым же изображениям относятся те, с которых мы хотим перенести стиль. В нашем случае стилевыми изображениями будут являться рисунки реальных художников.

Для решения нашей задачи в первую очередь необходимо было получить доступ к изображениям персонажей с соответствующей разметкой - контентным данным. В задаче определения 2D позы такой разметкой являются координаты ключевых точек скелета на изображении. Для получения необходимых данных было решено воспользоваться набором данных Mixamo [2]. Mixamo представляет собой открытый набор трехмерных персонажей, в нем представлено множество различных моделей готовых для анимирования, а также большое количество анимаций для каждого персонажа. Анимации - это последовательный набор поз. При этом поза параметризуется углами суставов скелета.

Из всего многообразия моделей представленных в датасете Mixamo было выбрано несколько персонажей и набор анимаций для каждого из них. Далее был произведен сбор данных, похожий на описанный в статье [22]. С помощью набора инструментов для компьютерной графики с открытым исходным кодом Blender [10] был произведен рендеринг изображений. При рендеринге для обеспечения разнообразия поз персонажей использовался каждый 10 кадр анимации. При этом рендеринг производился с 8 различных позиций камеры. Полигональная сетка каждой модели настраивалась так,

чтобы при рендеринге были видны только контурные линии, а все пространство между ними оставалось прозрачным. Пример полученных изображений можно найти на рис. 2.

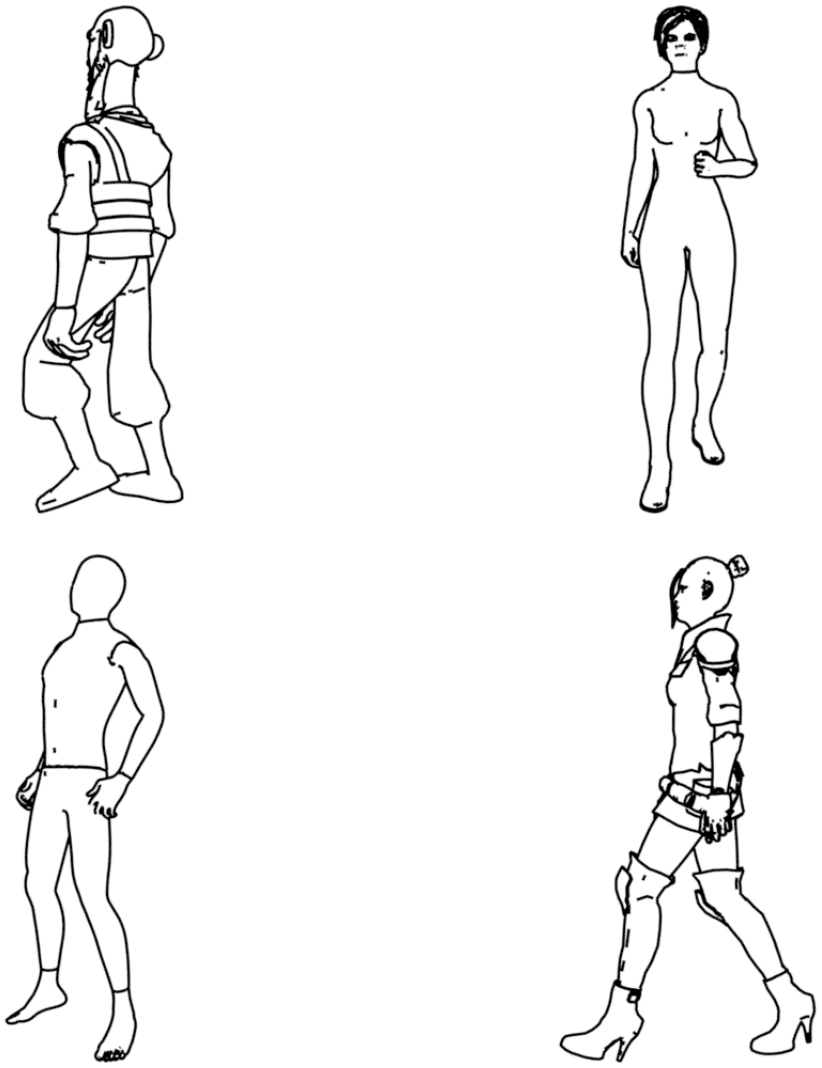

Рис. 2: Пример изображений из отрендеренного датасета

Для сохранения информации о скелете каждой модели, совпадающей с полученным силуэтом персонажа производилась перспективная проекция трехмерного скелета на текущую камеру.

В результате было получено порядка $200,000$ изображений со скелетами.

Подобный метод сбора данных позволяет, опираясь на обширные базы данных существующих моделей, генерировать искусственные наборы для

обучения нейросетевых моделей, решающих задачи, в которых необходима информация о 3D модели, соответствующей рисунку. Однако чистые рендеры не всегда похожи на реальные наброски (см. рис. 3), поэтому далее в работе мы постараемся приблизить их к таковым.

Помимо контентных изображений, для работы с моделями переноса стиля необходимо наличие датасета стилевых изображений, которые используются либо при обучении модели переноса стиля, либо как единичные примеры для моделей, обучение которых не требуется. В качестве данного набора данных мы собрали $13,000$ изображений в поисковых системах по релевантным запросам.

В данном датасете представлены наброски силуэтов человекоподобных персонажей в различных позах, нарисованные многими художниками в различных стилях. Примеры изображений из данного датасета можно найти на рис. 3.

Рис. 3: Пример изображений из датасета реальных скетчей

3. Применение существующих моделей

В данном разделе мы рассмотрим особенности некоторых моделей, применяющихся при решении задач переноса стиля между изображениями, а также опишем результаты их применения к задаче переноса стиля скетча на отрендеренное изображение.

