Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (05.13.11)

УДК 004.032.26
DOI: 10.24160/1993-6982-2022-1-120-129

Колоризация эскизов на основе генеративно-состязательных нейронных сетей

О.В. Бартеньев, Э.Р. Салахутдинов

Автоматическую колоризацию эскизов персонажей успешно выполняют генеративно-состязательные нейронные сети (GAN). Совершенствование подобных моделей может идти по таким направлениям, как повышение производительности и качества колоризации, снижение размера модели.
Предприняты шаги по повышению качества колоризации. На основании обзора известных решений создана и обучена начальная модель GAN, в кодере и декодере генератора которой по восемь блоков. Вторая модель GAN получена на основе первой в результате включения в блоки кодера генератора остаточных блоков и одновременного использования в декодере генератора блоков внимания и остаточных блоков. Третья модель GAN спроектирована на основе второй: в кодер и декодер генератора добавлено по одному блоку. Все модели, начальная и модифицированные, обучены на одном и том же наборе данных. Модели обучены либо с предоставлением дополнительной информации о цвете изображения (цветовой палитре эталонного изображения или цветовых метках-подсказках), либо без нее. Обученные модели оценены по качеству генерируемых ими изображений (раскрашенных эскизов), определяемому по метрике Frechet Inception Distance. Все модифицированные модели GAN генерируют более качественные изображения, чем начальная модель.

Ключевые слова: колоризация эскизов, генеративно-состязательные нейронные сети, глубокое обучение.

Для цитирования: Бартеньев О.В., Салахутдинов Э.Р. Колоризация эскизов на основе генеративно-состязательных нейронных сетей // Вестник МЭИ. 2022. № 1. С. 120—129. DOI: 10.24160/1993-6982-2022-1-120-129.

Sketch Colorization Based on Generative-Adversarial Neural Networks

O.V. Bartenyev, E.R. Salakhutdinov

Generative adversarial neural networks (GAN) successfully perform automatic colorization of character sketches. Such models can be further improved in such aspects as increasing the throughput, improving the colorization quality, and reducing the model size. Steps aimed at improving the colorization quality are taken. Known solutions are reviewed, and an initial GAN model with eight blocks in the generator encoder and decoder is developed and trained based on the review results. The second GAN model is obtained on the basis of the first one by including residual blocks in the generator encoder blocks with simultaneously using the attention blocks and residual blocks in the generator decoder. The third GAN model has been developed based on the second one: one block is added to the generator encoder and decoder. All models, including the initial and modified ones, have been trained on the same data set. The models have been trained either with or without using additional information about the image color (the color palette of the reference image or color hint labels). The trained models are evaluated with respect to the quality of the images they generate (colored sketches), determined by the Frechet Inception Distance metric. All modified GAN models generate images with quality superior to that of the initial model.

Key words: sketch colorization, generative adversarial neural networks, deep learning.

For citation: Bartenyev O.V., Salakhutdinov E.R. Sketch Colorization Based on Generative-Adversarial Neural Networks. Bulletin of MPEI. 2022;1:120—129. (in Russian). DOI: 10.24160/1993-6982-2022-1-120-129.

Введение

Задача преобразования черно-белого изображения в цветное, или его колоризация, встречается во многих сферах деятельности, например, при создании новых и реставрации старых кинофильмов, разработке медицинских и геологических атласов или проектировании одежды [1, 2]. Нередко изображение создается в два этапа. На первом рисуется эскиз, на втором — выполняется его колоризация. Число эскизов, подлежащих колоризации, весьма значительно. Так, для 10-минутного мультипликационного фильма требуется 15 — 18 тыс. цветных рисунков (из расчета 25 — 30 кадров в секунду). Автоматизация колоризации эскизов ведет к снижению стоимости мультфильма и времени его создания. Результат, разумеется, должен отвечать замыслу художника.

Дизайнеры (художники) нередко колоризируют эскизы при помощи специализированных программных средств, например, Adobe Photoshop. Данный способ, хотя и содержит некоторые элементы автоматизации, но все же весьма трудоемок и предполагает высокий уровень освоения используемого программного средства. В последние годы появляются решения, основанные на нейросетевом моделировании, позволяющие частично или полностью автоматизировать процесс колоризации эскизов. В качестве нейронных сетей (НС), выполняющих колоризацию, используются модели на основе архитектуры кодер–декодер, причем более качественно эту работу выполняют генеративно-состязательные НС (GAN — Generative Adversarial Networks).

Важной характеристикой выполняющей колоризацию НС является качество получаемого на ее выходе изображения. Цель настоящей работы — повышение качества колоризации, осуществляемой НС на основе GAN. Она достигается за счет одновременного использования в генераторе изображений блоков внимания и остаточных блоков, а также предоставления генератору дополнительной информации о цветовых характеристиках изображения обучающего множества.

