УДК 004.93
DOI: 10.25559/SITITO.18.202202.241-248

Научная статья

Использование предобученной нейросети (VGG16) для решения задачи переноса стиля изображения

М. Н. Б. Муаль
ФГАОУ ВО «Российский университет дружбы народов», г. Москва, Российская Федерация
Адрес: 117198, Российская Федерация, г. Москва, ул. Миклухо-Маклая, д. 6
bmouale@mail.ru

Аннотация

Задача переноса стиля изображения состоит в создании нового, ранее не существующего изображения путем комбинирования двух данных изображений – оригинального и стилевого. Оригинальное изображение формирует структуру, основные геометрические линии и формы результирующего изображения, в то время как стилевое изображение задает цвет и текстуру результата. Суть данного подхода заключается в том, что некая картинка преобразуется в новую с другим стилем, который был задан. Для решения таких задач обычно используют сверточные нейронные сети. На входе нейронной сети подаются две картинки: контент и стиль. Например, фотографию, а стилевое – картину знаменитого художника. Результирующим изображением в таком случае будет сцена, изображенная на исходной фотографии, выполненная в стилистике данной картины. Современные алгоритмы переноса стиля позволяют добиться хороших результатов, но результат работы таких алгоритмов оказывается либо неприемлемым ввиду чрезмерного искажения черт лица, либо слабо выраженным, не носящим характерные черты стилевого изображения. В этой работе мы рассмотрим, как адаптировать предобученную модель в решение задачи классификации и переноса изображения так, чтобы в результате было получено изображение, которое разукрасится в соответствии с исходным изображением. Наш основной вклад – это предложение нового метода обработки и переноса стиля изображений, основанного на предобученной модели VGG16.

Ключевые слова: распознавание изображения, перенос стиля, предобученная модель, сверточные нейронные сети, карта признаков, слои

Автор заявляет об отсутствии конфликта интересов.

Для цитирования: Муаль М. Н. Б. Использование предобученной нейросети (VGG16) для решения задачи переноса стиля изображения // Современные информационные технологии и ИТ-образование.2022.Т.18,№2.С.241-248.doi:https://doi.org/10.25559/SITITO.18.202202.241-248

Original article

Using a Pre-Trained Neural Network (VGG 16) to Solve the Image Style Transfer Problem

M. N. B. Mouale
Peoples' Friendship University of Russia, Moscow, Russian Federation
Address: 6 Miklukho-Maklaya St., Moscow 117198, Russian Federation
bmouale@mail.ru

Abstract

The task of image style transfer is to create a new, previously non-existent image by combining two given images – the original image and the styled image. The original image forms the structure, basic geometric lines and shapes of the resulting image, while the styled image sets the colour and texture of the result. The essence of this approach is that a certain image is transformed into a new one with a different style that has been set. To solve such problems, convolutional neural networks are usually used. The input to the neural network is two pictures: content and style. For example, a photograph, and the style is a painting by a famous artist. The resulting image would then be the scene depicted in the original picture, styled in the style of that picture. Modern style transfer algorithms give good results, but the result of such algorithms is either unacceptable due to excessive distortion of facial features, or weakly expressed, not bearing the characteristic features of the style image. In this paper, we consider how to adapt a pre-trained model in solving the image classification and transfer problem, so that the result is an image that is coloured according to the original image and highly pronounced. Our main contribution is to propose a new method for image processing and style transfer based on the pre-trained VGG16 model.

Keywords: face recognition, image recognition, convolutional neural networks, Regional Convolutional Neural Networks (R-CNN) model, bounding box, anchor

The author declares no conflict of interest.

For citation: Mouale M. N. B. Using a Pre-Trained Neural Network (VGG 16) to Solve the Image Style Transfer Problem. *Sovremennye informacionnye tehnologii i IT-obrazovanie = Modern Information Technologies and IT-Education.* 2022; 18(2):241-248. doi: https://doi.org/10.25559/SITITO.18.202202.241-248

Введение

VGG16 – модель сверточной нейронной сети, предложенная K. Simonyan и A. Zisserman из Оксфордского университета в исследовании [1]. Модель достигает точности 92.7% – топ-5 при тестировании на ImageNet в задаче распознавания объектов на изображении. Этот датасет состоит из более чем 14 миллионов изображений, принадлежащих к 1000 классам. VGG16 – одна из самых знаменитых моделей, отправленных на соревнование ILSVRC-2014. Она является улучшенной версией AlexNet, в которой заменены большие фильтры (размера 11 и 5 в первом и втором сверточном слое соответственно) на несколько фильтров размера 3x3, следующих один за другим. Сеть VGG16 обучалась на протяжении нескольких недель при использовании видеокарт Nvidia TITAN Black[1].

В настоящей статье изложены результаты нового метода обработки и переноса стиля изображений основан на предобученной модель VGG16.

