МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА МЕТОДА ПЕРЕНОСА СТИЛЯ ПИКСЕЛЬ-АРТ НА ИЗОБРАЖЕНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Озерова Данилы Николаевича

Научный руководитель
зав. каф., доцент, к. ф.-м. н. _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2023

ВВЕДЕНИЕ

Компьютерная графика претерпела сильные изменения за прошедшие десятилетия. Ранее разработчикам приходилось ограничиваться использованием восьмибитных изображений, в наше время возможно создание фотореалистичной графики, неотличимой от того, что люди могут видеть собственными глазами. Однако пиксельная графика (изображения в стиле пиксель-арт) запала в душу геймерам и игры в такой стилистике до сих пор востребованы. Этому служат много причин, в том числе эстетические и финансовые, однако для данной работы важен лишь факт актуальности данного ответвления компьютерной графики.

Классический способ создания пиксельной графики — тщательная попиксельная ручная прорисовка. Эта нетривиальная работа требует опыта и затраты большого количества времени, поскольку, в отличии от обычных изображений, художник сильно ограничен размерностью полотна (чаще всего это 8-ми, 16-ти и 32-х битный формат). Особенно сложной задачей становится попытка передать таким образом большое количество деталей.

Очевидным методом превратить изображение в пиксельное является уменьшение разрешения изображения. Такие подходы встречаются в ранних работах посвященных данной проблеме. Однако в силу того, что механизм алгоритма основан на работе с ядром, выходные изображения имеют размытые, нечеткие края. Эти ограничения подталкивают на исследование альтернатив данному подходу.

Актуальность данной работы состоит в проектировании новой модели перевода стиля из изображения в изображение для задачи создания изображения в стиле пиксель-арт, в которой будут исключены некоторые недостатки современных подходов, таких как наличие выборки парных изображений, отсутствие четкости у границ частей изображений, потеря цветовой гаммы при переводе.

Практическая ценность бакалаврской работы заключается в необходимости совершенствования методов переноса стиля из изображения в изображение, уменьшение времени на подготовку перед использованием. Сведение ручного труда к минимуму значительно снижает как временные, так и денежные затраты.

Предметом исследования является алгоритм переноса стиля изображений на изображения.

Цель исследования заключается в разработке новой модели перевода стиля из изображения в изображение в рамках задачи создания пиксель-арта.

Для достижения цели поставлены следующие **задачи**:

- Проанализировать предшествующие подходы к решению проблемы;
- Обоснованно выбрать метод реализации модели;
- Разработать архитектуру модели;
- Реализовать выбранную модель;
- Обучить выбранную модель;
- Протестировать модель на выбранных изображениях;
- Разработать приложение для удобства работы с обученной сетью.

Структура выпускной работы состоит из введения, двух разделов, заключения, списка используемой литературы и четырех приложений. Общий объем работы — 71 страница, из них 45 страниц — основное содержание, включая 16 рисунков, список использованных источников информации — 24 наименований.

1 Основное содержание работы

Задача переноса стиля и методы ее решения. Проблема переноса стиля изучается более двадцати лет как одна из важных областей компьютерного зрения. Ее задача состоит в синтезировании новых изображений, соединяя содержание и стиль разных изображений.

Первыми работами в этом направлении считаются разработки алгоритмов передачи стиля аналогии изображений (image analogy) и лоскутных изображений (image quilting). Такие алгоритмы обучаются на парных изображениях и основаны на алгоритмах генерации текстур на основе патчей.

Затем им на смену пришли алгоритмы нейросетевого стиля. Два наиболее актуальных метода решения задачи переноса стиля на сегодняшний день — это алгоритм нейросетевого переноса стиля (Neural Style Transfer или NST) и алгоритм циклической генеративно-состязательной сети (CycleGAN).

Алгоритм NST основан на идее отделения частей изображения, отвечающих за содержание и за стиль. Поэтому было предложено использовать архитектуру, где отсутствует слой, отвечающий за предсказание, а также которая хорошо подошла бы для работы с изображениями. Выбор пал на сверточные нейронные сети (convolutional neural networks, CNN), поскольку фильтры в CNN уже используются для выделения определенных признаков изображения. Таким образом происходит отделение частей стиля и содержание изображения, а затем происходит слияние двух изображений, у которого заменяется только часть стиля.