Программирование ведется на Python с привлечением библиотеки PyTorch.

Используемые обозначения и сокращения

При описании НС использованы следующие обозначения слоев и сокращения:

BN — BatchNormalization (слой пакетной нормализации);

Conv — Conv2D (сверточный слой);

ConvT — Conv2DTranspose (слой транспонированной свертки);

DR — Dropout (слой прореживания);

LReLU — LeakyReLU (функция активации);

GN — GroupNormalization (слой групповой нормализации);

MP — MaxPooling2D (слой подвыборки по максимуму).

НС — нейронная сеть;

БВ — блок внимания;

ОБ — остаточный блок;

ФП — функция потерь;

GAN — генеративно-состязательная НС (Generative Adversarial Networks).

Аналогичные работы

Во всех цитируемых публикациях в НС для колоризации используются GAN. Модели отличаются архитектурой и функциями потерь. Все генераторы выполнены на основе модели кодер–декодер. Автоматически раскрашенные изображения сравниваются с эталонными (взятыми из набора данных) с помощью различных метрик: Frechet Inception Distance (*FID*), Mean Opinion Score (*MOS*), Peak Signal-To-Noise Ratio (*PSNR*), Structural Similarity Index (*SSIM*), Feature Similarity Index (*FSIM* и *FSIMc*), Light-Sensitivity (*LS*) и Color Coded Local Binary Patterns (*CCLBP*).

В [3] генератор (U-net [4]) — НС, содержащая кодер–декодер. Архитектура U-net — многоуровневая, на каждом уровне объединяются соответствующие блоки кодера и декодера (рис. 1).

Кодер понижает разрешение карты признаков с целью выделения значимых признаков входного изображения, декодер, наоборот, повышает разрешение карты

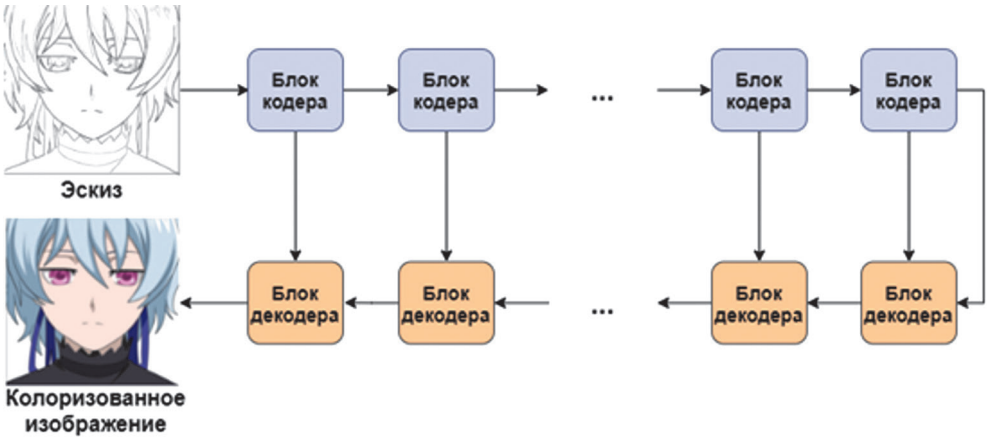

Рис. 1. Схема функционирования генератора U-net

признаков и, в конечном итоге, воспроизводит по заданному эскизу колоризованное изображение.

В [3] и кодер, и декодер генератора содержат по 8 блоков. Слои блока кодера: Conv-BN-ReLU. Слои блока декодера: ConvT-BN-ReLU. Дискриминатор состоит из шести слоев Conv. Потери генератора — взвешенная сумма четырех функций потерь (ФП): Pixel Level Loss, Feature Loss, Total Variation и Binary Cross Entropy. ФП дискриминатора — Binary Cross Entropy. Оценка качества сгенерированных изображений выполняется экспертами по шкале «нравится» — «не нравится».

В [5] генератор имеет архитектуру U-net. Кодер состоит из пяти блоков со слоями Conv-Poolling в каждом. Декодер использует 4 блока ResNeXt [6] и блок с двумя слоями Conv. ResNeXt формируется из остаточных блоков (ОБ), которые, согласно [7], предотвращают рост потерь, наблюдаемый при увеличении глубины НС. На вход первого блока кодера подается эскиз, а на вход третьего — маска с цветовыми метками-подсказками. Дискриминатор состоит из трех слоев Conv и шести блоков ResNetXt. В генераторе и дискриминаторе потери вычисляются как взвешенная сумма двух ФП: Content и Adversarial — в генераторе и Wasserstein и Penalty — в дискриминаторе. Применяемые метрики: *FID* (57.06) и *MOS* (3.186). *MOS* — средняя оценка качества сгенерированных изображений, проставляемая экспертами по 5-балльной шкале.