В первой части мы подробно рассмотрим готовые предобученные модели в Pytorch.

Во второй части мы рассмотрим пример использования предобученных моделей для реализации проектов и тонкая настройка (Fine Tuning) проекта.

В третьей части работы мы реализуем проект сети с использованием нового метода обработки и переноса стиля изображений основанный на предобученной модель VGG16.

Предобученные модели в библиотеке Pytorch

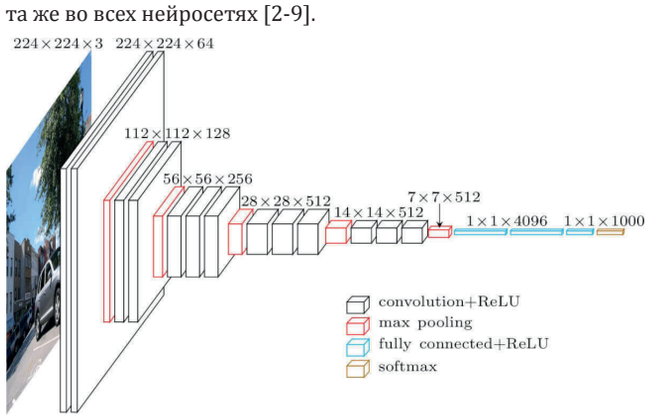
PyTorch – фреймворк машинного обучения для языка Python с открытым исходным кодом, созданный на базе Torch. Используется для решения различных задач: компьютерное зрение, обработка естественного языка. Разрабатывается преимущественно группой искусственного интеллекта Facebook. Написан на Python, C++, CUDA.

Датасет

ImageNet – набор данных, состоящий из более чем 15 миллионов размеченных высококачественных изображений, разделенных на 22000 категорий. Изображения были взяты из интернета и размечены вручную людьми-разметчиками с помощью краудсорсинговой площадки Mechanical Turk от Amazon. В 2010 году, как часть Pascal Visual Object Challenge, началось ежегодное соревнование – ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). В ILSVRC используется подвыборка из ImageNet размером 1000 изображений в каждой из 1000 категорий. Таким образом, тренировочный сет состоял из примерно 1.2 миллионов изображений, проверочный – 50000 изображений, тестовый – 150000 изображений.

Архитектура VGG16

Архитектура VGG16 представлена на рисунке 1. Далее изображения проходят через стек сверточных слоев, в которых используются фильтры с очень маленьким рецептивным полем размера 3x3 (который является наименьшим размером для получения представления о том, где находится право/лево, верх/низ, центр). В одной из конфигураций используется сверточный фильтр размера 1x1, который может быть представлен как линейная трансформация входных каналов (с последующей нелинейностью). Сверточный шаг фиксируется на значении 1 пиксель. Пространственное дополнение (padding) входа сверточного слоя выбирается таким образом, чтобы пространственное разрешение сохранялось после свертки, то есть дополнение равно 1 для 3x3 сверточных слоев. Пространственный пулинг осуществляется при помощи пяти maxpooling слоев, которые следуют за одним из сверточных слоев (не все сверточные слои имеют последующие maxpooling). Операция max-pooling выполняется на окне размера 2x2 пикселей с шагом 2. После стека сверточных слоев (который имеет разную глубину в разных архитектурах) идут три полносвязных слоя: первые два имеют по 4096 каналов, третий – 1000 каналов (так как в соревновании ILSVRC требуется классифицировать объекты по 1000 категориям; следовательно, классу соответствует один канал). Конфигурация полносвязных слоев одна и та же во всех нейросетях [2-9].



Р и с. 1. Архитектура VGG16[2]

F i g. 1. VGG16 architecture

Конфигурации сверточных сетей

Конфигурации сверточных сетей представлены на рисунке 2. Каждая сеть соответствует своему имени (A-E). Все конфигурации имеют общую конструкцию, представленную в архитектуре, и различаются только глубиной: от 11 слоев с весами в сети A (8 сверточных и 3 полносвязных слоя) до 19 (16 сверточных и 3 полносвязных слоя). Ширина сверточных слоев (количество каналов) относительно небольшая: от 64 в первом слое до 512 в последнем с увеличением количества каналов в 2 раза после каждого max-pooling слоя. Далее блок из одного слоя – Adaptive Average Pooling, а затем третий блок «классификатор» – из последовательности полносвязных слоев 4096 нейронов и последний на 1000 нейронов – обучена на наборе данных ImageNET – набор из 1000 классов. Загрузим готовую обученную модель и попытаемся использовать ее для своих целей. torchvision. models. vgg16(pretrained=True).eval() модели

[1] Милютин И. VGG16 – сверточная сеть для выделения признаков изображений [Электронный ресурс] // Neurohive. 23.11.2018. URL: https://neurohive.io/ru/vidy-nejrosetej/vgg16-model (дата обращения: 14.05.2022).

