|  |
| --- |
| **Отчет об исследовании** |
| **Перенос стиля фотографии с использованием глубоких нейронных сетей.** |
|  |
| **Пронченко Павел Леонидович** |
| **19.06.2019** |

**Содержание.**

Постановка задачи…………………………………………………………..3

Анализ предшествующих решений………………………………………..4

Методика решения задачи………………………………………………….6

Код модели, результаты и выводы……..………………………………….14

Список источников………………………………………………………....15

**1. Постановка задачи**

Актуальность.

В современных условиях большая роль в развитии общества принадлежит машинному обучению. При этом одним из важнейших направлений развития искусственного интеллекта является использование глубоких сверточных нейронных сетей. Перенос стиля между фотографией и художественным изображением является распространенной и хорошо изученной областью компьютерного зрения. Однако традиционные модели не очень хорошо подходят для переноса стиля между двумя фотографиями, поскольку фотографии, как правило, имеют очень локализованный стиль. При этом перенос стиля между двумя фотографиями потенциально может быть полезен для наложения фильтров в приложениях для обработки фото или для техник улучшения фотографий. В работе Luan A. и др. [10] были показаны методы оптимизации традиционных моделей переноса стиля в применении к фото.

Цель работы.

Исследовать алгоритмы глубокого переноса стиля фотографии и применить их в практической деятельности.

Задачи.

1. Изучить существующие подходы к переносу стиля изображения

2. Исследовать особенности глубокого переноса стиля фотографии

3. Разработать рабочий прототип кода модели на PyTorch

4. Определить возможные пути оптимизации кода

5. Исследовать возможные пути развития алгоритма переноса стиля

**2. Анализ предшествующих решений**

Большинству пользователей смартфонов известны приложения переноса стилей изображений. Подобные решения предлагает Facebook или приложение Prisma, которое показало взрывной рост установок на смартфоны сразу после выхода в магазинах приложений. Пользователям нравится возможность стилизации своих фото под стиль художников эпохи возрождения или импрессионистов. Основная работа по переносу стиля изображения была выложена в виде препринта на arxiv.org коллективом авторов Leon A. Gatys, Alexander S. Ecker, Matthias Bethge в сентябре 2015 года. Ими впервые было предложено использовать глубокие нейронные сети в задачах переноса стиля изображения.

До этой работы перенос стиля возможен был только кропотливым трудом художника-иллюстратора. В этой работе была впервые выдвинута идея оптимизации составной функции потерь, состоящей из функции потерь стиля и функции потерь контента. Однако данный алгоритм был не применим в случае необходимости переноса стиля с одной фотографии на другую.

В алгоритме Гэтиса даже если исходное изображение и образец являются фотографиями, конечное изображение всё равно напоминает рисунок, где чёткие линии и границы становятся размытыми, а текстуры накладываются на соседние объекты. Выглядит красиво, но не очень фотореалистично. Результат работы Deep Photo Style Transfer строго фотореалистичен. «Эффекты живописи» устранили за счёт запрета на пространственные искажения. Здесь перенос стилей ограничен только цветовым пространстве. Другими словами, формы объектов остаются точно такими же, как в оригинале.