Особенностью НС, описанной в [8], является применение ОБ со слоем Swish (SGB — Swish-gated Residual Block, рис. 2).

С помощью SGB на основе U-net в [8] создается 12-блочный генератор (по 6 блоков в кодере и декодере). В дискриминаторе 7 слоев: Conv-Conv-Conv-Conv-MP-Conv-MP. Потери на выходе генератора — взвешенная сумма ФП Per-pixel и Perceptual. ФП дискриминатора — Cross Entropy. Применяемые метрики: *FID* (106.55), *PSNR* (19.127), *SSIM* (0.84499), *FSIM* (0.873481) и *FSIMc* (0.857569). Согласно [8], НС с SGB обучаются быстрее, чем НС с традиционным ОБ (см. рис. 2).

В [9] выполнена модификация НС, представленной в [8]. Помимо эскиза, поступающего на вход генератора, дополнительно генератор принимает маску с цветовыми метками. ФП генератора — аддитивная с компонентами Perceptual и Huber, ФП дискриминатора та же, что и в [6]. Оценка качества колоризации выполняется по метрике *MOS*.

В [10] генератор имеет архитектуру U-net с ОБ, и кодер, и декодер имеют по 10 слоев соответственно Conv и ConvT. При обучении на вход кодера подается эскиз, а на вход декодера — информация о цвете колоризованного эскиза (эталонного изображения). Для ее получения применена НС VGG19, предобученная для классификации изображений. Дискриминатор используемой в [10] GAN состоит из пяти слоев Conv. Взамен известных предложены и употреблены собственные ФП. Оценки качества генерируемых изображений не приводятся.

В [11] генератор обладает 16-ю блоками, а дискриминатор — 5-ю со следующей архитектурой: Conv-BN-ReLU-Pooling. В дискриминаторе в качестве ФП используется Cross Entropy, а в генераторе — взвешенная сумма Cross Entropy и потерь, определяемых как расстояния между истинным и сгенерированным изображениями, вычисленные по метрикам L_1 и L_2. По этим метрикам отдельно вычисляют потери для цветных изображений и тех же изображений в оттенках серого цвета, что позволяет преодолевать такие проблемы, как растекание цвета по областям, отсутствие цвета и его низкую насыщенность. Оценка качества колоризации выполняется по метрикам *LS* (–0.262) и *CCLBP* (2.613).

Во всех рассмотренных моделях дискриминатор создан на основе сверточных слоев, а генератор — на базе U-net с различными реализациями ОБ. Исключение составляет модель [3], спроектированная на основе U-net без ОБ.

Архитектуру U-net применяют и в других задачах машинной обработки изображений. Так, в [12] U-net обучается решать задачу сегментации — выделять контуры поджелудочной железы на изображении, полученном в результате компьютерной томографии. При этом исследуются модификации модели с блоками внимания (БВ) и без них. Эксперименты, выполненные в [12], показывают, что включение в U-net БВ в среднем повышает точность сегментации по метрике DSC на 2...3%.

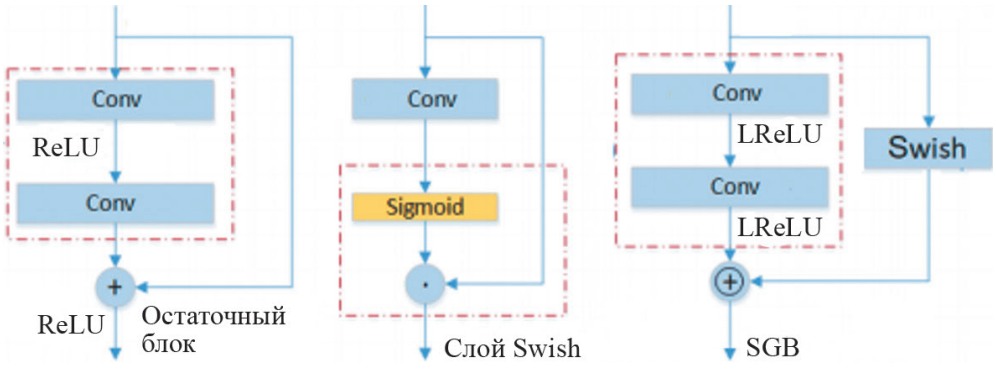

Рис. 2. Традиционный ОБ, слой Swish и SGB [8]

В [13] описан опыт одновременного употребления ОБ и БВ: добавление в модель НС обоих блоков обеспечивает повышение качества классификации изображений набора данных CIFAR-100 на 2,6% по сравнению с моделью, использующей только ОБ.

Выполненный обзор позволяет наметить следующие шаги по созданию GAN, не уступающей, а возможно, и превосходящей описанные выше аналоги по качеству колоризации:

- конструировать НС на основе U-net с использованием ОБ;
- добавить в U-net БВ, предполагая повысить за счет этого точность выделения контуров изображения (эскиза);
- снабжать НС при ее обучении дополнительной информацией о цвете изображения.