Существует множество различных подходов к решению задачи переноса стиля между изображениями [37]. Однако большинство из них требуют для работы наличия набора размеченных данных, то есть набора пар, где каждому изображению соответствует его вариант с перенесенным стилем, как например в работе [19]. В нашем случае подобный датасет отсутствует, поэтому в данной работе мы сконцентрируемся на моделях, решающих задачу unpaired image-to-image translation. Image-to-image translation - класс задач компьютерного зрения, в котором целью является обучить отображение между входным и выходным изображениями с помощью датасета из пар изображений. Более сложная вариация - unpaired image-to-image translation все также ставит перед нами задачу обучить отображение между различными видами изображений, однако теперь у нас имеются два набора изображений, не соотнесенных друг с другом.

Большинство работ, решающих задачу unpaired image-to-image translation представляют из себя различные вариации генеративно-состязательных сетей [15]. Подобные сети состоят, как правило, из двух частей - генератора и дискриминатора. В процессе обучения дискриминатор выступает в качестве бинарного классификатора, который учится различать подаваемые на вход реальные изображения от изображений, созданных сетью-генератором. Генератор же должен постепенно научиться создавать такие изображения, которые дискриминатор будет считать похожими на реальные примеры.

В процессе работы нами были обучены три модели, решающие задачу unpaired image-to-image translation, основанные на генеративно состязательной архитектуре. Все три модели показали неудовлетворительный результат, при решении задачи переноса стиля в нашей постановке. Причиной данной неудачи мы считаем малое количество обучающих данных, так как предобученные модели, которые предназначены для переноса стиля по одному

изображению показали более качественный результат, добившись генерации правдоподобных рисунков по рендеру и единственному примеру стиля.

3.1. Cycle Gan

В статье [38] описан один из первых подходов к решению задачи переноса изображения из исходного домена X в целевой домен Y при отсутствии спаренного набора данных. Целью данной модели является обучение такого отображения $G : X \to Y$, что распределение изображений из $G(X)$ неотличимо от распределения Y относительно состязательной функции потерь. Помимо этого вводится отображение $F : Y \to X$ совершающее преобразование обратное к G. Введение дополнительного отображения позволяет нам внести еще один компонент в общую функцию потерь, который отвечает за консистентность последовательного применения отображений F и G: $F(G(X)) \approx X$ и $G(F(Y)) \approx Y$. В дополнение были введены два дискриминатора D_X, D_Y, где D_X должен уметь различать элементы X и $F(Y)$, аналогично D_Y. Схему полученной модели можно увидеть на рис. 4.

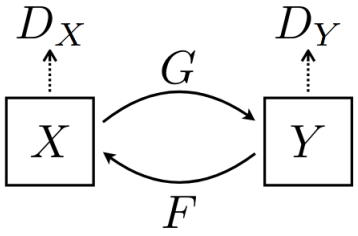

Рис. 4: Схематичное изображение модели CycleGAN [38].

Отображения F, G задаются сетями идентичной архитектуры описанной в статье [20]. Каждая сеть состоит из трех сверточных слоев, нескольких остаточных блоков, двух сверток с нецелым шагом и одного слоя отображающего фичи в RGB изображение. В качестве сетей-дискриминаторов D_X, D_Y была использована архитектура PatchGAN [18].

Состязательная функция потерь применялась к обоим отображениям. Так для $G : X \to Y$ и соответствующего дискриминатора D_Y состязательная функция потерь выглядит следующим образом:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(X)))]$$

Здесь $x \sim p_{data}(x)$ и $y \sim p_{data}(y)$ - распределения данных.

Помимо состязательной функции потерь дополнительно вводится функция потерь циклической согласованности, она отвечает за то, чтобы отображения, полученные в процессе обучения оказались взаимно обратными:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1]$$

Общая функция потерь выглядит следующим образом:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F)$$

Для эксперимента по переносу стиля с реальных рисунков на отрендеренные изображения была использована реализация на языке python [6]. В качестве домена X были использованы отрендеренные изображения трехмерных моделей, а в качестве домена Y - рисунки реальных авторов.

Модель обучалась на протяжении 100 эпох, в качестве алгоритма оптимизации использовался Adam [23].

В отличие от оригинальной статьи вместо классической состязательной функции потерь в процессе обучения использовался Least Squares GAN [26].

Пример результатов обучения модели CycleGAN на нашем наборе данных можно увидеть на рис. 5. На изображении видно, что модель очень качественно проводит реконструкцию изображения переводя изображения **realA** в **recA**, а **realB** в **recB**, однако при этом на сгенерированных $F(y)$ и $G(x)$ изображениях (**fakeB**, **fakeA**) появляются странные артефакты, совершенно не похожие на переносимый стиль. Подобные артефакты появлялись на всех результатах работы модели, в связи с этим результат обучения оказался неудовлетворительным, а данная модель в нашей постановке задачи не показала достаточного качества.

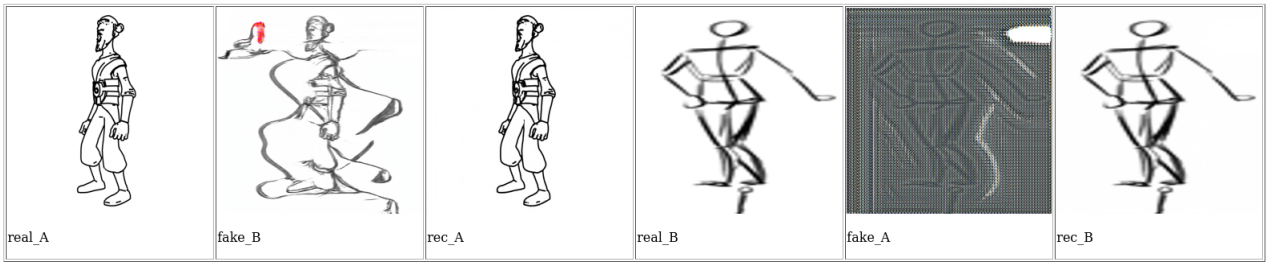

Рис. 5: Результаты обучения CycleGAN.