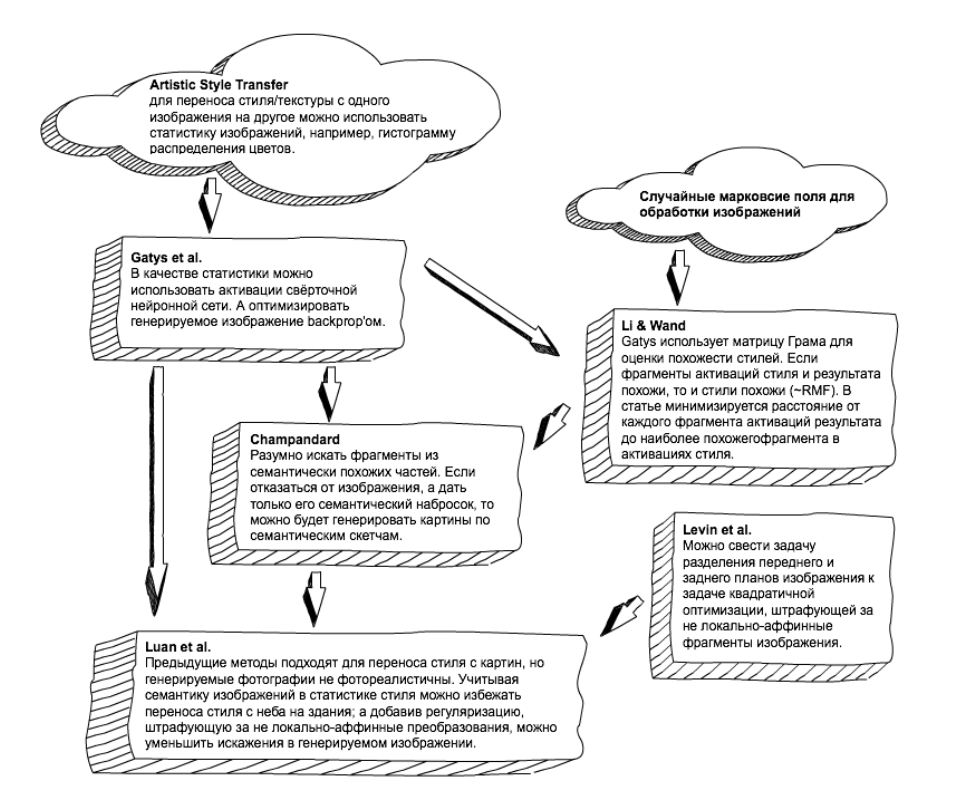


Рис. 1. Диаграмма эволюции подходов переноса стиля

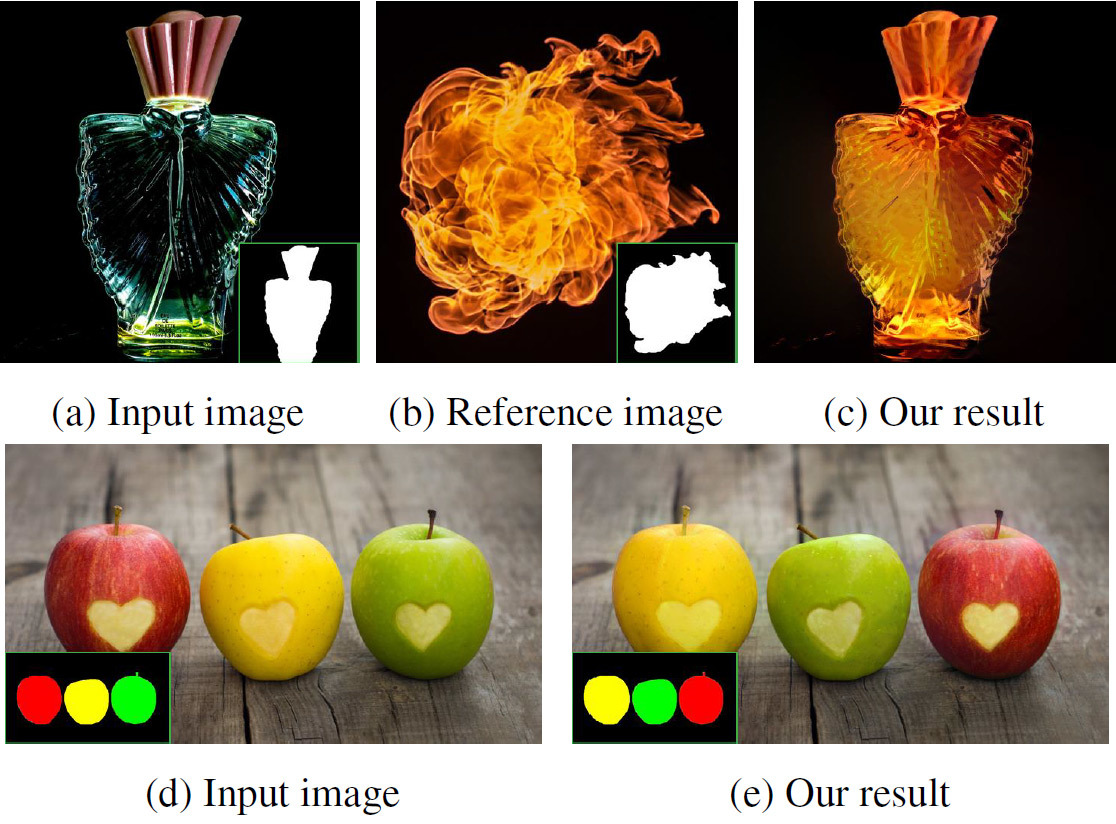
Наиболее наглядно описанный принцип проявляются в двух специально подобранных примерах (Рис. 2)