Следует отметить, что сравнение приведенных моделей по показателю «качество колоризации» практически невозможно, поскольку в рассмотренных работах, во-первых, для оценки качества берутся разные метрики и, во-вторых, обучение и тестирование проходят на разных наборах данных.

Управление цветом выходного изображения

На этапе обучения НС на вход дискриминатора подаются сгенерированное и реальное (эталонное) изображения из обучающего множества. Размер изображений — 512×512 пикселей. В программе изображения представляются в виде тензоров размера (*batch*, 3, 512, 512), где *batch* — число изображений в обучающем пакете; 3 — число компонентов цвета (используется система RGB). При этом возможны три варианта организации входных данных, поступающих на вход генератора.

Вариант 1.

На вход генератора поступает только эскиз [3, 5, 8, 14]. Преимущество заключается в полной автоматизации процесса колоризации, недостаток — в отсутствии возможности управления изображением.

Вариант 2.

Вместе с эскизом на вход генератора подается цветовая палитра из заданного числа цветов, извлекаемая из эталонного изображения [10, 15, 16]. В настоящей работе палитра включает 9 цветов и извлекается из эталонного изображения с помощью метода k-means Clustering. Каждый экземпляр палитры — монохромное изображение такого же размера, как и эталонное. На вход генератора подается тензор размера (1, 30, 512, 512) — результат конкатенации по 1-у измерению эталонного изображения и 9-и экземплярам цветовой палитры. Генератор порождает цветное изображение на основании эскиза с учетом заданной палитры.

Преимуществом такого подхода является возможность управления цветовым стилем генерируемого изображения. При генерации изображения, выполняемой обученной НС, цветовая палитра может быть задана по усмотрению дизайнера или извлечена из другого эталонного изображения. Недостаток данного варианта — невозможность локализовать область, к которой следует применить определенный цвет.

Вариант 3.

Использование цветовых меток-подсказок [5, 9, 11]. Генератор получает на вход эскиз с наложенной на него маской с цветовыми метками. Маска — белое изображение размера 512×512 с неотображаемой сеткой с заданным числом узлов (в работе — 40). В узле сетки может быть расположена метка 10×10 пикселей, цвет которой определяется цветом квадрата с такими же, как и у метки, координатами на эталонном изображении. При обучении GAN используемые узлы сетки выбираются случайным образом, после чего формируется маска с цветовыми метками-подсказками, накладываемая на эскиз. Таким образом, модель порождает цветное изображение с учетом цветовых меток.

При генерации изображения, выполняемой обученной НС, маска может быть задана уже не случайно, а по усмотрению дизайнера, что позволяет локализовать области, к которым необходимо применить определенный цвет.

Решаемые задачи

Цель работы — повышение качества колоризации эскизов, выполняемой GAN. Качество генерируемого GAN изображения оценивается по метрике Frechet Inception Distance (*FID*).

Созданы, обучены и протестированы следующие три модели GAN:

Модель 1 — аналогична модели, описанной в [3].

Модель 2.1 — модификация модели 1, в декодер генератора добавлены остаточные блоки и блоки внимания.

Модель 2.2 — модификация модели 2.1, глубина генератора на один блок больше, чем в модели 2.1.

Обучение и тестирование моделей велось с использованием всех рассмотренных вариантов управления цветом генерируемого изображения (каждая модель обучалась трижды). Все обученные 9 моделей протестированы на проверочном множестве. Результаты тестирования — усредненные оценки качества генерируемых изображений по метрике *FID*. Сравнение результатов позволило сделать вывод об эффективности выполненных модификаций.

Набор данных

В качестве источника данных для обучения и тестирования GAN взят набор данных Anime Sketch Colorization Pair [18], содержащий 17 769 изображений размером 1024×512 пикселей. Каждое изображение состоит из эскиза и цветного рисунка (рис. 3).

С целью повышения производительности каждое изображение поделено на два (эскиз и цветное изображение), размер каждой части — 512×512 пикселей. Набор данных разбит на обучающее и проверочное множества, содержащие 90 и 10% изображений.

Используемые архитектуры GAN

Создаваемые в работе GAN различаются генераторами. Дискриминаторы во всех моделях одинаковы. При описании моделей НС введены обозначения слоев с указанием их параметров. Так, Conv(x_i, 3×3) обозначает сверточный слой с числом выходных каналов x_i и размером окна свертки 3; MP(2×2) — слой подвыборки с размерами окна и шага 2; DR(0.5) — слой прореживания с коэффициентом 0,5; Linear(out) — линейный слой с размером выхода out, например, Linear(512).

