Оглавление

Для разработчиков

План работы

Для реализации проекта «Основанный на нейронной сети бот на платформе “Telegram” для переноса стилей разных художников на любые изображения» необходимо:

1. Определиться с выбором языков программирования
2. Создать Бота на платформе «Telegram»
3. Реализовать нейронную сеть
4. Выбрать каким образом соеденить бота и нейронную сеть
5. Соеденить нейронную сеть и бота
6. Сделать тесты с последующими выводами и обратной связью с разработчиками
7. Исходя из тестов сделать исправления в работе продукта

Контактная информация всех разработчиков:

Разработчик Бота – Шурин Артём:

GitHub: https://github.com/ArtemShurin

Telegram: https://t.me/Punished_VS

T. number: + 7 985 387 8133

Разработчик Нейронной сети – Конанов Сергей:

GitHub: https://github.com/Sp1d3rGH

Telegram: https://t.me/sixshooter_showdown

T. number: + 7 995 458 9933

Тестировщик продукта – Серегин Егор:

GitHub: https://github.com/Marth444

Telegram: https://t.me/Marth_444

T. number: + 7 900 448 778

Описание работы (теория)

Принцип работы бота:

Доступ к боту осуществляется по уникальному токену, сгенерированному при его создании.

Бот имеет три тригггера: команда (/start), триггер на текстовые сообщения, триггер на изображения. В триггере команды (/start) и триггере на текст изложены все инструкции для корректного взаимодейсвия пользователься с ботом.

Триггер фотографий обрабатывет два раздельно-присланные пользователем фотографии, сохраняет их, создаёт тройки файлов формата (jpg) для каждого пользователя (обрабатываемое фото, стилизующее фото и выходной результат). Сохранение информации о фотографиях проиходит в словаре. Ключём является id чата с пользователем, значением массив фотографий, с подобранными уникальными именами. Разделение фотографий из сообщений происходит по принципу, наличия уникального ключа: при обработке первого сообщения ключ создаётся, а в массив изображений загружается первое изображение, при обработке второго сообщения проверяется наличие ключа, и в случае положительного результата в массив значений записывается информация о второй и выходной фотографиях. Это необходимо для того, чтобы не происходило путаници при работе с ботом нескольких пользователей.

Далее массив значений каждого пользователя подаётся на вход нейронной сети с указанием колличества итераций. В процессе работы нейронной сети создаётся и сохраняется итоговый файл, который в последствии передаётся ботом пользователю. После программа очищает информацию о текущем пользователе, удаляя загруженнеы и созданные, его ключ и значение по ключу из словаря.

Принцип работы нейронной сети:

Для работы с изображениями будет использоваться известная свёрточная нейронная сеть VGG19, которая предварительно настроена на задачу классификации изображений. Принцип передачи стиля заключается в определении двух функций, одна из которых показывает отличие двух изображений, а другая описывает разницу между двумя стилями изображений. Сеть пытается преобразовать входное изображение так, чтобы минимизировать расстояние от функции различия изображения и функции различия стиля.

Функция начала работы нейронной сети получает на вход два изображения, которые мы представляем в виде массивов NumPy и подготавливаем к обработке сетью VGG19 с помощью функции библиотеки Keras.

После этого необходимо настроить модель VGG19, в которую мы загружаем промежуточное слои - это карты признаков, которые по мере углубления становятся более упорядоченными, они позволят сети лучше понимать изображения и выявлять особенности. Функция построения модели возвращает данные с промежуточных слоёв VGG19.

Теперь нужно определить функции потерь, о которых говорилось выше. Для различия изображений, в кратце, она описывает расстояние содержимого Lcontent между входным изображением X и изображением контента P по формуле: $$L^l_{content}(p, x) = \sum_{i,j} (F^l_{ij}(x) - P^l_{ij}(p))^2$$

Здесь $F^l_{ij}(x)$ и $P^l_{ij}(x)$ описывают соответствующие промежуточные представления о X и P.