Учитывая, что в репозитории и статье приводятся примеры качественной работы Cycle GAN, можно сделать вывод, что основной проблемой в работе данной модели является разреженность данных на которых происходит обучение. По сравнению с RGB изображениями, работа с которыми происходит в статье, наши данные являются сильно разреженными, так как большую часть изображения занимает белый фон, а семантика изображения содержится в тонких черных линиях.

3.2. Contrastive Unpaired Translation

Второй моделью избранной для экспериментов была модель CUT [27]. Данная модель предполагает, что каждый патч выходного изображения должен отражать контент соответствующего патча входного изображения. Для достижения этой цели авторы предлагают максимизировать взаимную информацию между соответствующими патчами используя контрастное обучение.

Данные для модели CUT все также представляют из себя два независимых набора $X = \{x \in \mathcal{X}\}, Y = \{y \in \mathcal{Y}\}$, где $\mathcal{X} \subset \mathbb{R}^{H \times W \times 3}, \mathcal{Y} \subset \mathbb{R}^{H \times W \times 3}$ - исходный и результирующий домены изображений соответственно.

В отличие от CycleGAN в модели CUT сеть-генератор G представляет из себя комбинацию двух сетей - энкодера и декодера, таким образом $G(x) = G_{dec}(G_{enc}(x))$.

В модели использовалась стандартная состязательная функция потерь:

$$\mathcal{L}_{GAN}(G, D, X, Y) = \mathbb{E}_{y \sim Y}[\log D(y)] + \mathbb{E}_{x \sim X}[\log(1 - D(G(x)))]$$

Для максимизации взаимной информации между входным и выходным изображениями использовался метод контрастной оценки шума.

$$L(v, v^+, v^-) = -\log \left[\frac{exp(v \cdot v^+ / \tau)}{exp(v \cdot v^+ / \tau) + \sum_{n=1}^{N} exp(v \cdot v_n^- / \tau)} \right]$$

- стандартная кросс-энтропия, показывающая вероятность того, что позитивный пример будет выбран среди всех представленных, где $v \in \mathbb{R}^K$ - вектор запроса, $v^+ \in \mathbb{R}^K$ - позитивный пример, $v^- \in \mathbb{R}^{N \times K}$ - матрица негативных примеров. В контексте модели вектор запроса соответствует вектору, описывающему патч на выходном изображении, вектор позитивного примера - описание соответствующего патча входного изображения, а матрица негативных примеров соответствует описаниям других патчей входного изображения.

Для определения описания патча была введена дополнительная двухслойная сеть H_l, которая брала карту признаков после применения l слоев сети $G_{enc}(x)$ и преобразовывала их в набор векторов признаков для каждой пространственной локации на изображении. Затем тот же трюк производился с результатом $G_{enc}(G(x))$, брались все возможные патчи и для каждого из них считался функционал L. Затем все суммировалось по всем возможным патчам и по всем нужным значениям количества используемых слоев G_{enc} и использовалось как одна из компонент функции потерь вместе с $\mathcal{L}_{GAN}$.

Схематичное изображение модели можно найти на рис. 6.

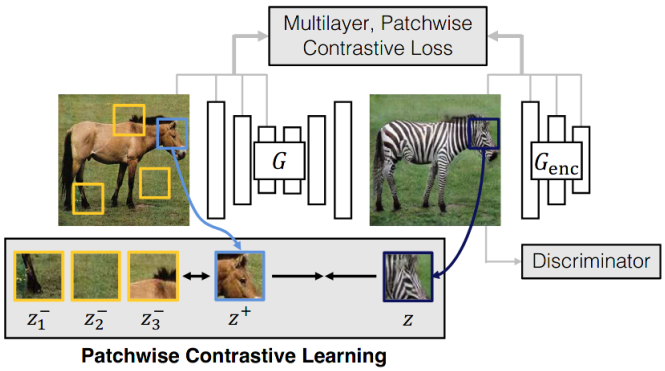

Рис. 6: Схематичное изображение модели CUT [27].

Эксперимент по переносу стиля с реальных рисунков на отрендеренные изображения проводился с помощью официальной pytorch реализации [5].

Так как модель предоставлена теми же авторами, что и CycleGAN настройки эксперимента практически не отличались, мы все также выбрали около 30,000 изображений рендеров и 13,000 реальных рисунков. Обучение модели производилось на протяжении 200 эпох. Финальные результаты обученной модели можно видеть на рис. 7.

В результате обучения модель научилась генерировать 2 типа изображений: либо сглаженные черные пятна, отдаленно напоминающие изначальное изображение, но полностью игнорирующее желаемый стиль, либо набор линий, некоторым образом схожий стилистически с изображением, с которого стиль переносился, однако потерявший любую схожесть с контентным изображением.

Как и в случае с моделью CycleGAN, модель CUT не оправдала на-

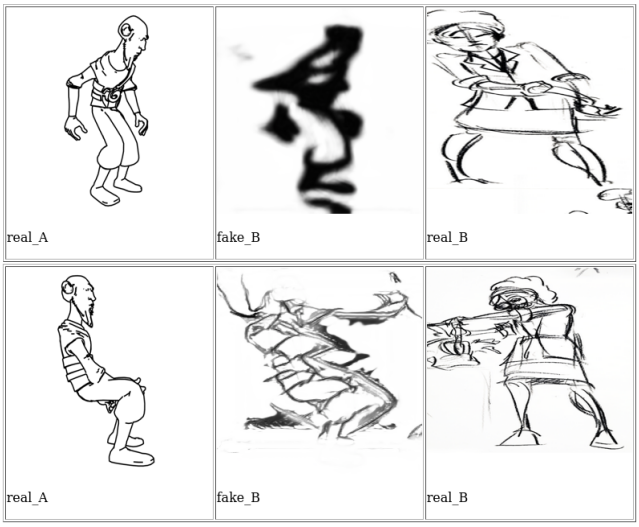

Рис. 7: Результаты обучения модели CUT.

ших ожиданий в применении к задаче переноса стиля и обучение на наших данных не помогло нам научить ее переносить стиль между разреженными изображениями.