[2] Там же.

находятся в модуле models – загружаем vgg16(pretrained=True) – с обученными весами для работы – eval(). class_idx = json.load(open(«imagenet_class_index.json»)) imagenet_class_index.json – здесь находится информация о классах в формате «идентификатор» : «название класса». И находим общее количество классов.

transforms.ToTensor(), transforms.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)).

Для каждой картинки применяем набор трансформаций: сначала превращаем картинку в тензор и производим нормализацию с параметрами, которые наиболее подходят для обработки картинки. Вывод архитектуры, все то, что мы рисовали: vgg16.

Слой AdaptiveAvgPool2d – любой вход преобразуется к размеру 7 на 7 – при входящей в слой матрице 21X14 окно свертки будет 2 на 3 и на выходе – 7X7, 2 на 2 на выходе все равно будет матрица 7на7. То есть при вхождении любого тензора в этот слой на выходе получается тензор 7 на 7. Получается универсальная система: подаем любую картинку на вход – на выход будет матрица 7 на 7, а это дает возможность добавить свои собственные слои.

pk = transform(pk). Применив указанный набор трансформаций, получаем тензор 3 на 600 на 600.

vgg16.avgpool(pk.unsqueeze(0)).

Если обратиться к нему напрямую, то на выходе будет 1 на 3 на 7 на 7. То есть размер входящей картинки может быть любым, и получим тот же вектор 3 на 7 на 7. tns = np.random.random(size=(1,3,2,2)). И, пробуя другую размерность, все равно такую же размерность. nn.Softmax()(vgg16(im.unsqueeze(0))).view(-1). im.unsqueeze(0) – добавляем размерность и подаем модель. И вызываем nn.Softmax – view(-1) и растягиваем в один вектор. torch.argsort(probas, descending=True). Отсортируем наш массив по вероятностям и выведем 5 максимальных полученных вероятностей: idx2label[sortedOuputs[i]], probas[sortedOuputs[i]])).

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Р и с. 2. Конфигурации сверточных сетей [1, С. 3]

F i g. 2. Convolutional Network Configurations [1, P. 3]

Использования предобученных моделей для реализации проектов и тонкая настройка (Fine Tuning) проекта

Описание метода обработки и переноса стиля изображений основан на предобученной модель VGG16

Мы рассматриваем задачу определения породы собак по изображениям, 60 пород. В стандартном наборе – если и есть породы, то хорошо если 3, 4, а скорее всего есть только класс «собака». Можно взять модель без обученных весов и обучить, но это и ресурсы, и нужна очень большая база: 10000 рисунков каждой породы и неограниченные ресурсы Fine Tuning позволяет взять готовую модель, оставить сверточные слои, а классификатор поставить свой, удалить классификатор на 1000 классов и заменить на классификатор из Dense слоя на 60 нейронов. Сверточные слои заморозим, они уже умеют распознавать признаки, но нужно построить модель со своими классами. Можно обучить только классификатор – это 1 вариант.

Второй вариант – обучать всю сеть, но с маленьким шагом: уровня 1e-5 – 1e-7, чтобы обученные веса не потеряли своих значений и подстроились под нашу задачу.

Возможен симбиоз: сначала обучаем только классификатор с обычным шагом, а потом уменьшаем наш шаг и дообучаем полностью всю модель для смещения весов и подстройки под наше изображение [10-20].

Убираем стандартный классификатор, и для распознания картинки из базы CIFAR10 нужно добавить классификатор – полносвязный слой на 10 нейронов. Итак, на вход: 3 на 32 на 32. И размерность на первом слое 64-32-32, затем будет после MaxPooling – 128-16-16, MaxPooling – 256-8-8, MaxPooling – 512-4-4, MaxPooling – 512-2-2 – еще один MaxPooling слой -512-1-1 и из AdaptiveAvgPool2d выход 512-7-7 – будет здесь всегда. Сюда нужно добавить полносвязный слой на 10 нейронов.

Таким образом, получаем готовую модель под решение своих задач со своим собственный классификатор. Но маленьких моделей слишком сильная сверточная сеть, поэтому она дает лишь 61% точность. Можно увеличить размер входящей картинки или убавить количество слоев – тогда будет возможно увеличить точность.

