УДК 004.932

**Куприянов И. В, Давыденко А. А**

# Колоризация черно-белых изображений с помощью нейросетей

**1. Введение.** Колоризация черно-белых фотографий – объемная и трудоемкая задача. Художники тратят большое количество времени для восстановления цвета, так как для того, чтобы понять, какой именно оттенок нужно присвоить каждому фрагменту фотографии, часто приходится находить цветные аналоги этих фрагментов. На одно изображение может уйти месяц работы, если художник хочет добиться максимальной точности. Автоматизация этого процесса [1] позволит значительно сократить время людей, занимающихся колоризацией вручную.

В данной работе используются *сверточные нейронные сети* [2] для построения модели глубокого обучения [3], которая, принимая на вход черно-белое изображение, на выход выдает раскрашенное.

Так как раскрашивание любых фотографий требует объемного набора данных, было принято решение для начала работать с изображениями определенного класса: фотографии улиц, домов, переулков и прочей городской среды.

**2. Подготовка данных.** В первую очередь необходимо задать цветовое пространство для представления изображений.

**2.1. Цветовое пространство.** Если работать в классическом $RGB$, то на вход модели должно подаваться три матрицы, так как черно-белые изображения состоят из оттенков серого, которые получаются комбинацией красного, зеленого и синего в разных пропорциях. На выходе модели так же будет три матрицы для компонент $R, G, B$.

Но кроме $RGB$ существуют и другие цветовые пространства, например цветовое пространство $Lab$. Здесь используется иная цветовая модель, где компонента $L$ отвечает за значение «светлоты», а

*Куприянов Илья Владимирович* – студент, Санкт-Петербургский государственный университет; e-mail: st075958@student.spbu.ru, тел.: +7(999)520-55-17

*Давыденко Александр Александрович* – старший преподаватель, Санкт-Петербургский государственный университет; e-mail: a.davydenko@spbu.ru, тел.: +7(911)210-57-92

компоненты $a$ и $b$ – за положение цвета в диапазоне от зеленого до красного и от синего до желтого соответственно.

В случае черно-белых изображений на вход модели будет подаваться всего лишь одна матрица, так как компонента $L$, по сути, и является черно-белым представлением изображения, а компоненты цветового пространства $a$ и $b$ отсутствуют. На выходе достаточно получать только компоненты $a$ и $b$, так как за слой $L$ можно взять исходное изображение. Поэтому выбор этого цветового пространства позволит сократить объем вычислений и количество затрачиваемой памяти.

**2.2. Набор обучающих данных.** В открытом доступе не было найдено готового набора данных, который подходит для поставленной задачи. Поэтому данные собирались самостоятельно. Для того чтобы понять, в каких пропорциях составлять *датасет*, было скачено около 300 старых фотографий городской среды, после чего использовался классификатор *InceptionResNetV2* [4], ставящий в соответствие изображению вектор, содержащий 1000 признаков, где каждый признак – вероятность принадлежности определенному классу. Для каждого изображения было выбрано три самых вероятных класса, затем подсчитано количество классов.

**2.3. Обработка данных.** Скаченные с помощью утилиты *Yandex Grabber* [5] изображения были преобразованы в квадратные, размером $256 \times 256$ пикселей. В цветовом пространстве *CIE Lab* значение цветов меняются от $-128$ до 128. Если разделить каждое значение на 128, то они окажутся в границах от $-1$ до 1. Это сделано для того, чтобы можно было сравнивать погрешность вычисления, так как на выходе сети мы получаем значения именно в таких границах. Для достижения хороших результатов нужно много обучающих примеров. Для искусственного увеличения размера датасета используют прием, который называется *аугментация данных*. Главная идея метода – получение нескольких изображений из одного с помощью разных манипуляций. В этой работе для аугментации данных обучающие изображения преобразуются с помощью сдвига относительно центра, приближения, поворота и отражения по вертикали.

**3. Первая версия модели.** Для работы с изображениями в глубоком обучении используются *сверточные нейронные сети*. Они справляются намного лучше линейных моделей, потому что сохраняют пространственное представление изображения.

**3.1. Архитектура.** В этой работе используется модель, построенная на архитектуре [6] *Autoencoder* со вложенными сверточными слоями. Первый блок – кодировщик, состоит из *Convolutional* слоев. Вместо *MaxPooling* используются те же сверточные слои, но с шагом 2. Этот подход позволяет не терять пространственную информацию и избавиться от искажений при понижении размера изображений. Второй блок – декодировщик, нужен для повышения размера изображения, так как в первом блоке мы уменьшаем размер в 8 раз. Для повышения используются три слоя *UpSampling*.