Для различия стилей этот процесс заключается в описании функции потерь для стиля главного изображения X, изображения стиля A и расстояния между представлениями матрицей Грама стиля этих двух картинок.

Суммарное влияние каждого слоя на функцию потерь: $$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$

Здесь G^l_{ij} — это внутреннее произведение между векторизированной картой признаков i и j в слое L.

G^l_{ij} и A^l_{ij} — это соответствующие представления на слое L входного изображения X и изображения стиля A.

N_l описывает количество карт объектов, каждая из которых имеет размер M_l = (высота * ширина).

Функция потерь всех слоёв:

$$L_{style}(a, x) = \sum_{l \in L} w_l E_l$$

Тут взвешивается влияние потери каждого слоя от какого-либо фактора wl.

$$(w_l = \frac{1}{\|L\|})$$

После работы сети получается массив NumPy выходного изображения, который мы деобрабатываем от VGG19 и выводим.

Установка и пользование

Как пользоваться ботом

1) Добавляем бота по ссылке (t.me/PhotoConverterAK_bot)
2) Вводим первую команду /start
3) Следуем интрукция, отправляемым ботом вам в чат

Советы для пользования:

1) Используйте картинки хорошего разрешения
2) Бот относительно плохо работает с человеческими лицами, но хорошо переносит стиль картинки без человеческих фигур
3) Старайтесь для адекватного результата не использовать неподходящие пары фотографий по концепции

Способ установки на свой компьютер

1) Убедиться, что на компьютере установлен Python последней версии, pip, а также имеется интерпретатор
2) Необходимо также установить необходимые пакеты через windows cmd из списка (pip install -U matplotlib, pip install tensorflow, pip install pytelegrambotapi, pip install ipython)
3) Скачиваем файлы из репозитория по ссылке (https://github.com/Sp1d3rGH/Telegram-Neural-Style-Transfer)
4) Запускаем файл с исходным кодом бота (Bot_source.py), убедившись, что в той же директории присутствует файл с нейронной сетью (neural_network_source.py)

• Важно: бот работает пока вы вручную не завершите работу файла с кодом.

Тесты

Итерация равна 100

• Исходное фото: разрешение 1280 X 960

Изображение стиля: разрешение 1200 X 796

Результат, полученный в боте:

Видно, что бот обрабатывает изображение и выдаёт результат

• Исходное фото: разрешение 1024 X 683

Изображение стиля: разрешение 736 X 549

Результат, полученный в боте:

• Исходное фото: разрешение 1280 X 853

Изображение стиля: разрешение 1200 X 867

Результат, полученный в боте:

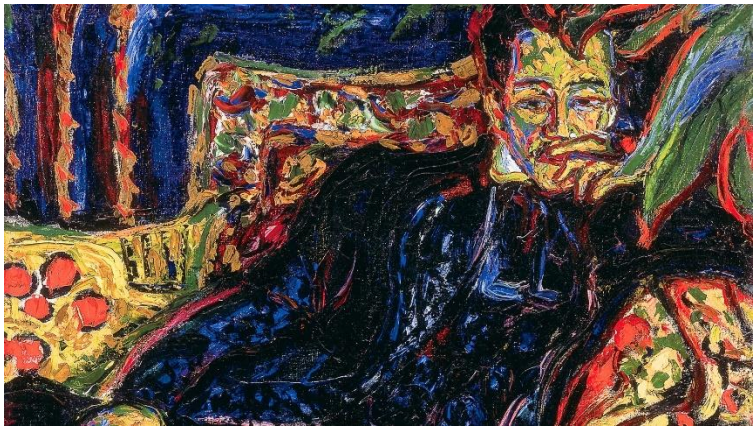
• Исходное фото: разрешение 1200 X 800

Изображение стиля: разрешение 6030 X 3980

Результат, полученный в боте:

• Исходное фото: разрешение 2953 X 2402

Изображение стиля: разрешение 1600 X 901



Результат, полученный в боте:

Вывод

Исходя из тестов, проведённых на итерации 100, можно сделать следующие выводы:

• Изображения, содержащие воду, лес и другие природные объекты – обрабатываются хорошо.