Рис. 2. Примеры работы алгоритма Deep Photo Style Transfer

Как видим, границы объектов чётко сохраняются. Преобразование происходит только в цветовом пространстве. В первом случае стиль огня переносится на флакон с духами, а во втором случае меняется текстура яблок в соответствии с шаблоном.

Второе достижение авторов программы — решение проблемы с переносом стиля между неподходящими объектами на исходной фотографии и образце стиля. Это сделано за счёт семантической сегментации. Поэтому стили домов переносятся только между домами, и стиль неба влияет только на небо. Во многих случаях семантическая сегментация очень эффективно работает, если только на образце стиля есть те же семантические объекты, что на исходном изображении.

Два вышеупомянутых подхода обеспечили исключительно реалистичное преобразование фотографий.

**3. Методика решения задачи.**

Перенос стиля между двумя фотографиями – очень актуальная проблема в компьютерном зрении. При наличии как изображения контента, так и изображения стиля перенос в фотографическом стиле воссоздает изображение контента в стиле второго изображения. Например, мы могли бы сфотографировать мост в дневное время и изменить изображение так, чтобы оно выглядело так, как будто оно было снято на закате.

Применение традиционной методологии переноса стиля (Gatys L.) к двум фотографиям имеет серьезные ограничения. Картины, как правило, имеют очень обобщенный стиль, при этом фотографии, как правило, имеют более локализованный стиль, где передний план изображения может стилистически отличаться от фона. Из-за этого методы передачи могут ошибаться относительно того, как или где передавать стиль. Традиционная передача стиля имеет тенденцию искажать края выходного изображения. Это не имеет большого значения при переносе стиля с картины на картину т.к. нарисованное изображение обычно имеет элемент сюрреализма. Однако это искажение при передаче между двумя фотографиями может создать нереалистичное выходное изображение.

В статье Deep Photo Style Transfer Луан и др. [10] предлагают два улучшения традиционных методов передачи стилей для лучшей передачи между двумя фотографиями. Первое улучшение - это семантическая сегментация изображений и стилей. Используя эту сегментацию, стиль может передаваться только между определенными слоями, чтобы избежать различий в стилях и проблем с несоответствием содержимого. Второе улучшение - это функция регуляризации фотореализма, добавленная к функции потерь, которая детализирует вывод для исключения искажения изображения. Мы сосредоточимся на реализации локальной передачи стиля с использованием сегментированных изображений.

Данная работа направлена ​​на воссоздание статьи Deep Photo Style Transfer Луана и др. [10]. Эта статья основана на хорошо известной работе Gatys L. [3].

**3.1. Перенос стиля**

Целью переноса стиля, является передача стиля или текстуры данного изображения другому интересующему изображению. Эта идея была впервые введена в [3], которую мы будем использовать в качестве отправной точки для текущей работы.

Формально у нас есть два изображения: изображение контента и изображение стиля . Цель состоит в том, чтобы сгенерировать новое изображение так, чтобы оно содержало контент из в стиле из . Для этого сначала нужно извлечь контент и стиль из любого заданного изображения, а затем объединить их в выходное изображение.

Необходимо использовать предварительно обученную сверточную нейронную сеть, выходные данные каждого сверточного слоя показывают по-разному отфильтрованные версии входного изображения. Как правило, слои более низкого уровня воспроизводят точные значения пикселей, тогда как изображения более высокого уровня отображают контент более высокого уровня. Стиль не так легко извлечь из изображения, но мы сделаем и это, используя ту же обученную сверточную нейронную сеть.