3.3. UGATIT

Модель UGATIT [21] представляет из себя классическую состязательную нейронную сеть с добавлением вспомогательных классификаторов и специального AdaLIN слоя.

Пусть $x \in \{X_s, X_t\}$ - сэмпл данных из исходного и целевого доменов. Сеть-генератор в данном случае состоит из трех частей: энкодера E_s, декодера G_t и вспомогательного классификатора η_s, где $\eta_s(x)$ выдает вероятность того, что x пришел из X_s. При этом вспомогательный классификатор использует внутри признаки извлеченные из изображения разными слоями E_s. Использование весов данного классификатора впоследствии позволяет посчитать карту признаков внимания, на которой затем запускается декодер G_t.

Помимо этого в генератор также встроены специальные AdaLIN слоя, которые являются интерполяцией между Instance Normalization и Layer Normalization слоями.

$$AdaLIN(a, \gamma, \beta) = \gamma \cdot (\rho \cdot \frac{a - \mu_I}{\sqrt{\sigma_I^2 + \epsilon}} + (1 - \rho) \cdot \frac{a - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}}) + \beta$$

В данной формуле μ_I, μ_L и σ_I, σ_L представляют собой внутриканальные и внутрислойные среднее и стандартное отклонение соответственно. γ, β - параметры, генерируемые полносвязным слоем из карты признаков внимания, а ρ - коэффициент отвечающий за выбор между Instance Normalization и Layer Normalization.

В сети-дискриминаторе также находится дополнительный классификатор, который помогает добывать карты признаков внимания, на которых обучается бинарный классификатор, отличающий реальные изображения от сгенерированных. Общий вид модели можно найти на рис. 8.

Как и в предыдущих двух случаях для запуска экспериментов использовался код, предоставленный авторами статьи [9]. Модель запускалась в стандартной конфигурации на 1,000,000 итераций. В качестве исходного домена использовались отрендеренные изображения, а в качестве целевого домена -

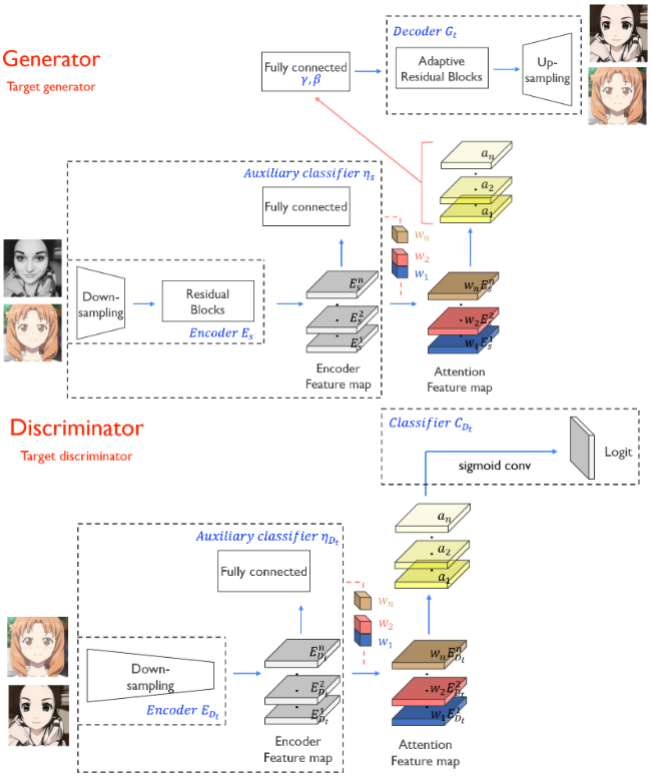

Рис. 8: Схематичное изображение модели UGATIT [21].

рисунки реальных художников.

К сожалению по результатам обучения данная модель не научилась выполнять возложенную на нее задачу. Ей так и не удалось перенести стиль с реальных рисунков, сохранив при этом семантику рендеров. Как видно на рис. 9 модель действительно добавляет линии, которых изначально не было на рендере, что делает результат ее работы похожим на неаккуратный набросок. В некоторых случаях ей даже удается сохранить образ визуально похожим на изначальное изображения. Однако в большинстве случаев модель полностью теряет изначальную форму персонажа, заполняя результирующее изображение множеством трудно описываемых артефактов.

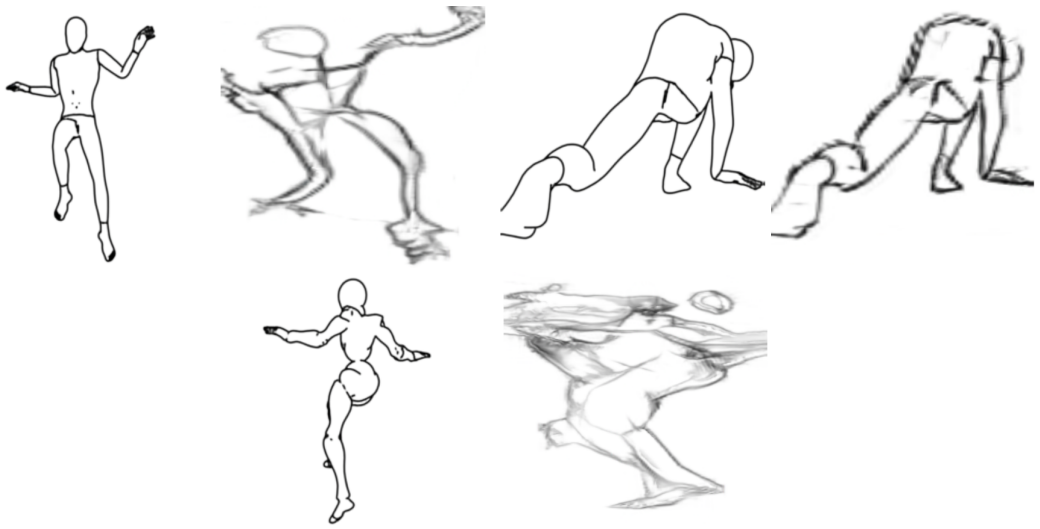

Рис. 9: Результаты обучения модели UGATIT.