Алгоритм CycleGAN работает на основе идеи генеративно-состязательных сетей. Эта идея основана на генерации изображений из входящего источника. Обычно генеративно-состязательные сети используют две части: для генерации изображений из случайного шума и для определения было ли оно сгенерировано или нет. Сеть CycleGAN отлична от стандартной архитектуры GAN тем, что в нем используется две части для генерации и две для определения и на вход вместо вектора случайного шума поступает изображение.

Главное различие нейросетевых алгоритмов NST и CycleGAN заключается в использовании пар изображений. В то время как алгоритму NST требуются заранее сопоставленные пары изображений с различием в стилях и одинаковой содержательной части, CycleGAN, благодаря своей архитектуре, не требует таких пар.

В данной работе используется метод глубокого обучения без учителя. Этот метод был выбран, поскольку обучение с учителем требует большого числа пар изображений с одинаковым освещением, углом наклона и так далее.

Принцип работы CycleGAN. Пусть имеются две различные неупорядоченные коллекции изображений X и Y.

Данный метод ставит перед собой две основные цели:

— Научится преобразовывать изображения из множества X в множество Y и обратно;

— Сохранять целостность изображений: измененное изображение должно лишь стилистически отличаться от оригинала.

Реализация CycleGAN. Для работы метода используются две важные части: дискриминатор и генератор. Работа генератора заключается в обучение функций $G : X \to G(X) \approx Y$ и $F : Y \to F(Y) \approx X$. Другими словами, G учится генерировать изображение из коллекции X, а F из Y. Соответствующие этим функциями дискриминаторы пытаются корректно различать сгенерированные изображения от случайно выбранных из множеств X и Y. Для функции G и ее дискриминатора D_Y используется следующая целевая соревновательная функция:

$$L_{GAN}(G, D_Y, X, Y) = M_{y \approx p_{data}(y)}(\log(D_Y(y))) + M_{x \approx p_{data}(x)}(\log(1 - D_Y(G(x)))),$$

где $G(y)$ обозначает выход генератора, когда он получает в качестве входа изображение y из множества Y; $D_Y(x)$ вероятность дискриминатора того, что исходные данные x реальны; $D_Y(G(x))$ вероятность дискриминатора того, что сгенерированное изображение $G(x)$ реально; $M_{y \approx p_{data}(y)}$ и $M_{x \approx p_{data}(x)}$ обозначают среднюю вероятность для всех исходных данных и синтетических данных соответственно.

Состязательность алгоритма заключается в подходах разных частей сети к заданной целевой функции: в то время как генератор пытается ее минимизировать (пытаясь представить дискриминатору сгенерированные изображения за настоящие), дискриминатор пытаться максимизировать (обучаясь различать настоящие от сгенерированных).

Таким образом, было найдено решение для первой цели метода CycleGAN, однако, для сохранения целостности (авторы статьи используют термин циклоустойчивость (cycle consistency), этого недостаточно. Для выполнения этого

условия отображение изображения в целевое множество, а затем обратно должно быть как можно более близкое к изначальному:

$$X \to G(X) \to F(G(X)) \approx X$$

и

$$Y \to F(Y) \to G(F(Y)) \approx Y.$$

С целью поддержания циклоустойчивости вводится дополнительная функция потерь. Она приводит условие в действие. Для этого необходимо минимизировать $L1$ расстояние между $F(G(X))$ и x, а также между $G(F(y))$ и y.

$$L_{\text{цикл}}(G, F) = M_{x \sim p_{data}(x)}|F(G(x)) - x| + M_{y \sim p_{data}(y)}|G(F(y)) - y|.$$

Двунаправленность сети необходимо для того, чтобы гарантировать что преобразованное изображение $G(x)$ связано с содержанием исходного изображения (поскольку генератор может выдавать любое изображение из множества Y).