Реализация проекта сети с переносом стиля изображения

Перенос стиля

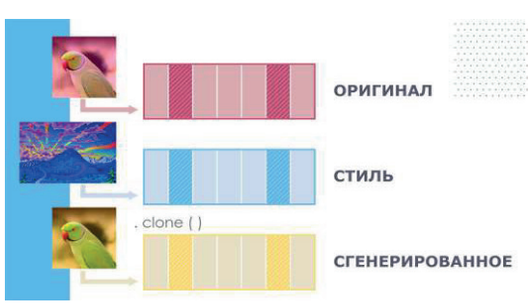
Если в качестве примера будет нарисованный попугай в стиле второго рисунка, то какой будет выход у рисунка? Здесь у нас нет такой базы, по которой будет проводиться обучение. Все, что у нас есть, это два изображения: оригинал и стиль изображения. Недостаточно просто наложить две фотографии, нужно, чтобы каждый объект: крыло, клюв, черное пятно за спиной – также раскрашивался в соответствии с той цветовой палитрой, как рисунок стиля, то есть цвет клюва должен быть цветом неба рисунка стиля. А ошибка считается на основе ошибок двух частей ошибки изображения и ошибки стиля (см. рисунки 3-5) [21-25].

Чтобы сохранить контекст исходного изображения при проходе через механизм сверточных слоев, нужно выделить конту-

ры исходного изображения. Готовую модельку самый первый сверточный слой – вернет 64 карты признака, то, что выделяет первый сверточный слой на исходном изображении и наиболее важные светлые изображения, так ищем контуры на исходном изображении – берем и прогоняем через сеть VGG16, или VGG19 – с добавлением еще одного сверточного слоя в каждый блок.

Метод получает картинку и извлекает список признаков из каждого слоя. Если индекс попадает в список индексов, то добавляют признаки в список по всем слоям – 5 элементов таких слоев.

Р и с. 3. Схема постановки задачи
F i g. 3. Problem statement scheme



Р и с. 4. Схема постановки задачи
F i g. 4. Problem statement scheme

Р и с. 5. Схема постановки задачи
F i g. 5. Problem statement scheme

Описание и результат выполнения метода переноса стиля изображения

Мы рассматриваем случай, когда есть исходное изображение, на нём изображен попугай. Мы обнаружим попугая на изображении, если видим контуры: голову, клюв, глаз, крыло, некоторые формы, которые говорят о том, что на этой картинке изображен попугай. При этом, если этот попугай не будет зелёный, а допустим, красный, то это нам не должно мешать определить, есть на этом изображение попугай, или нет. В другом случае, нужно определить стиль (цвета, текстура изображения, способ исполнения). Наша цель – это увидеть на конечном изображение попугая и характерные признаки.

Для контента исходного изображения (то, что хотим сохранить), нейросеть выделяет признаки на изображение и среди этих признаков довольно много признаков, связанных с контурами изображения, т.е. с контурами попугая. Можно взять эти признаки с разных слоев нейросети и сопоставить. Возьмём оригинал изображения и сравним признаки, изученные нейросетью для исходного изображения и для целевого изображении. И с помощью среднеквадратической ошибкой посчитаем разницу. Т.е. если какой-то признак пропал на изображении, то разница получится большая, а если контуры были сохранены, то ошибка будет меньше. Просчитаем функцию ошибки **MSE** по всем картам признаков, которые будем доставить в сети.

Ошибку исходного изображения через 19 сверточных слоев и соберем набор карт признаков с определенных слоев – их можно определить рандомно или случайно – это не принципиально, для нашего исходного изображения наберем такие примеры. То же самое мы делаем для нашего целевого изображения. И останется только сравнить попарно эти наборы и вычислить стандартную MSE ошибку, а это в плане контента важно, чтобы сохранились контуры. Мы поможем нашей сети – на выход мы копируем оригинальное изображение. Ошибка изображения будет работать на то, чтобы контуры рисунка оставались на картинке стиля.

Стилевая ошибка: мы будем извлекать ту же информацию, что и при получении ошибки изображения, но значение будет другое. Скрещивая попугая и горы, нужно получить информацию о цвете, для этого используется матрица Грама – матрица корреляции. С первого сверточного слоя получаем набор из 64 карт признаков: дальше нулевой фильтр – картинки с попугаями – поэлементно умножаем элементы на картинках и суммируем. Черный – 0, не пересекаются – значение будет близкое к нулю. Чем больше элементы пересекаются, тем больше цифра корреляции. И так для всех возможных вариантов – на выходе получится матрица 64-64.

Определим матрицу корреляций и матрицу Грама:

Для нахождения близости по стилю предлагают использовать матрицу Грама, вычисляемую по следующей формуле:

$$G_l(x) = \frac{1}{H_l W_l} F_l(x)^T F_l(x) \in R^{cl \times c_l}$$

Если интегрировать $F_l(x)$ как матрицу, состоящую из объектов размерности CI, то матрица Грама будет равна нецентрированной матрице ковариаций. Таким образом, матрица Грама содержит информацию о том, какие каналы карты признаков зависят друг от друга, где на пересечении рассчитанные нами значения корреляции, как для изображения стиля и для целевого изображения. 5 наборов карт признаков: для каждого из них считаем матрицу корреляции, и для целевого изображения то же самое 5 наборов – признаки, а дальше применяем ошибку MSE для матриц корреляции попарно уже позволяет

накладывать на матрицу изображения. Метод получает картинку и извлекает список признаков из каждого слоя. Если индекс попадает в список индексов, то добавляют признаки в список по всем слоям – 5 элементов таких слоев.