3.4. AdaIn

Модель представленная в статье [17] являет собой одну из первых реализаций переноса стиля на изображение в потоковом формате. То есть для применения данной модели ее не обязательно обучать на наборе стилевых изображений, а достаточно одного изображения с примером. Подобная простота применения модели кроется в использовании специального Adaptive Instance Normalization слоя.

Как видно на рис. 10 модель состоит из нескольких энкодеров, в качестве которых использовалась предобученная архитектура VGG-19, а также одного декодера, который был реализован как инвертированная сеть VGG-19, где слои пулинга заменены на слои повышающие размерность за счет ближайших соседей (nearest up-sampling). Первый энкодер используется для переведения изображения в пространство признаков, в котором затем к изображению-контенту и изображению-стилю применяется Adaptive Instance Normalization.

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{1}$$

Из формулы 1 ясно, что слой AdaIN получает на вход признаковое описание контентного x и стилевого y изображений и заменяет межканальное значение среднего и дисперсии признаков x так, чтобы они совпадали со

значениями, соответствующими y.

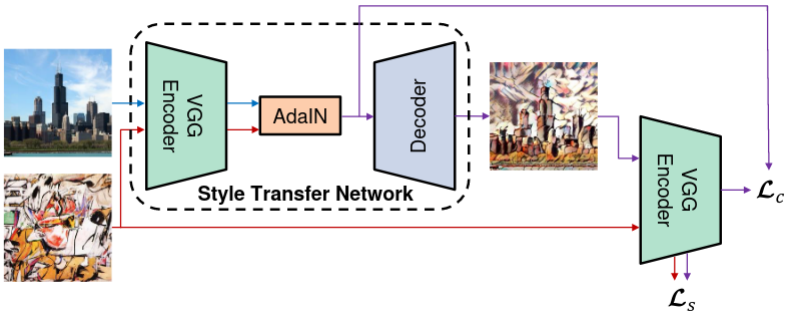

Рис. 10: Схематичное изображение модели AdaIN [17].

Второй энкодер используется для подсчета функций потерь. Контентная функция потерь описывается формулой 2, где t - результат AdaIN слоя, g - сеть-декодер, f - сеть энкодер для подсчета функции потерь. Мы измеряем евклидово расстояние между вектором признаков, полученном на выходе из AdaIN слоя и вектором признаков, посчитанном на реконструированном изображении.

$$\mathcal{L}_C = ||f(g(t)) - t||_2 \tag{2}$$

Формула 3 описывает стилевую функцию потерь. Здесь ϕ_i соответствует результату применения первых i слоев второго энкодера, а s - стилевому изображению. В данной функции потерь мы измеряем отклонение внутриканальных статистик реконструированного изображения от статистик, посчитанных на стилевом изображении.

$$\mathcal{L}_S = \sum_{i=1}^{L} ||\mu(\phi_i(g(t))) - \mu(\phi_i(s))||_2 + \sum_{i=1}^{L} ||\sigma(\phi_i(g(t))) - \sigma(\phi_i(s))||_2 \tag{3}$$

Для экспериментов нами была выбрана реализация модели AdaIN на языке python [1]. В репозитории поставлялась модель, предобученная на датасетах MS COCO [25] и WikiArt [7]. В качестве стилевых изображений были использованы несколько примеров из нашего набора с реальными рисунками. В качестве контентных изображений были использованы рендеры трехмерных

моделей. Результат применения модели AdaIN с тремя различными стилями можно увидеть на рис. 11.