Постановка задачи. В рамках данной работы используются коллекции изображений X (картинки в мультипликационном стиле) и Y (картинки в стиле пиксель-арт). Цель работы — научиться отображать множество X в множество Y и обратно, используя обучающие выборки $\{x_i\}_{i=1}^N$ и $\{y_i\}_{i=1}^N$, где $y_i \in Y$ и $x_i \in X$. Обозначим распредение данных как $x \sim p_{data}(x)$ и $y \sim p_{data}(y)$. Сеть будет обучатся в двух направлениях (bi-directional net). Двунаправленность является необязательной частью сети, однако это предоставляет новые данные для обучения сетей, поэтому в данной работе был использован такой подход.

Метод решения задачи. Разработанная архитектура предполагает двунаправленное обучение, направления которого обозначим как прямое и обратное.

Прямое направление: На вход поступает изображение c_i в мультипликационном стиле, происходит генерация его отображения p_{c_i} в стиле пиксель-арт, которое может быть восстановлено из p_{c_i} в c_i', такое что $c_i \approx c_i'$. Обучение производится в следующем порядке PreNet $\to$ ArtNet $\to$ DeArtNet.

Обратное направление: На вход подается изображение в стиле пиксель-арт p_j, трансформируется в мультяшное изображение c_{p_j}, которое может быть восстановлено в p_j', такое что $p_j \approx p_j'$. Порядок обучения следующий: DeArtNet

$\rightarrow$ PreNet $\rightarrow$ ArtNet.

1.1 Обучающий набор

Обучающий набор $T = \{P, C\}$ состоит из двух множеств $P = \{p_i\}_{i=1}^{N}$ пиксельных изображений и $C = \{c_j\}_{j=1}^{N}$ изображений в рисованном стиле. Данный набор был собран из открытых банков изображений в сети Интернет. Изображения были масштабированы к размеру 256×256, поскольку с большим размером происходит ошибка памяти видеоустройства. Собранные изображения не связаны между собой, не являются парными. На каждой итерации цикла обучения производится выборка случайного изображения из множеств P и S.

Архитектура нейронной сети. Архитектура нейронной сети состоит из трех частей: PreNet (PN), ArtNet (AN), DeArtNet (DN).

Конвейер из подсетей PN и AN можно рассматривать как сеть для генерации пиксель-арта. Здесь PN выполняет функции извлечения признаков, а также создает разноформатные (multiscale grid) пиксельные изображения. Задачей AN является выделить края и увеличить их четкость у предварительно сгенерированных изображений, не изменяя важных частей композиции. Подсеть DN выполняет роль трансформатора пиксельных изображений обратно в изображения в рисованном стиле.

Целевая функция. Поскольку перед каждой подсетью ставятся разные цели, то и функции их потерь будет различны.

Задача подсети PN заключается в первичном преобразовании изображения в рисованном стиле в пиксельный стиль. Её целевая функция описана ниже:

$$L_{PN} = L_{GAN}(PN, D_{PN}, F) + L_{L1\&\text{град}}(PN, F) + L_{L1\&\text{град}}(PN, B),$$

где F и B — направления тренировки (прямое и обратное соответственно). Поскольку в прямом направлении доступно только предварительно обработанное изображение, то, здесь лучше всего использовать состязательную (adversarial) loss-функцию для отображения из C в P. D_{PN} это дискриминатор в сети PN. Потеря $L1$ используется для сохранения исходной цветовой гаммы, а потеря градиента вводится для плавности и четкости границ в изображении. В обратном направлении потери $L1$ и градиента сравниваются с входным пиксельным изображением.

Как упоминалось ранее, данные из сети PN после увеличения разрешения поступают в сеть AN для получения итоговой версии пиксельной графики. Для данной сети используем следующую целевую функцию:

$$L_{AN} = L_{GAN}(AN, D_{AN}, F) + L_{L1\&\text{град}}(AN, F) + L_{\text{зерк}}(DN \to AN, B),$$

где D_{AN} — дискриминатор подсети ArtNet, а потери $L1$ и градиента также используются для сохранения цвета и четкости границ. Зеркальная потеря используется для выполнения критерия циклоустойчивости.