Результат:

Оптимизируем веса так, что попиксельное сходство падает и замещается стилем, но не вымещается.

Loss=content loss + style loss

Обнуляем градиенты и считаем веса, затем изменяем значения. Проводим обучения: ошибка контента падает. Изменения происходят небольшие, но объекты становятся все более уникальными. Появляется контрастность.

Обучение с различными шагами

Загружаем модель и запускаем модель обучения. Через модель пропускаем изображение, получаем карты признаки. Затем признаки основной картинки, стилевой – на первом шаге признаки изображения и целевого изображения будет равны. По всем тройкам проходим – получаем размер, сверток, получаем размер и количество каналов, изменяем размер, и получаем матрицу корреляции. Считаем коэффициенты, насколько стиль буде¨ ¨ревалировать над контентом.

$$styleLoss+ = contentCriteria(f_1, f_2) \times kf_2$$

Далее обнуляем градиенты и считаем веса. Затем получаем изменяем значения. Проводим обучения: ошибка контента падает. Изменения происходят небольшие, но объекты становятся все более уникальными. Появляется контрастность.

Первые 100 эпох

```
Шаг [10/600], Ошибка для оригинала: 108.6528, Ошибка для стиля: 1.5729523187009978e+19
Шаг [20/600], Ошибка для оригинала: 142.9272, Ошибка для стиля: 6.421507996269937e+18
Шаг [30/600], Ошибка для оригинала: 143.3061, Ошибка для стиля: 3.72236005707402 44e+18
Шаг [40/600], Ошибка для оригинала: 145.3548, Ошибка для стиля: 2.437901502535172e+18
Шаг [50/600], Ошибка для оригинала: 144.7937, Ошибка для стиля: 1.7428130663214612e+18
Шаг [60/600], Ошибка для оригинала: 145.2337, Ошибка для стиля: 1.3452978314385818e+18
Шаг [70/600], Ошибка для оригинала: 145.3654, Ошибка для стиля: 1.0957929420119081e+18
Шаг [80/600], Ошибка для оригинала: 145.6941, Ошибка для стиля: 9.267002857268183e+17
Шаг [90/600], Ошибка для оригинала: 145.9705, Ошибка для стиля: 8.040705169303798e+17
Шаг [100/600], Ошибка для оригинала: 146.1997, Ошибка для стиля: 7.1115731761732e+17
```

Первые 200 эпох

```
Шаг [110/600], Ошибка для оригинала: 146.4362, Ошибка для стиля: 6.3751470884847 62e+17
Шаг [120/600], Ошибка для оригинала: 146.6672, Ошибка для стиля: 5.774544359370 26e+17
Шаг [130/600], Ошибка для оригинала: 146.8909, Ошибка для стиля: 5.272496355011461e+17
Шаг [140/600], Ошибка для оригинала: 147.0867, Ошибка для стиля: 4.846499508361625 6e+17
Шаг [150/600], Ошибка для оригинала: 147.2752, Ошибка для стиля: 4.480090694379 1104e+17
Шаг [160/600], Ошибка для оригинала: 147.4387, Ошибка для стиля: 4.160059968050954e+17
Шаг [170/600], Ошибка для оригинала: 147.6069, Ошибка для стиля: 3.878270256136 8474e+17
Шаг [180/600], Ошибка для оригинала: 147.7597, Ошибка для стиля: 3.628731281849 7126e+17
Шаг [190/600], Ошибка для оригинала: 147.9002, Ошибка для стиля: 3.406313136251208e+17
Шаг [200/600], Ошибка для оригинала: 148.0392, Ошибка для стиля: 3.206836443636326e+17
```

Первые 300 эпох

```
Шаг [210/600], Ошибка для оригинала: 148.1763, Ошибка для стиля: 3.026618442233938e+17
Шаг [220/600], Ошибка для оригинала: 148.3067, Ошибка для стиля: 2.863233763125 4938e+17
Шаг [230/600], Ошибка для оригинала: 148.4333, Ошибка для стиля: 2.715033339596636e+17
Шаг [240/600], Ошибка для оригинала: 148.5494, Ошибка для стиля: 2.580330967486 9555e+17
Шаг [250/600], Ошибка для оригинала: 148.6617, Ошибка для стиля: 2.456899104958 0544e+17
Шаг [260/600], Ошибка для оригинала: 148.7633, Ошибка для стиля: 2.343453041392 3533e+17
Шаг [270/600], Ошибка для оригинала: 148.8704, Ошибка для стиля: 2.239340800754 3194e+17
Шаг [280/600], Ошибка для оригинала: 148.9707, Ошибка для стиля: 2.143454796877 66e+17
Шаг [290/600], Ошибка для оригинала: 149.0706, Ошибка для стиля: 2.055044613076 4186e+17
Шаг [300/600], Ошибка для оригинала: 149.1621, Ошибка для стиля: 1.973299531323 8016e+17
```