Следующие параграфы объясняют более формально детали извлечения контента и стиля. Позже мы объясним, как создать изображение, которое соответствует контенту и стилю одновременно.

**3.1.1 Извлечение контента**

Как указано выше, после каждого сверточного слоя у нас есть feature map (карта активации нейронов) входного изображения контента

Пусть:

– количество сверточных фильтров в *l-ом* сверточномслое;

– размер каждой финальной карты активации ()

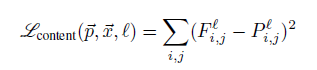
До сих пор мы представляли выход слоя как трехмерное изображение размеров (). По-иному можно это представить, сохраняя выходы слоя *l,* в виде двухмерной матрицы , где – активация *i*-го фильтра в позиции *j* в слое *l.*

Такая матрица называется матрицей представления признаков в слое *l* и будет служить основным строительным блоком для переноса контента.

**3.1.2 Генерация изображения, соответствующего входному контенту**

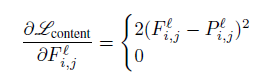
Чтобы создать изображение, воссоздающее контент из входного изображения , мы начнем генерацию изображения с белого шума .

Пусть тогда и будут представлениями признаков (feature map) в слое *l* из и соответственно. Таким образом, мы можем вычислить функцию потерь контента (content loss) между двумя представлениями признаков как:



(1)

Если мы используем ReLU после каждого сверточного слоя, производная от функции потерь контента это:

, при

(2)

При этом мы можем вычислить и обновить, используя

обратное распространение ошибки (backpropagation) до тех пор, пока не будет сгенерирован результат приближенный к исходному в некотором слое *l*. Таким образом, мы описали способ генерации изображения , который копирует содержимое нашего исходного содержимого изображения . Рассмотрим, как повторить желаемый стиль.

**3.1.3 Извлечение стиля**

Чтобы извлечь стиль из изображения, на выходе каждого слоя сверточной нейронной сети, мы вычисляем корреляции между каждой картой активации. Эта информация закодирована в матрице , которая называется матрицей Грама. Формально каждый элемент матрицы может быть рассчитан как:

 (3)

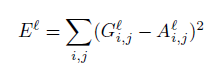
который является матричным произведением векторов каждой карты активации.

Как и при извлечении контента, нижние слои сверточной сети отвечают за локальные структуры стиля, тогда как слои более высокого уровня отвечают за более общие оттенки стиля.

**3.1.4 Генерация изображения, соответствующего стилю исходного изображения**

Так же, как мы сделали для генерации контента, мы начинаем генерацию случайного изображения с белого шума . Пусть тогда и будут матрицы Грама после слоя *l* из и соответственно, где - это исходное изображение, стиль которого мы хотим перенести.

Таким образом, мы можем вычислить вклад слоя *l* в общую функцию потерь как:

 (4)

Рассматривая несколько слоев сверточной сети для оценки стиля, мы можем выразить общую функцию потерь стиля как:

 (5)

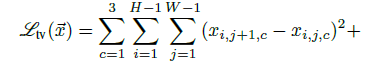
Мы можем аналогично обратно распространить ошибку, чтобы оптимизировать так, чтобы его стиль очень близок к стилю исходного изображения.

**3.2. Генерация производного изображения с контентом и стилем соответствующих исходных изображений**

Теперь, когда у нас есть методология генерации изображения, которое соответствует контенту одного изображения и стилю другого изображения, мы хотим объединить эти два компонента и создать выходное изображение, которое сочетает в себе как контент, так и стиль. Мы достигаем этого путем совместной минимизации расстояния от вектора случайного изображения до вектора изображения стиля и изображения контента .

Наконец, в дополнение к функциям потерь контента и стиля, мы дополнительно включаем в общую функцию потерь функцию потерь вариаций, также называемую потерями TV. Эта функция потерь позволяет убрать шум с итогового изображения, обеспечивая плавность изменчивости между соседними пикселями.

Функцию потерь TV определим так:

(6)

Таким образом, общая функция потерь (total loss), которую мы стремимся минимизировать:

 (7)

Где и – настраиваемые гиперпараметры.