• Нейронная сеть не распознаёт силуэт человека на фоне и обрабатывает людей исходя из общего определения стиля, хотя они могу отличаться манерой написания конкретно в этом стиле.

• Нейронная сеть хорошо определяет яркие художественные стили и на их основе обрабатывает изображения.

• Скорее всего нужно указать в разделе для пользователей, что не стоит рассчитывать на удовлетворяющий результат, когда исходная фотография содержит много людей или фото, выбранное стилем, является картиной поколения художников 19 века (крайне экспрессивный стиль картины).

• Необходимо указать пользователю, что исходная фотография может не подходить для обработки каким-то стилем, хотя обработка и будь осуществлена, но выходное изображение может потерять очертания исходного изображения (яркий пример с кубизмом и изображение современного города – обработка “съедает” очертания домов и скорее всего на больших итерациях они совсем пропадут).

• На данной итерации обработка занимает 5 минут.

Итерация равна 500

• Исходное фото: разрешение 1280 X 960

Изображение стиля: разрешение 1200 X 796

Результат, полученный в боте.

Предыдущий результат.

Видно, что количество итераций влияет на результат в лучшую сторону

• Исходное фото: разрешение 1024 X 683

Изображение стиля: разрешение 736 X 549

Результат, полученный в боте.

Предыдущий результат.

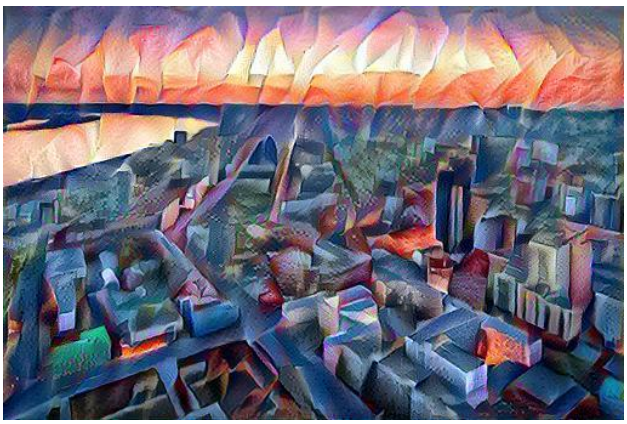
Особой разницы не видно между 100 и 500 итерациями

• Исходное фото: разрешение 1280 X 853

Изображение стиля: разрешение 1200 X 867

Результат, полученный в боте.

Предыдущий результат.



Особой разницы не видно между 100 и 500 итерациями

• Исходное фото: разрешение 1200 X 800

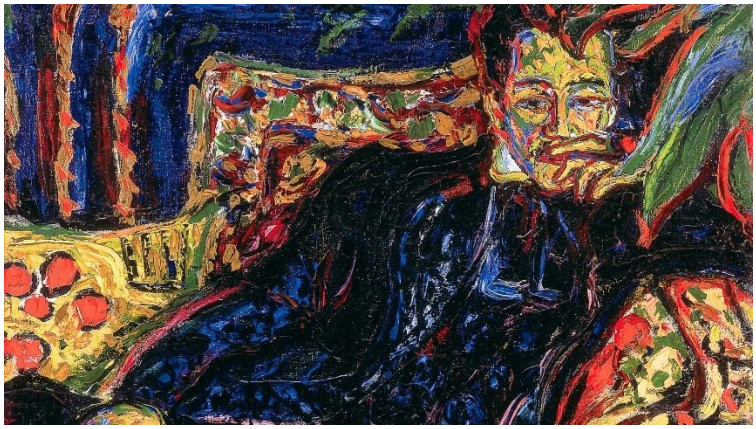
Изображение стиля: разрешение 6030 X 3980

Результат, полученный в боте.