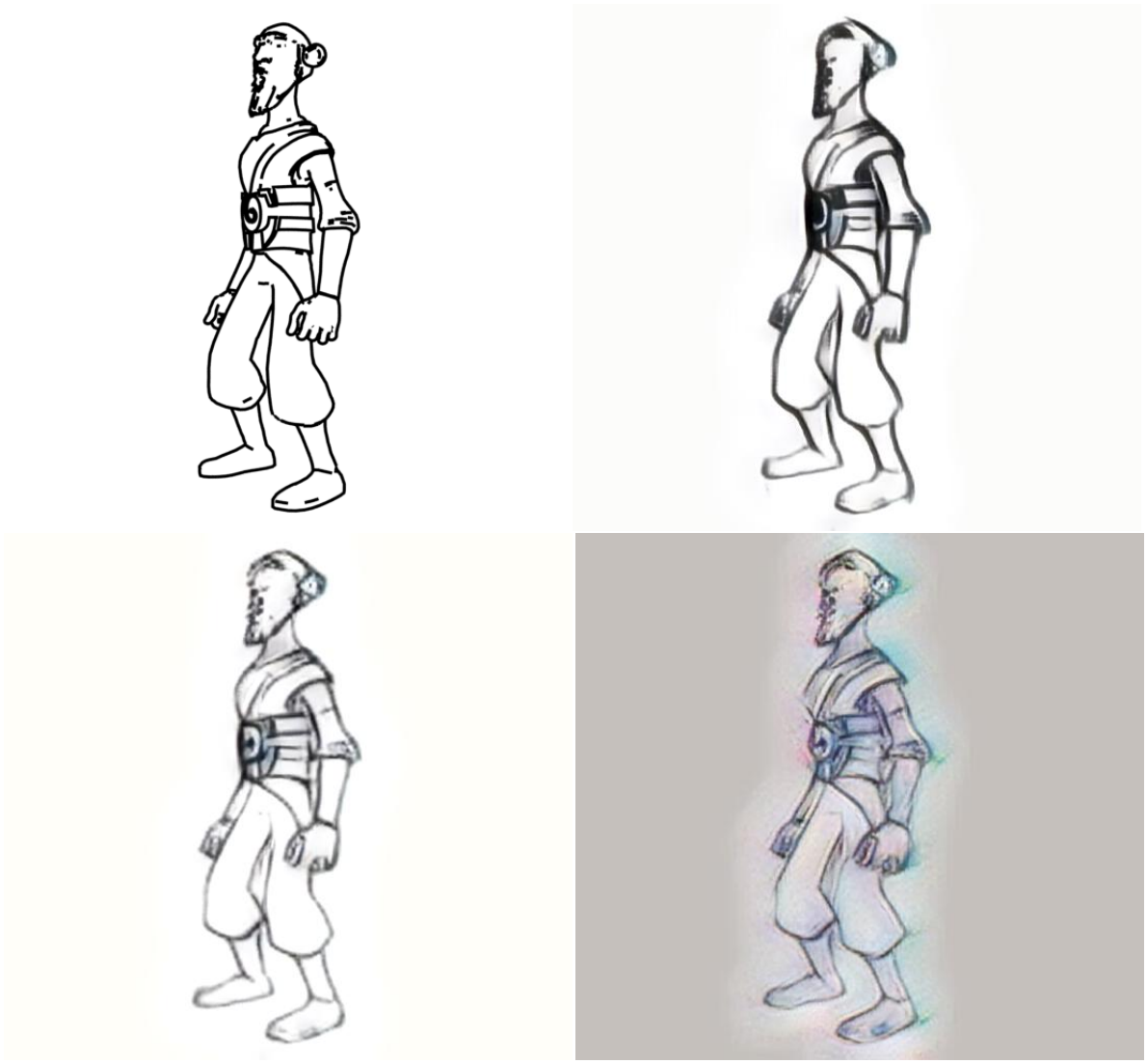

Рис. 11: Результаты полученные с помощью модели AdaIN с различными стилевыми изображениями.

AdaIN стал первой моделью, успешно перенесшей стиль с рисунков на рендеры. Помимо прочего данная модель позволила нам варьировать стиль результирующего изображения в зависимости от передаваемого стилевого изображения. В дальнейшем именно эта модель была использована нами для создания датасета правдоподобных рисунков.

3.5. $StyTr^2$

Последней моделью, использовавшейся для переноса стиля в нашей задаче, была модель, описанная в статье [12]. В данной статье авторы предлагают воспользоваться архитектурой трансформеров [35] в задаче переноса стиля между изображениями.

Как видно на рис. 12 изображения стиля и контента разбиваются на патчи (квадраты, вырезанные из исходного изображения). Затем патчи линейно проецируются в пространство признаков. Затем к признакам изображения-контента применяется контентно зависимое пространственное кодирование (CAPE) и они передаются на вход трансформеру-энкодеру. Патчи стилевого изображения передаются в другой энкодер. Затем результаты работы этих сетей передаются в многослойный трансформер-декодер, который старается стилизовать последовательность патчей контентного изображения в соответствии с описанием стиля, полученном из стилевых патчей. Затем результат работы трансформер-декодера передается в сверточный декодер, который повышает размерность выхода до размеров изначального изображения и выдает финальный результат.

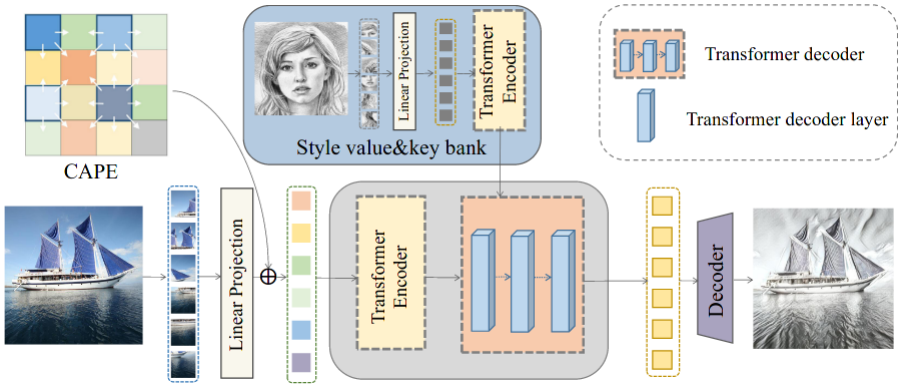

Рис. 12: Схематичное изображение модели $StyTr^2$ [12].

Для экспериментов была использована предобученная на датасетах MS COCO [25] и WikiArt [7] модель представленная в репозитории от авторов статьи [8].

На рис. 13 представлены результаты переноса различных стилей набросков на изображение-рендер. Как видно из примеров у модели получается

делать изображение более похожим на нарисованное, однако стили различаются между собой не так сильно как при применении модели AdaIN. Тем не менее данную модель мы также использовали для сбора датасета правдоподобных рисунков.