**3.2. Обучение и результаты.** Первая версия модели обучалась с помощью *google colab*, без использования *GPU*. Размер батча экспериментальным путем был выбран равным 100. Обучение на 1000 эпохах заняло около 8 часов. На рис. 1 видно, что на первых 20 эпохах accuracy валидационного множества больше, чем при дальнейшем обучении. Но при попытке раскрасить черно-белые фотографии они получались тусклыми, и цвет почти не было заметно.

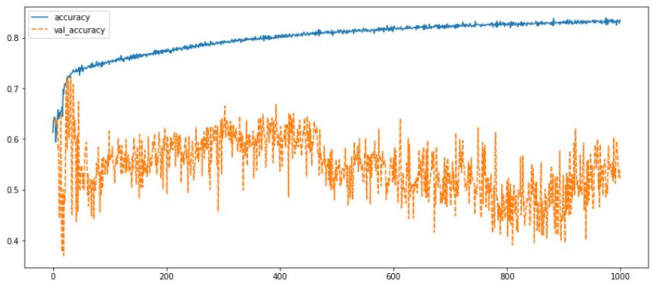


**Рис. 1.** График accuracy в зависимости от эпохи обучения

Для 400 эпох accuracy получилась равной 0,61. В цифрах это не много, но визуальные результаты выглядят приемлемо, хотя, конечно, есть определенные проблемы.

Если смотреть на раскрашенные с помощью модели черно-белые изображения, у которых нет цветных аналогов, нельзя оценить, насколько точно она справляется с передачей цвета. Поэтому были взяты цветные изображения, преобразованы в черно-белые, а затем запущена модель на «обесцвеченном» изображении (рис. 4).

**4. Вторая версия модели.** Ключевое отличие архитектуры второй версии модели [7] в том, что кодировщик и декодировщик

связаны через классификатор *InceptionResNetV2* (рис. 2).

**4.1. Улучшение архитектуры.** Классификатор обучен на датасете *ImageNet* [8], состоящем из более миллиона изображений. На вход классификатору подается изображение размером $299 \times 299$ пикселей, а на выход мы получаем вектор размером 1000, в котором значение $i$-го элемента обозначает вероятность принадлежности классу $i$. Данный подход, когда уже обученная модель используется в аналогичной задаче, называется *transfer learning*.

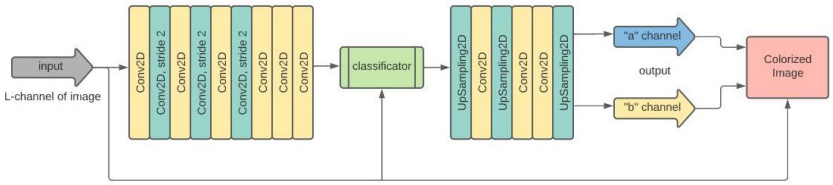


**Рис. 2.** Архитектура второй версии нейронной сети

Классификатор позволяет модели распознавать, что изображено на картинке, и сопоставлять представление объекта со схемой раскрашивания. Это означает, что модель будет «аккуратнее» раскрашивать изображения, т. е. цельные объекты должны быть раскрашены более равномерно, чем в предыдущей версии.

**4.2. Обучение.** Для обучения этой модели был расширен датасет, в итоге набор обучающих данных содержит 4000 изображений разных классов. Из-за увеличения датасета, google colab не справлялся с тем, чтобы загрузить все данные в память. Также в модель был добавлен классификатор, а это очень сильно нагружает вычислительные затраты на обучение. В итоге было принято решение обучать модель на стационарном компьютере с видеокартой TESLA P100. Одна эпоха занимала около 3-х минут, но видеокарта сильно грелась, поэтому каждую 5-ю эпоху приходилось делать остановку на 5 минут, чтобы карта не перегревалась. В итоге обучение длилось около 10 часов, accuracy на тестовом наборе изображений получилась равной 0,56. Качество стало немного хуже, чем в прошлый раз, но это закономерно, ведь модель обучалась на большем количестве данных, причем новый датасет получился более обобщенным, чем предыдущий.

**4.3. Метод скользящего окна.** В обновленной модели появилась новая проблема. Дело в том, что модель обучена на изображе-

ниях размером 256 × 256 пикселей. Прошлая версия модели могла раскрашивать изображения произвольного размера, но в текущей версии это не работает, из -за наличия классификатора, т. е. раскрашивание возможно только для изображений размером 256 × 256.