На вход дискриминатора подаются эскиз и цветное изображение, которое берется либо из обучающего множества, либо с выхода генератора. Задача дискриминатора — классифицировать входную пару изображений как истинную или ложную. Пара изображений помечается True, если цветное изображение пришло из обучающего множества, или False — в противном случае. Во всех моделях дискриминатор — сверточная НС, состоящая из шести блоков. Каждый содержит следующие слои: Conv(x_i, 3×3)-BN-ReLU-MP(2×2)-DR(0.5)-Linear(512)-BN-DR(0.5)-Linear(1), x_i = (64, 128, 256, 512, 512, 512), где i — номер блока.

В каждой создаваемой GAN генератор имеет архитектуру U-net. На вход генератора подается эскиз, на выходе генератора порождается RGB-изображение. Размеры входного и выходного изображений одинаковы — 512×512 пикселей. Созданные GAN обучаются с использованием всех вариантов управления цветом изображения.

В модели 1 (начальной) генератор, построенный по аналогии с [3], включает кодер и декодер из восьми блоков каждый.

Составы i-го блоков кодера и декодера:

Conv(x_i, 3×3)-BN-LReLU, x_i = (64, 128, 256, 512, 512, 512, 512, 512).

ConvT(x_i, 3×3)-BN-DR(0.5)-LReLU, x_i = (512, 512, 512, 512, 256, 128, 64, 3).

В моделях 2.1 и 2.2 генератор конструируется с применением ОБ в кодере и ББ с ОБ в декодере. В модели 2.1 и кодер, и декодер генератора имеют по восемь блоков, в модели 2.2 — по девять. Блок кодера — ОБ, архитектура которого показана на рис. 4.

Число выходных каналов в сверточных слоях i-го ОБ в модели 2.1: x_i = (64, 128, 256, 512, 512, 512, 512, 512), в модели 2.2 добавляется ОБ с Conv(512, 3×3). Данные, поступающие на узел ⊕, суммируются.

В моделях 2.1 и 2.2, следуя рекомендациям, приведенным в [19], вместо слоя пакетной нормализации BN, употребленного в модели 1, взят слой групповой нормализации GN.

Блок декодера содержит и ОБ, и ББ (рис. 5).

Архитектура ОБ декодера такая же, как и ОБ кодера (см. рис. 4), архитектура ББ заимствована из [12] (рис. 6).

Число выходных каналов в сверточных слоях ОБ и ББ декодера в моделях 2.1 и 2.2: x_i = (512, 512, 512, 512, 256, 128, 64, 3); x_i = (512, 512, 512, 512, 512, 256, 128, 64, 3).

Вычисление потерь

Корректировка весов созданных моделей НС проходит в процессе их обучения в результате обратного распространения ошибки, вычисляемой как взвешенная сумма четырех функцией потерь [3]:

$$Loss = c_1 BCE(x, y) + \\ + c_2 PLL(x, y) + c_3 FLL(x, y) + c_4 TVL(y),$$

Рис. 3. Пример изображения из набора данных

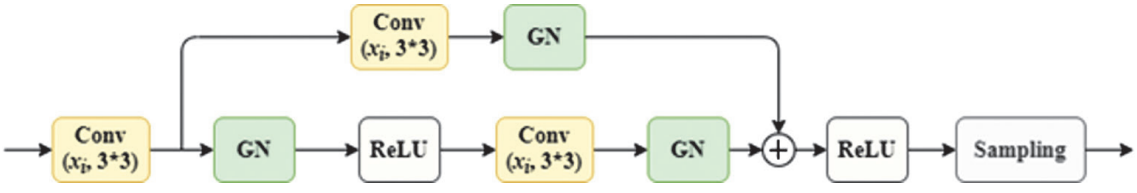

Рис. 4. Архитектура остаточного блока

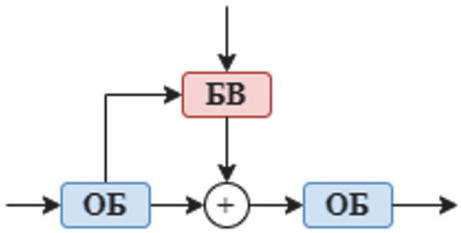

Рис. 5. Архитектура блока декодера

где c_i — весовые коэффициенты, $c_1 = 1$; $c_2 = 100$; $c_3 = 0{,}01$; $c_4 = 0{,}0001$ [3]; x, y — истинное (эталонное) и сгенерированное изображения; компоненты *Loss* — функции, оценивающие различие между x и y (*BCE*, *PLL* и *FLL*) и непрерывность цветовой гаммы y (*TVL*). В обучаемых НС x и y — тензоры размера (*batch*, 3, 512, 512), где *batch* — число изображений в обучающем пакете, 3 — число компонентов цвета, 512 — высота (ширина) изображения в пикселях.