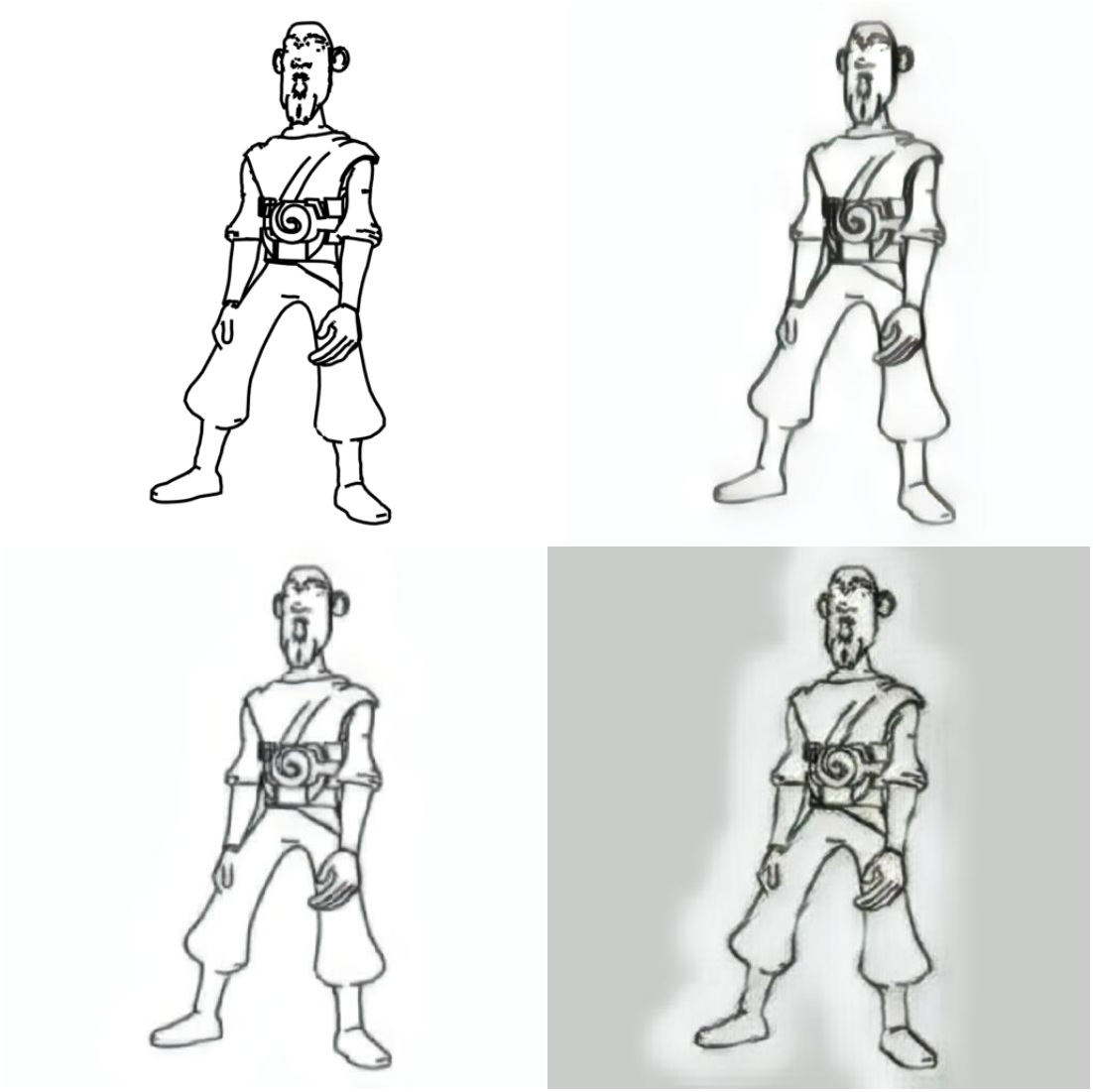

Рис. 13: Результаты полученные с помощью модели $StyTr^2$ с различными стилевыми изображениями.

4. Применение искусственных данных в задаче определения 2D позы

Для экспериментов нами была выбрана модель Deep High Resolution Net [33] - модель для определения 2D позы человека по изображению. Данная модель была предобучена на датасете ImageNet [11] для извлечения признаков из изображения, а также обучена на датасете COCO [4] для определения ключевых точек позы человека на изображении. В нашем эксперименте модель Deep High Resolution Net выступала в качестве оценивающего механизма, то есть мы обучали модель на различных наборах данных, а затем сравнивали качество обученных моделей на валидационной выборке, представляющей из себя более 800 реальных скетчей с размеченными ключевыми точками скелета. Качество работы модели оценивалось по метрике Mean Average Precision (MAP).

Для изучения возможности применения искусственных данных в задаче определения 2D позы персонажа было собрано несколько наборов искусственных данных.

В первую очередь был использован датасет, собранный из визуализированных изображений 3D моделей, взятых из набора данных Mixamo [2]. Было выбрано 5 персонажей и несколько десятков анимаций. Затем взят каждый десятый кадр анимации и визуализирован с помощью набора инструментов для компьютерной графики с открытым исходным кодом Blender [10]. Более подробное описание сбора датасета можно найти в главе 2.

Помимо этого для придания искусственным данным большей схожести с реальными рисунками были использованы модели AdaIN [17] и $StyTr^2$ [12]. Из нашего набора данных реальных неразмеченных скетчей было выбрано 12 различных изображений, каждое из которых было использовано для переноса стиля на собранный датасет визуализированных моделей Mixamo.

Итого в качестве искусственных данных для обучения модели Deep High Resolution Net были использованы:

- Набор визуализированных изображений Mixamo.
- Набор изображений с одним перенесенным стилем моделью AdaIN.

- Набор изображений с двенадцатью перенесенными стилями моделью AdaIN.

- Набор изображений с двенадцатью перенесенными стилями моделью $StyTr^2$.

Помимо искусственных данных для эксперимента были использованы аугментации [29], которые являются широко известным способом улучшить качество модели компьютерного зрения за счет внесения изменений в данные, а также помогают бороться с переобучением модели. Для создания аугментаций была использована библиотека Albumentations [3]. В качестве аугментаций были использованы афинные преобразования изображения, различные виды сглаживания, мультипликативный и гауссов шум.

В нашем понимании аугментации, помимо классической роли борьбы с переобучением, должны помочь отобразить в исходных данных изменения пропорций и перспективы, к которым зачастую прибегают художники в процессе создания скетчей.

Первый эксперимент представлял собой использование предобученной на ImageNet модели Deep High Resolution Net для обучения на скетчах, полученных непосредственно из процесса рендеринга, а также на скетчах полученных с помощью переноса стиля моделью AdaIN с реальных рисунков на скетчи, полученные из процесса рендеринга. Модель обучалась на протяжении 10 эпох, валидация происходила на отложенном наборе реальных рисунков с размеченными ключевыми точками.

Результаты данного эксперимента можно видеть в таблице 1. По таблице видно, что обучение на стилизованных скетчах дало ощутимый прирост качества модели по сравнению с обучением на скетчах без перенесенного стиля.

Датасет	MAP
sketch	0.33
$AdaIN_{12}$	0.40

Таблица 1: Результаты экспериментов с моделью Deep High Resolution Net, предобученной на датасете ImageNet.