Первой идеей было разбить изображение на квадраты 256 × 256, раскрасить их отдельно, и потом снова «склеить». Если изображение не квадратное, можно просто дополнять его черными линиями справа и снизу до квадратного так, чтобы стороны делились на 256. Но у такого метода есть один большой недостаток — так как части изображения раскрашиваются независимо друг от друга, стыки частей получаются очень заметными (рис. 3а).

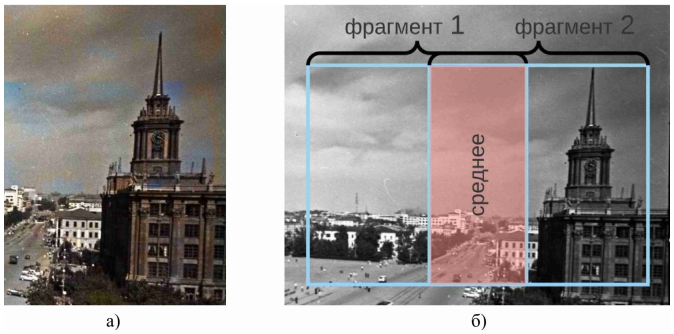


**Рис. 3.** а) часть изображения, состоящее из раскрашенных «склееных» квадратов; б) схема работы метода скользящего окна

Для решения этой проблемы был применен метод скользящего окна (рис. 3б). Идея состоит в том, чтобы делить изображение на квадратные блоки, которые будут пересекаться, раскрашивать их отдельно, а в пересечении этих квадратов брать средний цвет. Чем меньше шаг (гиперпараметр скользящего окна), с которым будет двигаться это окно, тем менее заметны будут линии стыков.

**4.4. Результаты и сравнение с предыдущей версией.** Визуальное сравнение результатов работы первой и второй модели можно увидеть на рис. 4. Видно, что вторая версия модели стала раскрашивать фотографии более аккуратно, хотя определенные проблемы, такие как тусклость и скудность цветовой палитры остались. На рис. 5 представлено детальное сравнение на черно-белой фотографии, у которой нет настоящих цветных аналогов.


**Рис. 4.** Сравнение результатов работы


**Рис. 5.** Детальное сравнение результатов

**5. Заключение.** В результате работы было собрано несколько версий датасета, которые могут быть использованы в дальнейшем. Также было рассмотрено две архитектуры сети для раскрашивания черно-белых изображений и реализованы оба подхода [9]. Модель неплохо справляется с общими планам улиц, зданий и другими изображениями городской среды. С фотографиями людей и интерьеров все сложнее. В дальнейшем планируется разнообразить набор данных, чтобы модель могла раскрашивать фотографии разных классов.

## Литература

1. Zhang R., Isola P., Efros A. A. Colorful image colorization // European Conference on Computer Vision (ECCV). 2016. P. 649–666.

2. Very Deep Convolutional Networks for Large-Scale Image Recognition [Электронный ресурс]: URL:https://arxiv.org/abs/1409.1556 (дата обращения: 05.10.2021).

3. Дрокин И. С. об одном алгоритме последовательной инициализации весов глубоких нейронных сетей и обучении ансамбля нейронных сетей // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2016. Т. 12. № 4. С. 66–74.

4. Bhatia Y., Bajpayee A., Raghuvanshi D., Mittal H. Image captioning using Google's inception-resnet-v2 and recurrent neural network // Twelfth International Conference on Contemporary Computing (IC3). 2019. P. 1–6.

5. Yandex Grabber – утилита для скачивания изображений [Электронный ресурс]: URL:https://ufahameleon.ru/soft.aspx?id=2 (дата обращения: 15.10.2021).

6. Chen Y., Luo Y., Ding Y., Yu B. Automatic colorization of images from chinese black and white films based on CNN // International Conference on Audio, Language and Image Processing (ICALIP). 2018. P. 97–102.

7. Shankar S. R., Mahesh G., Murthy K. V., Ravibabu D. A Novel approach for gray scale image colorization using convolutional neural networks // International Conference on System, Computation, Automation and Networking (ICSCAN). 2020. P. 1–8.

8. Image Net dataset [Электронный ресурс]: URL:http://www.image-net.org/ (дата обращения: 01.11.2021).

9. Реализация моделей и веса [Электронный ресурс]: URL: https://github.com/IlyaKuprik/ColorizeImage (дата обращения: 20.03.2022).