Предыдущий результат.

• Исходное фото: разрешение 2953 X 2402

Изображение стиля: разрешение 1600 X 901



Результат, полученный в боте.

Предыдущий результат.

Вывод

Исходя из тестов, проведённых на итерации 500, можно сделать следующие выводы:

• 100 итераций и 500 отличаются не так сильно, стоить сделать тесты на 1000 итераций и сравнить с 100.

• Некоторые изображения практические не изменились (возможно дело в способе реализации нейронной сети)

Итерация равна 1000

• Исходное фото: разрешение 1280 X 960

Изображение стиля: разрешение 1200 X 796

Результат 1000 итераций.

Предыдущий результат 100 итераций

Качество обработки увеличилось

• Исходное фото: разрешение 1024 X 683

Изображение стиля: разрешение 736 X 549

Результат 1000 итераций.

Предыдущий результат 100 итераций

Опять изменений почти нет (предположительно специфика стиля)

• Исходное фото: разрешение 1280 X 853

Изображение стиля: разрешение 1200 X 867

Результат 1000 итераций.

Предыдущий результат 100 итераций

Очертание города постепенно пропадает (предположительно специфика Кубизма)

• Исходное фото: разрешение 1200 X 800

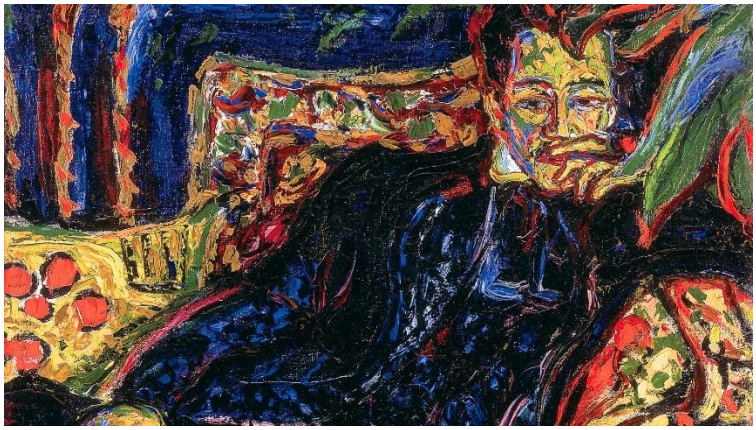
Изображение стиля: разрешение 6030 X 3980

Результат 1000 итераций.

Предыдущий результат 100 итераций

• Исходное фото: разрешение 2953 X 2402

Изображение стиля: разрешение 1600 X 901



Результат 1000 итераций.

Предыдущий результат 100 итераций

Вывод

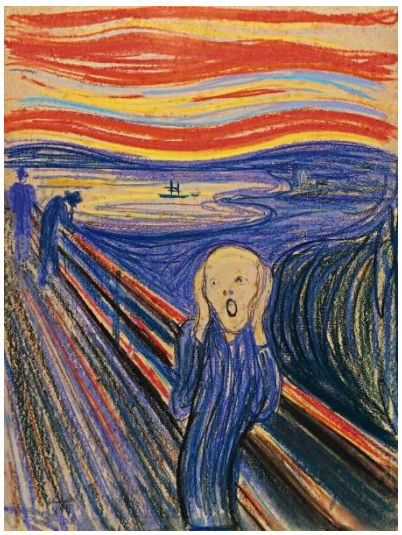
Предположения, выдвинутые в выводе с 100 итерациями, подтвердились. 1000 итераций лучше обрабатывает фотографии, что очевидно, но не всегда это так. Всё зависит от стиля. Так же итоговые фотографии почему всегда имеют формат 512 на 416. Разработчикам обратить внимание!