В качестве второго эксперимента мы взяли два набора данных: набор рендеров и набор результатов применения AdaIN к рендерам. Обучили модели на этих датасетах на протяжении 10 эпох, взяли лучшие модели и запустили валидацию обеих моделей на отложенных выборках из представленных наборов данных. Получившиеся результаты представлены в таблице 2. В результате модель, обученная на датасете, обработанном с помощью AdaIN показала лучшее качество на обеих отложенных выборках, превзойдя модель, обученную на простых скетчах без имитации стиля.

	AdaIN	Sketch
AdaIn	0.993	0.938
Sketch	0.99	0.968

Таблица 2: Результаты перекрестного сравнения моделей обученных на сгенерированных скетчах и скетчах, обработанных моделью AdaIN с применением 12 стилей. Столбец соответствует набору данных, на котором модель обучалась, а строка - набору данных, на котором происходила валидация

В качестве еще одного эксперимента мы исследовали несколько моделей полученных из модели Deep High Resolution Net предобученной на датасете COCO с помощью дообучения на различных датасетах. В качестве бейзлайна была использована модель предобученная на COCO без дообучения. На валидационной выборке из реальных рисунков эта сеть выдала результат 0.55 MAP. Модели, дообученные на наших датасетах, полученных с помощью применения моделей AdaIN и $StyTr^2$ для переноса 12 стилей на весь датасет рендеров, показали результат ниже бейзлайна и ниже модели, обученной на неизмененных скетчах. Модель, обученная на наборе данных с одним перенесенным с помощью AdaIN стилем также показала результат ниже бейзлайна, однако показала себя лучше, чем обе модели с 12 перенесенными стилями.

Помимо этого была обучена модель с использованием аугментаций из библиотеки Albumentations, которая показала наилучший результат по сравнению со всеми остальными моделями добившись значений в 0.66 MAP. Результаты эксперимента представлены в таблице 3.

Из проведенных экспериментов можно сделать вывод, что применение переноса стиля для создания искусственных данных может дать прирост в

Эксперимент	MAP
baseline	0.55
clean sketch	0.62
$AdaIN_1$	0.54
$AdaIN_{12}$	0.51
$StyTr_{12}$	0.47
Augmentations	**0.66**

Таблица 3: Результаты экспериментов с моделью Deep High Resolution Net, предобученной на датасете MS COCO.

качестве при обучении моделей для реконструкции 2D позы. Однако при использовании предобученной модели наши искусственные данные снижают качество работы модели. При этом ощутимый прирост качества становится очевиден при использовании аугментаций, классических с точки зрения применения в компьютерном зрении.

Заключение

В рамках работы было изучено несколько архитектур нейронных сетей, позволяющих переносить стиль между изображениями. Эксперименты показали, что генеративно состязательные сети плохо приспособлены для работы с разреженными изображениями, какими являются например наброски.

Были отобраны архитектуры, позволяющие правдоподобно переносить стиль с реальных рисунков на изображения, сгенерированные по трехмерным моделям. Такими архитектурами оказались модели, основанные на специальных слоях, осуществляющих смешивание статистик контента и стиля.

Были собраны наборы данных правдоподобных скетчей с помощью моделей AdaIN и $StyTr^2$ с использованием различных стилевых источников. На данных наборах была обучена модель для определения 2D позы персонажа по изображению, которая показала себя не лучше предобученной модели и модели обученной на данных, непосредственно сгенерированных из трехмерных моделей.

Было изучено применение пространственных аугментаций при обучении модели определения 2D позы, которые позволили превзойти качество, полученное на данных без переноса стиля.

Список литературы

[1] Adain repository. https://github.com/naoto0804/pytorch-AdaIN.

[2] Adobe's mixamo. https://www.mixamo.com/#/.

[3] Albumentations. https://albumentations.ai/.

[4] Coco dataset. https://cocodataset.org/.

[5] Cut repository. https://github.com/taesungp/contrastive-unpaired-translation.

[6] Cyclegan repository. https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

[7] K. nichol. painter by numbers, wikiart. https://www.kaggle.com/c/painter-by-numbers.

[8] Style transfer with transformers repository. https://github.com/diyiiyiii/StyTR-2.

[9] Ugatit repository. https://github.com/znxlwm/UGATIT-pytorch.

[10] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Yingying Deng, Fan Tang, Xingjia Pan, Weiming Dong, Chongyang Ma, and Changsheng Xu. Stytrˆ 2: Unbiased image style transfer with transformers. *arXiv preprint arXiv:2105.14576*, 2021.

[13] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[14] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

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

[16] Burne Hogarth. *Dynamic Figure Drawing*. Watson-Guptill, 1996.

[17] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL: `http://arxiv.org/abs/1611.07004`, `arXiv:1611.07004`.

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[20] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[21] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. *arXiv preprint arXiv:1907.10830*, 2019.

[22] Kangyeol Kim, Sunghyun Park, Jaeseong Lee, Sunghyo Chung, Junsoo Lee, and Jaegul Choo. Animeceleb: Large-scale animation celebfaces dataset via controllable 3d synthetic models. *arXiv preprint arXiv:2111.07640*, 2021.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[26] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[27] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, 2020.

[28] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.

[29] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[30] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)*, 37(1):1–13, 2018.

[31] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

[32] Karan Singh. A fresh perspective. In *Proceedings of the Graphics Interface 2002 Conference, May 27-29, 2002, Calgary, Alberta, Canada*, pages 17–24, May 2002. URL: http://graphicsinterface.org/wp-content/uploads/gi2002-3.pdf.

[33] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[34] Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Disney Editions, New York, N.Y., 1st hyperi edition, 1981.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] Leo Brodie Walt Stanchfield. *Gesture Drawing for Animation*. Independently published, 1 edition, 2020.

[37] Changshen Zhao. A survey on image style transfer approaches using deep learning. *Journal of Physics: Conference Series*, 1453(1):012129, jan 2020. doi:10.1088/1742-6596/1453/1/012129.

[38] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.