Примененные функции потерь:

BCE — бинарная перекрестная энтропия (Binary Cross Entropy), стандартная функция библиотеки PyTorch; *PLL* — усредненная попиксельная разница между x и y (Pixel Level Loss), $PLL = mean(abs(x - y))$; *TVL* — оценка непрерывности цветовой гаммы (Total Variation Loss) сгенерированного изображения,

$$TVL = sqrt\left(sum\left(square\left(y[:, 1:, :, :] - y[:, :-1, :, :]\right)\right) + \right.$$
$$\left. + sum\left(square\left(y[:, :, 1:, :] - y[:, :, :-1, :]\right)\right)\right).$$

Первое слагаемое *TVL* — оценка непрерывности цветовой гаммы по вертикали, второе — по горизонтали изображения. Вычисление первого слагаемого *TVL* проиллюстрируем следующим примером (для наглядности взят тензор с двумя измерениями):

```
import torch
x = torch.rand(4, 4) # Создаем тензор размера (4, 4)
# tensor([[0.2581, 0.3784, 0.3709, 0.7966],
#         [0.4987, 0.1052, 0.4198, 0.4515],
#         [0.1869, 0.0538, 0.1612, 0.0352],
#         [0.9333, 0.7381, 0.6737, 0.9039]])
x1 = x[1:, :] # Тензор без первой строки
# tensor([[0.4987, 0.1052, 0.4198, 0.4515],
#         [0.1869, 0.0538, 0.1612, 0.0352],
#         [0.9333, 0.7381, 0.6737, 0.9039]])
x2 = x[:-1, :] # Тензор без последней строки
# tensor([[0.2581, 0.3784, 0.3709, 0.7966],
#         [0.4987, 0.1052, 0.4198, 0.4515],
#         [0.1869, 0.0538, 0.1612, 0.0352]])
tvl_1 = torch.sqrt(torch.sum(torch.square(x1 – x2)))
print(tvl_1) # tensor(1.6238)
```

FLL — оценка различия между выделенными признаками изображений x и y:

$$FLL = mean(sqrt(sum(square(fx - fy)))),$$

где fx, fy — признаки изображений x и y, выделенные НС VGG16, предобученной для классификации изображений (доступ к VGG16 обеспечивает PyTorch).

Использованные для вычисления потерь функции *abs*, *mean*, *sqrt*, *square* и *sum* возвращают абсолютную величину, среднее значение, квадратный корень, квадрат и сумму компонентов своего аргумента. В случае *BCE*, *PLL* и *TVL* аргументом является тензор размера (*batch*, 3, 512, 512), в случае *FLL* — тензор размера (*batch*, 3, 224, 224).

Примеры генерируемых изображений

Приведем примеры изображений, сгенерированных моделями 1 и 2.1, обученными с применением различных вариантов управления цветом изображения.

На рисунке 7 даны эскиз и изображение, созданное генератором модели 1 (использован первый вариант управления цветом).

На рисунке 8 представлен результат работы генератора модели 2.1, обученного по второму варианту управления цветом. При генерации использована цветовая палитра, извлеченная из левого изображения. Слева направо следующие изображения: источник цветовой палитры, эскиз и сгенерированное.

На рисунке 9 изображены эскиз с метками и результат работы генератора модели 2.1, обученного по варианту 3 управления цветом порождаемого изображения. Метки-подсказки расставлены на эскизе дизайнером.

Оценка качества генерируемых изображений

Оценка качества генерируемых изображений выполнена по метрике Frechet Inception Distance (*FID*) [20], характеризующей различие (расстояние) между двумя изображениями.

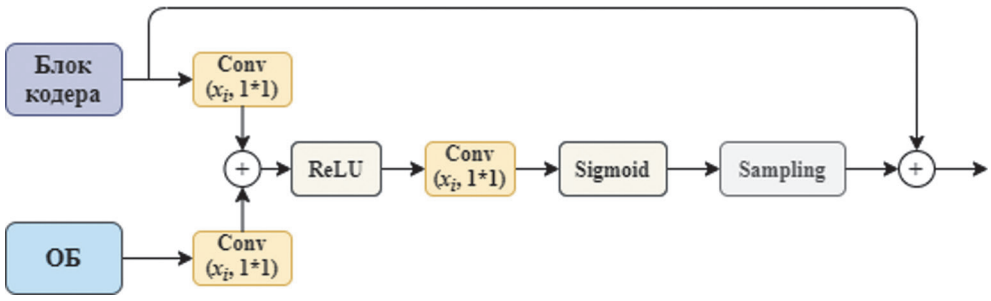

Рис. 6. Архитектура блока внимания

Рис. 7. Колоризация, выполненная генератором модели 1

Рис. 8. Перенос цветового стиля эталонного изображения

Рис. 9. Колоризация с заданием меток-подсказок

При ее вычислении между изображениями x и y изображение рассматривается как двумерное гауссовское распределение со средним μ и матрицей ковариации Σ:

$$FID = \left\|\mu_x - \mu_y\right\|_2^2 + tr\left(\Sigma_x + \Sigma_y - 2\left(\Sigma_x \Sigma_y\right)^{\frac{1}{2}}\right),$$

где tr — сумма диагональных элементов результата.