Посредством обратного распространения ошибки, после нескольких итераций получаем выходное изображение , которое объединяет контент из и стиль из .

На рисунке 3 показано, как сгенерировать изображение, сохраняя контент и стиль исходных изображений.

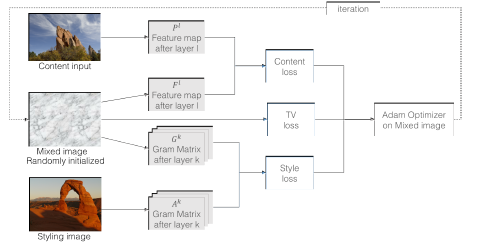


Рис. 3. Генерация изображения с сохранением стиля и контента исходных изображений.

**3.3. Глубокий перенос стиля фотографий**

Глубокий перенос стиля фотографий основан на традиционном переносе стиля изображения, в то же время он пытается сохранить фотореализм изображений и обобщить контент и стиль изображений.

Авторы из [10] предлагают метод, который улучшает алгоритм переноса стиля через две основные идеи.

Во-первых, они включают функцию регуляризации фотореализма в общую фукцию потерь. Во-вторых, они выполняют семантическую сегментацию входов, чтобы избежать проблемы различий стиля и контента. Таким образом, небо из входного изображения, будет стилизовано на основе неба в изображении стиля, без шума от других частей изображения стиля.

В статье [10] решено использовать предварительно сегментированные изображения, чтобы было возможно оптимизировать задачу переноса стиля. Чтобы перенести стиль и контент, мы создаем маски на основе сегментированной версии изображения. Пусть *k* будет количеством сегментов в изображении высоты *H* и ширины *W*. Затем мы генерируем *k* масок высоты *H* и ширины *W,* где *i*-я маска содержит значение *True* в каждом пикселе, соответствующем *i-*му сегменту изображения и *False* во всех других областях. Таким образом, мы создали маску *M* формы (*H x W x k*).

Напомним, что для создания изображения, которое соответствует изображению стиля, нам необходимо рассчитать матрицы Грама из стилевого представления разных слоев *l.* Назовем *Jl* это представление матриц активации слоя.

Далее мы изменяем размер маски M в соответствии с формой стиля представление *Jl*. Наконец, каждый канал *Jl*  умножается на каждый из слоев маски. Таким образом, у нас есть *k* «замаскированных» версий представления функции *Jl*, которые будут использоваться для вычисления *k* сегментных матриц Грама *G* и *A* для стиля и смешанного изображения соответственно. Это делается с использованием изображения стиля и смешанного изображения (для смешанного изображения используется маска из изображения содержимого), а затем на полученных матрицах Грама вычисляется функция потери стиля.

Потеря стиля переопределяется следующим образом:

(8)

Поэтому функцию полной потери в алгоритме Deep Photo Style Transfer  
тогда можно записать так:

(9)

На рисунке 4 показано, как мы использовали сегментированные изображения для локального переноса стиля.

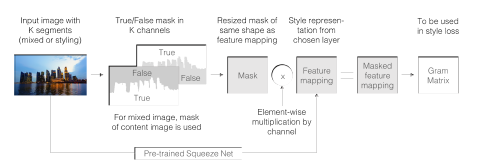
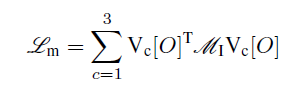


Рисунок 4. Использование сегментированных изображений для локализованного переноса стиля

В [10], авторы добавляют дополнительную функцию потерь к целевой функции, чтобы сохранить края и усилить фотореализм. Функция потерь фотореализма определяется как:

(10)

где - векторизованная версия текущего выходного изображения, а - матрица Лапласа ([Кирхгофа](http://wikiredia.ru/wiki/%D0%9A%D0%B8%D1%80%D1%85%D0%B3%D0%BE%D1%84,_%D0%93%D1%83%D1%81%D1%82%D0%B0%D0%B2_%D0%A0%D0%BE%D0%B1%D0%B5%D1%80%D1%82)) исходного изображения контента [9].