Первые 400 эпох

```
Шаг [310/600], Ошибка для оригинала: 149.2527, Ошибка для стиля: 1.897239611881554e+17
Шаг [320/600], Ошибка для оригинала: 149.3340, Ошибка для стиля: 1.826414742177 0547e+17
Шаг [330/600], Ошибка для оригинала: 149.4171, Ошибка для стиля: 1.760252145371 7094e+17
Шаг [340/600], Ошибка для оригинала: 149.4974, Ошибка для стиля: 1.698578132988 068e+17
Шаг [350/600], Ошибка для оригинала: 149.5741, Ошибка для стиля: 1.640876793555 45344e+17
Шаг [360/600], Ошибка для оригинала: 149.6497, Ошибка для стиля: 1.586728766464 3277e+17
Шаг [370/600], Ошибка для оригинала: 149.7261, Ошибка для стиля: 1.535837011779 2563e+17
Шаг [380/600], Ошибка для оригинала: 149.7966, Ошибка для стиля: 1.487875111984 8243e+17
Шаг [390/600], Ошибка для оригинала: 149.8646, Ошибка для стиля: 1.442617151801262e+17
Шаг [400/600], Ошибка для оригинала: 149.9358, Ошибка для стиля: 1.399824846442 9875e+17
```

Первые 500 эпох

```
Шаг [410/600], Ошибка для оригинала: 150.0058, Ошибка для стиля: 1.359320126565 0085e+17
Шаг [420/600], Ошибка для оригинала: 150.0763, Ошибка для стиля: 1.320906024969 8304e+17
Шаг [430/600], Ошибка для оригинала: 150.1931, Ошибка для стиля: 1.284894270381097e+17
Шаг [440/600], Ошибка для оригинала: 150.3253, Ошибка для стиля: 1.252618450145 1162e+17
Шаг [450/600], Ошибка для оригинала: 150.4017, Ошибка для стиля: 1.220368227914 2195e+17
Шаг [460/600], Ошибка для оригинала: 150.5404, Ошибка для стиля: 1.193827219810 5498e+17
Шаг [470/600], Ошибка для оригинала: 150.7326, Ошибка для стиля: 1.173216101253 7754e+17
Шаг [480/600], Ошибка для оригинала: 151.0229, Ошибка для стиля: 1.181253445353472e+17
Шаг [490/600], Ошибка для оригинала: 151.4545, Ошибка для стиля: 1.235950197465 4157e+17
Шаг [500/600], Ошибка для оригинала: 151.2945, Ошибка для стиля: 1.193892846910 8326e+17
```

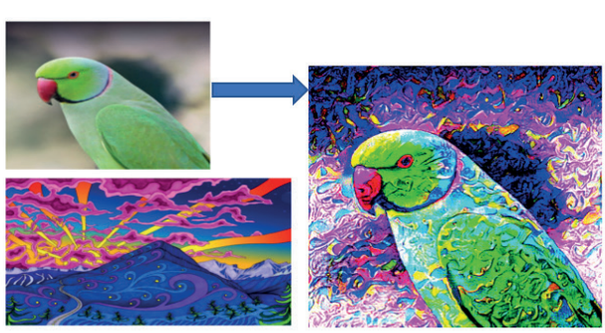
Первые 500 эпох

```
Шаг [510/600], Ошибка для оригинала: 151.3776, Ошибка для стиля: 1.169802547146 2605e+17
Шаг [520/600], Ошибка для оригинала: 149.9437, Ошибка для стиля: 1.105410078162 2886e+17
Шаг [530/600], Ошибка для оригинала: 150.4061, Ошибка для стиля: 1.114225495337 3286e+17
Шаг [540/600], Ошибка для оригинала: 154.3599, Ошибка для стиля: 3.144260688847831e+17
Шаг [550/600], Ошибка для оригинала: 149.1942, Ошибка для стиля: 2.938603567222 62e+17
Шаг [560/600], Ошибка для оригинала: 153.1192, Ошибка для стиля: 2.548814841066 2912e+17
Шаг [570/600], Ошибка для оригинала: 151.1968, Ошибка для стиля: 1.649420170701 0458e+17
Шаг [580/600], Ошибка для оригинала: 150.9484, Ошибка для стиля: 1.371169941335 5725e+17
Шаг [590/600], Ошибка для оригинала: 151.3110, Ошибка для стиля: 1.222953282830 336e+17
Шаг [600/600], Ошибка для оригинала: 151.0451, Ошибка для стиля: 1.095364235556 2906e+17
```

Р и с.6. Результаты вариации Loss с различными шагами

F i g. 6. Loss Variation Results with Various Steps

Изображение с перенесенным стилем



Загрузка изображения из набора или с компьютера

Распознание изображения с перенесенным стилем

Р и с. 7. Попугай разукрасился в соответствии с исходным изображением

F i g. 7. The parrot is painted in accordance with the original image

В результате получается попугай, который разукрасился в соответствии с исходным изображением.