Подсеть DN принимает пиксельное изображение на вход и выводит изображение в рисованом стиле. Аналогично предыдущим примерам ее функция потерь содержит соревновательную, $L1$ и градиентную части в обратном направлении. В прямом направлении используется функция зеркальной потери для регуляризации. Функция подсети DeartNet определена так:

$$L_{DN} = L_{GAN}(DN, D_{DN}, B) + L_{L1\&\text{град}}(DN, B) + L_{\text{зерк}}(PN \to DN, F).$$

Зеркальная потеря. Поскольку наша модель проводит обучение без учителя, существует необходимость в циклоустойчивости процесса. С этой целью в модель вводится зеркальная потеря, которая будет стремится к выполнения следующего условия: в результате цепочки преобразований $c \to p_c \to c'$, должно выполнятся $c \approx c'$, а также карты признаков данных изображений должны быть схожи. Определим зеркальную потерю для прямого направления:

$$L_{\text{зерк}}(PN \to DN, F) = \sum^{l} \omega_l E^l_{\text{зерк}}(F) + \omega_{out} E^{\text{выход}}_{\text{зерк}}(F),$$

где F означает прямое направление, $E_{\text{зерк}}$ — зеркальную ошибку между картами признаков, $E^{\text{выход}}_{\text{зерк}}$ — зеркальную ошибку между входным и выходным изображениями.

$$E^l_{\text{зерк}} = \sum |f^l_{PN} - f^{N-1}_{DN}|,$$

где N - общее число частей в каждой подсети, символ f означает карту призна-

ков на последнем слое блока. Фактически здесь вычисляется $L1$ потеря между блоком l в подсети PN и блоком $N - 1$ в DN. Цель этой функции — минимизировать разницу между картами признаков, чтобы они были «зеркально» похожи.

Потери между входом и выходом определим как:

$$E_{\text{зерк}}^{\text{выход}} = \sum |c_i - c_i'|,$$

где c_i входное изображение в мульт стиле, c_i' выходное изображение в этом же стиле, которое было сгенерировано подсетью DN.

Все вышеописанные функции верны и для обратного направления.

Состязательная функция потерь. Для каждой подсети вводится состязательная потеря:

$$L_{GAN}(PN, D_{PN}, F) = \\ M_{p \sim p_{data}(x)}(log(D_{PN}(p))) + M_{c \sim p_{data}(y)}(log(1 - D_{PN}(PN(c)))),$$

где D_{PN} — это дискриминатор этой сети; $PN(c)$ обозначает выход генератора, когда он получает в качестве входа изображение c из множества X; $D_{PN}(c)$ вероятность, определенная дискриминатором, что исходные данные x реальны; $D_{PN}(PN(c))$ вероятность, определенная дискриминатором, что сгенерированное изображение $PN(c)$ реально; $M_{y \approx p_{data}(y)}$ и $M_{x \approx p_{data}(x)}$ обозначают среднюю вероятность для всех исходных данных и синтетических данных соответственно.

Таким образом, генератор PN учится генерировать изображение из коллекции Y. Соответствующий дискриминатор пытается корректно различить сгенерированные изображения от случайно выбранных из множества Y.

Цель дискриминатора — максимизировать эту функцию, в то время как PN пытается минимизировать ее. Таким образом, эти две сети — генератор и дискриминатор — состязаются друг с другом в игре с нулевой суммой. Математически это выражается как задача минимаксной оптимизации функции:

$$\min_{PN} \max_{D_{PN}} L_{GAN}(PN, D_{PN}, F).$$

В остальных сетях аналогично.

Функции потерь L1 и градиента.

Состязательная функция потерь продемонстрировала свою эффективность в различных работах по задачам перевода изображения в изображение. Однако в отличии от них, в данной работе данные непарные (обучение без учителя), поэтому для сохранности цвета и обеспечения четкости граней изображения вводим дополнительные потери $L1$ и градиента для каждой из подсетей.

$$L_{L1\&\text{град}}(PN, F) = \sum^{r}(E^{r}_{L1}(PN, F) + E^{r}_{\text{град}}(PN, F)),$$

где r означает определенный масштаб сглаживания, поскольку разработанный метод генерирует пиксель арт с разными эффектами сглаживания. Таким образом в данной формуле происходит суммирование потерь $L1$ и градиента по всем типам сглаживания.