**Код модели, результаты и выводы.**

Код модели выложен на github [14]. Авторы статьи [10] реализовали алгоритм на смеси Lua и Matlab. Одной из задач обучения была реализация модели в популярной библиотеке PyTorch. Используя наработки автора [13] удалось собрать модель на PyTorch, оптимизировав ее под вычисления на GPU. В качестве feature extractor использовался предобученная на ImageNet архитектура VGG19. Для улучшения процесса обучения был изменен оптимизатор на Adam и осуществлен подбор гиперпараметров модели (соотношение стиля/контента, learning rate, регуляризатор фотореализма и т.д.)

К сожалению, мощности видеокарты на позволили работать с изображениями с высоким разрешением. Результат переноса стиля отражен на рисунке 5.

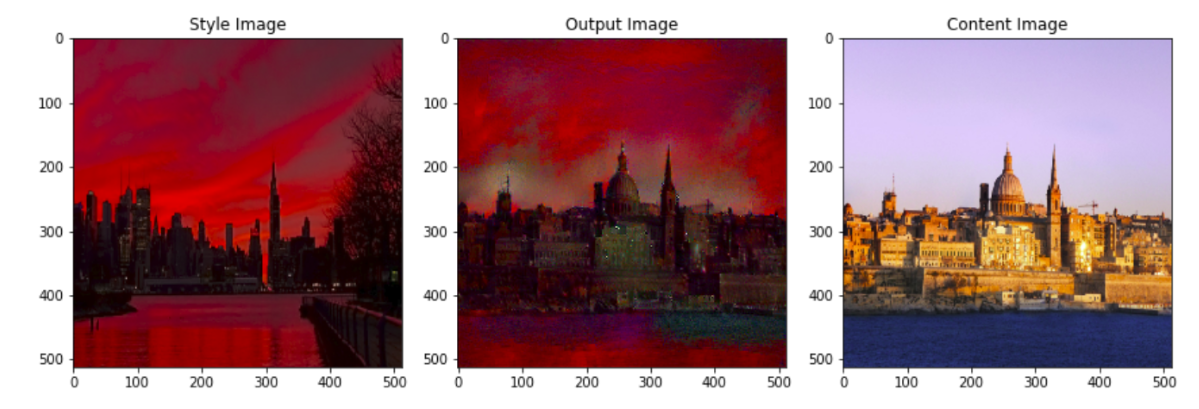


Рис. 5. Результат работы модели по переносу стиля.

На рис. 5 видно, что не удалось достигнуть модельных результатов, описанных в статье [10]. Фотореалистичность изображения нарушается — необходимо обучать модель на более производительных видеокартах с высоким разрешением. Необходимо далее работать с гиперпараметрами модели, чтобы достигнуть модельного результата.



Рис. 6. Модельные результаты работы из репозитория [7]

В дальнейшем хотелось бы исследовать возможность ускорения алгоритма работы по переносу стиля, экстракции стиля определенного художника/страны/города, а также генерирования изображений в определенном стиле при помощи GAN.

**Список источников**

[1] C. 231n. Assignment 3. <http://cs231n.github.io/>assignments2017/assignment3/.

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. CoRR, abs/1606.00915, 2016.

[3] L. Gatys, A. Ecker, and M. Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015.

[4] P. Gonchar. neural-art-mini. https://github.com/pavelgonchar/neural-art-mini, 2016.

[5] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. arXiv:1602.07360, 2016.

[6] J. Johnson. neural-style. https://github.com/jcjohnson/neural-style, 2015.

[7] F. Juan. deep-photo-styletransfer. https://github.com/luanfujun/deep-photo-styletransfer/tree/master/examples.

[8] T. Lee. Keras: Deep learning for python. https://github.com/fchollet/keras/blob/master/keras/applications/vgg19.py, 2017.

[9] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR06).

[10] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. arXiv preprint arXiv:1703.07511, 2017.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.

[12] Aly Kane, Amelia Lemionet, Fjori Shemaj, Photo Style Transfer in TensorFlow. <http://cs231n.stanford.edu/reports/2017/pdfs/412.pdf>

[13] https://github.com/yagudin/PyTorch-deep-photo-styletransfer

[14] <https://github.com/ppronchenko/study/blob/master/Diplom-Project/Deep_Photo_Style_Transfer.ipynb>