Заключение

В работе была предложен метод обработки и переноса стиля изображений, основанный на предобученной модель VGG16. Метод позволяет получить картинку и извлекать список признаков из каждого слоя. Если индекс попадает в список индексов, то добавляются признаки в список. По всем слоям – 5 элементов таких слоев. Метод был, применим для переноса стиля изображения попугая. После 600 эпох обучения, получаем следующие значения ошибки для оригинала изображения и для стиля: ошибка для оригинала равна 151.0451, ошибка для стиля равна 1.0953. Это довольно хороший результат, который показывает эффективность работы предложенного метода. В результате получаем попугая, который разукрасился в соответствии с исходным изображением.

References

[1] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio Y., LeCun Y. (eds.) *Proceedings of the 3rd International Conference on Learning Representations (ICLR-2015).* San Diego, CA, USA; 2015. p. 1-14. (In Eng.) doi: https://doi.org/10.48550/arxiv.1409.1556

[2] Tao Y. Image Style Transfer Based on VGG Neural Network Model. *2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA).* IEEE Computer Society, Dalian, China; 2022. p. 1475-1482. (In Eng.) doi: https://doi.org/10.1109/AEECA55500.2022.9918891

[3] Li M. -A., Xu D. -Q. A Transfer Learning Method based on VGG-16 Convolutional Neural Network for MI Classification. *2021 33rd Chinese Control and Decision Conference (CCDC).* IEEE Computer Society; 2021. p. 5430-5435. (In Eng.) doi: https://doi.org/10.1109/CCDC52312.2021.9602818

[4] Girshick R., Donahue J., Darrell T., Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Columbus, OH, USA; 2014. p. 580-587. (In Eng.) doi: https://doi.org/10.1109/CVPR.2014.81

[5] Li S.Z., Jain A.K. Handbook of Face Recognition. Springer, London; 2011. 2nd Ed. 699 p. (In Eng.) doi: https://doi.org/10.1007/978-0-85729-932-1

[6] Erhan D., Szegedy C., Toshev A., Anguelov D. Scalable Object Detection Using Deep Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE Computer Society, Columbus, OH, USA; 2014. p. 2155-2162. (In Eng.) doi: https://doi.org/10.1109/CVPR.2014.276

[7] He K., Zhang X., Ren S., Sun J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2015; 37(9):1904-1916. (In Eng.) doi: https://doi.org/10.1109/TPAMI.2015.2389824

[8] Deng J., Russakovsky O., Krause J., Bernstein M.S., Berg A., Fei-Fei L. Scalable multi-label annotation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'14).* Association for Computing Machinery, New York, NY, USA; 2014. p. 3099-3102. (In Eng.) doi: https://doi.org/10.1145/2556288.2557011

[9] Mouale M.N.B., Kozyrev D.V., Houankpo H.G.K., Nibasumba E. Development of a Neural Network Method in the Problem of Classification and Image Recognition. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2021; 17(3):507-518. (In Russ., abstract in Eng.) doi: https://doi.org/10.25559/SITITO.17.202103.507-518

[10] Russakovsky O., Deng J., Huang Z., Berg A.C., Fei-Fei L. Detecting Avocados to Zucchinis: What Have We Done, and Where Are We Going? *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV'13).* IEEE Computer Society, USA; 2013. p. 2064-2071. (In Eng.) doi: https://doi.org/10.1109/ICCV.2013.258

[11] Su H., Deng J., Fei-Fei L. Crowdsourcing Annotations for Visual Object Detection. *AAAI Human Computation Workshop*. AAAI Press, Palo Alto, California; 2012. p. 1-7. Available at: http://vision.stanford.edu/pdf/bbox_submission.pdf (accessed 14.05.2022). (In Eng.)

[12] Deng J., Dong W., Socher R., Li L.-J., Kai Li, Fei-Fei L. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society; 2009. p. 248-255. (In Eng.) doi: https://doi.org/10.1109/CVPR.2009.5206848

[13] Krizhevsky A., Sutskever I., Hinton G.E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. 2017; 60(6):84-90. (In Eng.) doi: https://doi.org/10.1145/3065386

[14] Mäenpää T. The Local Binary Pattern Approach to Texture Analysis: Extensions and Applications. *Acta Universitatis Ouluensis: Series C, Technica*. 2003; 187:1-75. (In Eng.)