Тесты одной картинки на разные стили

Исходная картинка:

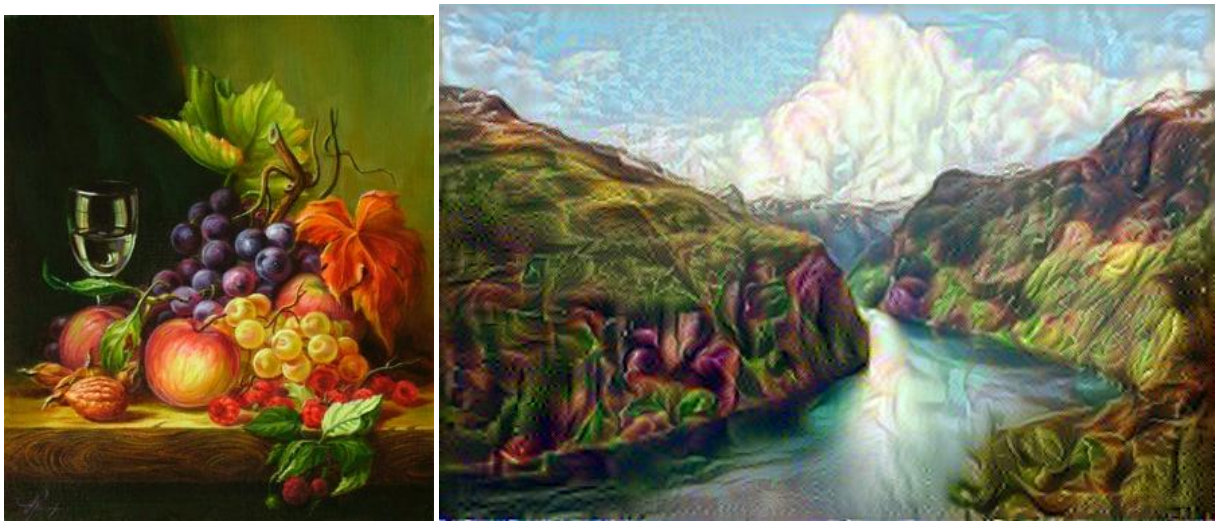
Стиль и результат соответственно:

1)



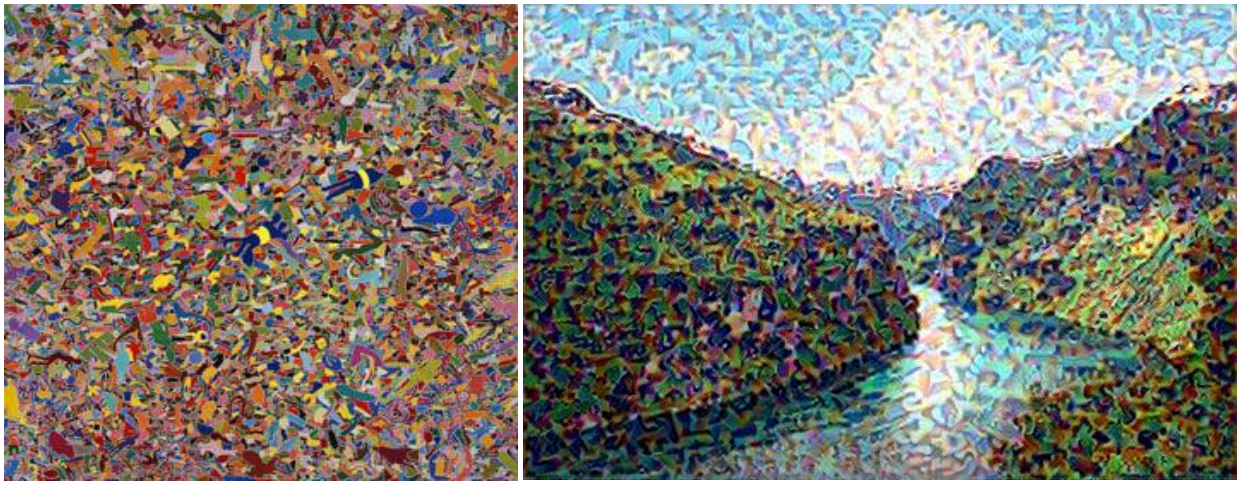
2)