Все модели GAN обучены трижды: использованы все варианты управления цветом изображения. Обучение велось на 100 эпохах с применением оптимизатора Adam. При тестировании обученной модели взяты эскиз и соответствующее ему цветное (эталонное) изображение проверочного множества (см. рис. 3). Эскиз подается на вход генератора. Модель генерирует изображение (раскрашенный эскиз), после чего вычисляется *FID* между созданным генератором и эталонным изображениями. Процедура выполняется для всех изображений проверочного множества. Полученные значения *FID* усредняются (таблица).

Оценки качества колоризации по метрике FID

Модель	Варианты управления цветом		
	1	2	3
1	112,63	103,58	99,09
2.1	98,29	89,20	83,44
2.2	96,41	84,28	79,86
Рост качества колоризации			
$FID_1/FID_{2.1}$	1,146	1,161	1,188
$FID_1/FID_{2.2}$	1,168	1,229	1,241
$FID_{2.1}/FID_{2.2}$	1,020	1,058	1,045

Сопоставление результатов тестирования моделей позволяет сделать вывод о продуктивности идеи построения генератора на основе U-net с применением ОБ в кодере и БВ с ОБ в декодере. Увеличение глубины генератора на один блок также дает рост качества колоризации, правда незначительный. Визуально изображения, сгенерированные моделями 2.1 и 2.2 по одному и тому же эскизу, малоразличимы.

Заключение

Анализ задачи колоризации эскизов и методов ее решения, а также методов решения иных задач машинной обработки изображений (классификации и сегментации) позволил выбрать для построения генератора GAN архитектуру U-net и наметить (а затем предпринять) шаги по улучшению качества генерируемых изображений, связанные с модификацией архитектуры генератора и предоставлением НС на этапе ее обучения дополнительной информации о цветовых характеристиках изображения.

В качестве начальной модели генератора (модель 1) взята U-net без ОБ и БВ. Модель 2 получена из модели 1 в результате применения в блоках кодера ОБ, а в блоках декодера и ОБ, и БВ. Модель 3 образована в результате увеличения глубины генератора модели 2 на один блок. Дополнительная информация о цвете изображения формируется либо как палитра цветов, либо в виде маски с цветовыми метками. Все модели обучены с применением трех вариантов управления цветом выходного изображения. Параметры обучения моделей одинаковы. Сравнение качества изображений (см. табл), генерируемых моделями, позволяет сделать вывод о продуктивности принятых решений, что, в свою очередь, помогает наметить шаги по повышению качества результата:

- использование БВ не только в декодере, но и в кодере;
- применение смешанного варианта управления цветом выходного изображения — одновременного употребления на этапе обучения цветовой палитры и цветовых меток в качестве дополнительных сведений о цвете эталонного изображения.

GAN, способные раскрашивать эскизы, можно включить в процесс автоматизации изготовления разного рода продукции, например, автоматического создания мультипликационного фильма на основе сценария и библиотеки эскизов. Также подобные НС способны быть частью автоматического художника, генерирующего на первом этапе эскизы, колоризируемые затем одной из представленных GAN.

Литература

1. **Анатомический** атлас человека в картинках [Электрон. ресурс] www.medikweb.ru/anatom_atlas/ (дата обращения 01.03.2021).

2. **Геологические** карты и атласы [Электрон. ресурс] www.jurassic.ru/maps.html (дата обращения 01.03.2021).

3. **Yifan L., Zengchang Q., Zhenbo L., Hua W.** Autopainter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks [Электрон. ресурс] www.arxiv.org/pdf/1705.01908.pdf (дата обращения 01.03.2021).

4. **Ronneberger O., Fischer P., Brox T.** U-Net: Convolutional Networks for Biomedical Image Segmentation. [Электрон. ресурс] www.arxiv.org/pdf/1505.04597.pdf (дата обращения 01.03.2021).

References

1. **Anatomicheskiy** Atlas Cheloveka v Kartinkakh [Elektron. Resurs] www.medikweb.ru/anatom_atlas/ (Data Obrashcheniya 01.03.2021). (in Russian).

2. **Geologicheskie** Karty i Atlasy [Elektron. Resurs] www.jurassic.ru/maps.html (Data Obrashcheniya 01.03.2021). (in Russian).

3. **Yifan L., Zengchang Q., Zhenbo L., Hua W.** Autopainter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks [Elektron. Resurs] www.arxiv.org/pdf/1705.01908.pdf (Data Obrashcheniya 01.03.2021).

4. **Ronneberger O., Fischer P., Brox T.** U-Net: Convolutional Networks for Biomedical Image Segmentation. [Elektron. Resurs] www.arxiv.org/pdf/1505.04597.pdf (Data Obrashcheniya 01.03.2021).