[15] Rothe R., Guillaumin M., Van Gool L. Non-maximum Suppression for Object Detection by Passing Messages Between Windows. In: Cremers D., Reid I., Saito H., Yang M.H. (eds.) *Computer Vision – ACCV 2014. ACCV 2014. Lecture Notes in Computer Science*. Vol. 9003. Springer, Cham; 2015. p. 290-306. (In Eng.) doi: https://doi.org/10.1007/978-3-319-16865-4_19

[16] Shan C., Gong S., McOwan P.W. Facial expression recognition based on Local Binary Patterns: A comprehensive study. *Image and Vision Computing*. 2009; 27(6):803-816. (In Eng.) doi: https://doi.org/10.1016/j.imavis.2008.08.005

[17] Maturana D., Mery D., Soto Á. Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor Classification. *2009 International Conference of the Chilean Computer Science Society*. IEEE Computer Society, Santiago, Chile; 2009. p. 125-132. (In Eng.) doi: https://doi.org/10.1109/SCCC.2009.21

[18] Visani M., Garcia C., Jolion J.M. Bilinear Discriminant Analysis for Face Recognition. In: Singh S., Singh M., Apte C., Perner P. *Pattern Recognition and Image Analysis. ICAPR 2005. Lecture Notes in Computer Science.* Vol. 3687. Springer, Berlin, Heidelberg; 2005. p. 247-256. (In Eng.) doi: https://doi.org/10.1007/11552499_28

[19] Chen D., Cao L. Face recognition based on multi-module singular value features and probabilistic subspaces analysis. *2011 4th International Congress on Image and Signal Processing*. IEEE Computer Society, Shanghai, China; 2011. p. 1508-1512. (In Eng.) doi: https://doi.org/10.1109/CISP.2011.6100445

[20] Alizadeh F., Nalousi S., Savari C. Face Detection in Color Images using Color Features of Skin. *International Journal of Computer and Information Engineering*. 2011; 5(4):366-372. (In Eng.) doi: https://doi.org/10.5281/zenodo.1063276

[21] Xu L.S., Meng M.Q.-H., Wang K.Q. Pulse Image Recognition Using Fuzzy Neural Network. *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE Computer Society, Lyon, France; 2007. p. 3148-3151. (In Eng.) doi: https://doi.org/10.1109/IEMBS.2007.4352997

[22] Lou G., Shi H. Face image recognition based on convolutional neural network. *China Communications*. 2020; 17(2):117-124. (In Eng.) doi: https://doi.org/10.23919/JCC.2020.02.010

[23] Houankpo H.G.K., Kozyrev D.V., Nibasumba E., Mouale M.N.B., Sergeeva I.A. A Simulation Approach to Reliability Assessment of a Redundant System with Arbitrary Input Distributions. In: Vishnevskiy V.M., Samouylov K.E., Kozyrev D.V. *Distributed Computer and Communication Networks. DCCN 2020. Lecture Notes in Computer Science.* Vol. 12563. Springer, Cham; 2020. p. 380-392. (In Eng.) doi: https://doi.org/10.1007/978-3-030-66471-8_29

[24] Arulogun O.T., Omidiora E.O., Olaniyi O.M., Ipadeola A.A. Development of Security System using Facial Recognition. *The Pacific Journal of Science and Technology*. 2008; 9(2):377-385. (In Eng.)

[25] Panetto H., Cecil J. Information systems for enterprise integration, interoperability and networking: theory and applications. *Enterprise Information Systems.* 2013; 7(1):1-6. (In Eng.) doi: https://doi.org/10.1080/17517575.2012.684802

Поступила 14.05.2022; одобрена после рецензирования 06.07.2022; принята к публикации 15.07.2022.
Submitted 14.05.2022; approved after reviewing 06.07.2022; accepted for publication 15.07.2022.

Об авторе:

Муаль Мутуама Нда Бьенвеню, аспирант кафедры прикладной информатики и теории вероятностей, факультет физико-математических и естественных наук, ФГАОУ ВО «Российский университет дружбы народов» (117198, **Российская Федерация**, г. Москва, ул. Миклухо-Маклая, д. 6), **ORCID: https://orcid.org/0000-0002-7230-5714,** bmouale@mail.ru

Автор прочитал и одобрил окончательный вариант рукописи.

About the author:

Moutouama N'dah B. Mouale, Postgraduate Student of the Department of Applied Probability and Informatics, Faculty of Science, Peoples' Friendship University of Russia (6 Miklukho-Maklaya St., Moscow 117198, Russian Federation), **ORCID: https://orcid.org/0000-0002-7230-5714,** bmouale@mail.ru

The author has read and approved the final manuscript.