Определим $L1$ потерю в прямом направлении для различного r как:

$$E^{r}_{L1}(PN, F) = \sum |I^{r}_{\text{вход}} - I^{r}_{\text{выход}}|.$$

Производится масштабирование $I_{\text{вход}} \to I^{r}_{\text{вход}}$, так чтобы размеры $I_{\text{вход}}$ и $I^{r}_{\text{выход}}$ совпадали.

Определим потерю градиента в прямом направлении для различных масштабов r как:

$$\begin{aligned} &E^{r}_{\text{град}}(PN, F) = \\ &= \sum \Big(\Big| |I^{r}_{\text{вход}}(x, y) - I^{r}_{\text{вход}}(x-1, y)| - |I^{r}_{\text{выход}}(x, y) - I^{r}_{\text{выход}}(x-1, y)| \Big| + \\ &\qquad + \Big| |I^{r}_{\text{вход}}(x, y) - I^{r}_{\text{вход}}(x, y-1)| - |I^{r}_{\text{выход}}(x, y) - I^{r}_{\text{выход}}(x, y)| \Big| \Big) \end{aligned}$$

Функции потерь $L1$ и градиента в других подсетях аналогичны.

Технологии разработки. В качестве языка написания програмного кода выбран язык Python, поскольку на нем располагается официальная реализация алгоритма CycleGAN, а также он отлично подходит для работы с нейронными сетями. Рабочим фремворком выбран PyTorch, поскольку он является гибким, подробно документированным, удобным в отладке и также наиболее быстро среди аналогов обеспечивает параллельную работу на видеоустройствах.

Оборудование для обучения. Обучение производилось на устройстве со

следующими характеристиками:

- Процессор: одноядерный процессор Xeon с гиперпоточностью @2,3 ГГц, то есть (1 ядро, 2 потока);
- Видеокарта: NVIDIA Tesla K80 2496 ядер CUDA, 12 ГБ GDDR5 VRAM;
- Оперативная память: 12 ГБ;
- Диск: 50 ГБ.

Время непрерывного обучения 150 эпох составило примерно неделю.

Приложение для работы с моделью. Для удобства работы с сетью было принято решение создать настольное приложение, которое будет обладать базовым необходимым функционалом: загрузкой фотографии с диска и ее дальнейшей стилизацией в стиль пиксель-арт.

ЗАКЛЮЧЕНИЕ

В данной работе была спроектирована и разработана нейронная сеть для задачи создания изображения в стиле пиксель-арт из рисованных изображений. Важной частью используемого в работе метода является отсутствие необходимости в парных тренировочных данных, поскольку такой объем идентичных пар (с различием лишь в стиле) собрать из открытых источников невозможно, поскольку, даже если художник берет рисунок за основу, это не означает, что он будет соблюдать все исходные детали.

В ходе проектирования было разработано три подсети для различных внутренних целей. Была введена потеря градиента для сохранения изображением своего цветового пространства. Также использована зеркальная потеря в целях поддержания циклоустойчивости модели.

Во время выполнения данной работы были успешно осуществлены следующие задачи:

- Исследована задача переноса стиля в области компьютерного зрения;
- Изучены сопутствующие работы по данной тематике;
- Проанализированы существующие алгоритмы области, на этой основе был выбран подход для дальнейшего исследования;
- Спроектирована циклическиустойчивая генеративно-состязательная сеть со значительными модификациями, основанными на современных исследованиях;
- Спроектированная сеть была разработана с использованием фреймворка PyTorch языка Python;
- Нейронная сеть была обучена с использованием собранного датасета;
- Было разработано настольное приложение для использования обученной нейронной сети.

Для тестирования использовались изображения различающиеся цветовой гаммой, разрешением, фоном. Кроме того, было проведено сравнение разработанной модели с двумя алгоритмами. По окончании тестирования была построена таблица и отображены некоторые из тестов, демонстрирующие преимущества и недостатки разработанной модели.

Результаты работы демонстрируют, что метод способен переносить стиль пиксель-арт на рисунки и создает их реалистичнее, чем алгоритмы по понижению дискретизации изображений и нейросетевой алгоритм CycleGAN.