3)



4)

5)



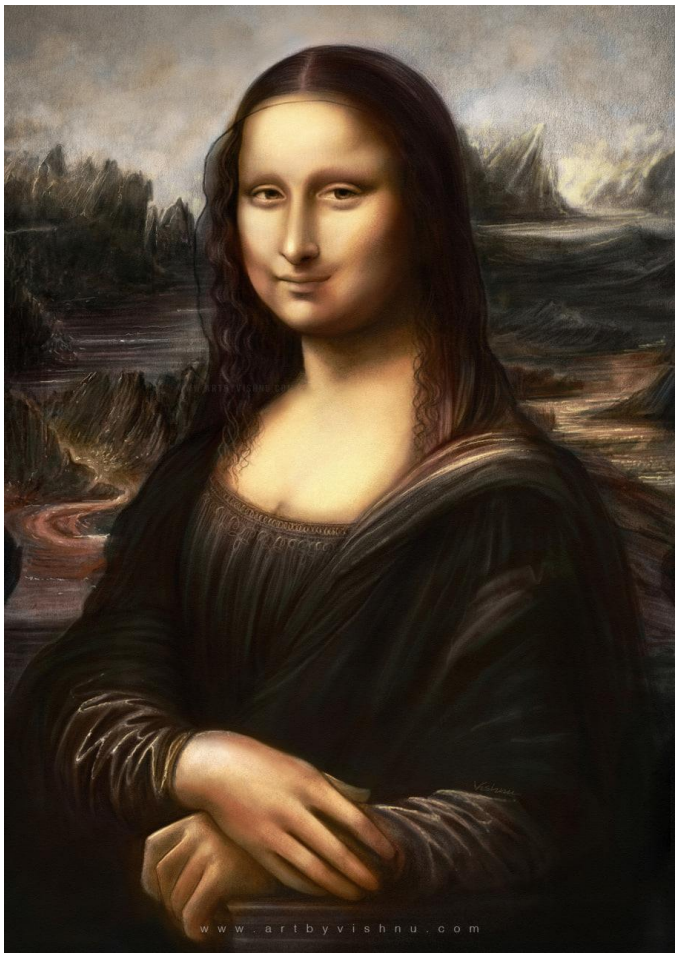
6)

Вывод

Исходя из тестов произведённых на одной картинке следует, что продукт действительно работает правильно. Нейронная сеть распознаёт особенности разных стилей, но не всегда исходная картинка вообще может одновременно конвертируема в изображение любого стиля и выглядеть адекватно. Снова подтвердилась информация, что ландшафтные картины бот обрабатывает лучше, чем людей и их лица, но нужен отдельный тест.

Тест исключительно на портрет и работу с человеческим лицом

Разберём пару фотографий:



Получим:

Видна работа программы, но результат неудовлетворителен для пользователя. Нейронная сеть плохо распознаёт лица и работает с ними.

Отдельный тест на работу с разрешение получаемого изображения и зависимость от разрешения исходного изображения и стиля

Для примера будем работать с исходным изображение планет и как стиль дадим программе одну и туже картинку, но вторую очень плохого качества.

Исходная:

Стили:

1) 1280 X 1038

2) 148 X 150

На выходе имеем:

1)

2)

Вывод

Полученные изображения одинакового разрешения, но второе сильно хуже качеством. Соответственно формат итогового изображения никак не зависит от исходных (всё делается на уровне кода) и чем хуже стиль изображения, тем хуже получится итоговое изображение.

Тест после обратной связи с разработчиком на момент разрешения выходного файла

Входная картинка: 1280 X 720

Стиль: 1280 X 1038

Результат: 1280 X 720

Вывод: видна работа бота, разрешение обработанного изображения соответствует разрешения исходного файла.