5. **Yuanzheng C., Xinzhu M.** User-guided Deep Anime Line Art Colorization with Conditional Adversarial Networks [Электрон. ресурс] www.arxiv.org/pdf/1808.03240.pdf (дата обращения 01.03.2021).

6. **Xie S. e. a.** Aggregated Residual Transformations for Deep Neural Networks [Электрон. ресурс] www.arxiv.org/pdf/1611.05431.pdf (дата обращения 01.03.2021).

7. **Kaiming H., Xiangyu Z., Shaoqing R.** Deep Residual Learning for Image Recognition. [Электрон. ресурс] www.arxiv.org/pdf/1512.03385.pdf (дата обращения 01.03.2021).

8. **Gang L., Xin C., Yanzhong H.** Anime Sketch Coloring with Swish-gated Residual U-net and Spectrally Normalized GAN // Eng. Letters. 2019. V. 27. Iss. 3. Pp. 1—7.

9. **Ye R. e. a.** Interactive Anime Sketch Colorization with Style Consistency via a Deep Residual Neural Network // Proc. Intern. Conf. Technologies and Appl. Artificial Intelligence. Kaohsiung, 2019. Pp. 1—5.

10. **Lvmin Z., Yi J., Xin L.** Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN [Электрон. ресурс] www.arxiv.org/abs/1706.03319 (дата обращения 01.03.2021).

11. **Lvmin Z. e. a.** Two-stage Sketch Colorization [Электрон. ресурс] www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8944253 (дата обращения 01.03.2021).

12. **Ozan Oktay e. a.** Attention U-Net: Learning Where to Look for the Pancreas [Электрон. ресурс] www.arxiv.org/pdf/1804.03999.pdf (дата обращения 01.03.2021).

13. **Fei W. e. a.** Residual Attention Network for Image Classification [Электрон. ресурс] www.arxiv.org/pdf/1704.06904.pdf (дата обращения 01.03.2021).

14. **Masashi A., Yuichi S., Yasuyuki T.** Do You Like Sclera? Sclera-region Detection and Colorization for Anime Character Line Drawings // Intern. J. Networked and Distributed Computing. 2019. V. 7(3). Pp. 113—120.

15. **Kataoka Y., Matsubara T., Uehara K.** Automatic Manga Colorization with Color Style by Generative Adversarial Nets // Proc. XVIII IEEE/ACIS Intern. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Kanazawa, 2017. Pp. 495—499.

16. **Junsoo L., Eungyeup K., Yunsung L.** Reference-based Sketch Image Colorization using Augmented-self Reference and Dense Semantic Correspondence www.arxiv.org/pdf/2005.05207 (дата обращения 01.03.2021).

17. **Ren H., Li J., Gao N.** Automatic Sketch Colorization with Tandem Conditional Adversarial Networks // Proc. XI Intern. Symp. Computational Intelligence and Design. Hangzhou, 2018. Pp. 11—15.

18. **Taebum K.** Anime Sketch Colorization Pair [Электрон. ресурс] www.kaggle.com/ktaebum/anime-sketch-colorization-pair (дата обращения 01.03.2021).

19. **Yuxin W., Kaiming H.** Group Normlization [Электрон. ресурс] www.arxiv.org/pdf/1803.08494.pdf (дата обращения 01.03.2021).

20. **Heusel M., Ramsauer H., Unterthiner T., Nessler B., Hochreiter S.** GANs Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium [Электрон. ресурс] www.arxiv.org/pdf/1706.08500.pdf (дата обращения 01.03.2021).

5. **Yuanzheng C., Xinzhu M.** User-guided Deep Anime Line Art Colorization with Conditional Adversarial Networks [Elektron. Resurs] www.arxiv.org/pdf/1808.03240.pdf (Data Obrashcheniya 01.03.2021).

6. **Xie S. e. a.** Aggregated Residual Transformations for Deep Neural Networks [Elektron. Resurs] www.arxiv.org/pdf/1611.05431.pdf (Data Obrashcheniya 01.03.2021).

7. **Kaiming H., Xiangyu Z., Shaoqing R.** Deep Residual Learning for Image Recognition. [Elektron. Resurs] www.arxiv.org/pdf/1512.03385.pdf (Data Obrashcheniya 01.03.2021).

8. **Gang L., Xin C., Yanzhong H.** Anime Sketch Coloring with Swish-gated Residual U-net and Spectrally Normalized GAN. Eng. Letters. 2019;27;3:1—7.

9. **Ye R. e. a.** Interactive Anime Sketch Colorization with Style Consistency via a Deep Residual Neural Network. Proc. Intern. Conf. Technologies and Appl. Artificial Intelligence. Kaohsiung, 2019:1—5.

10. **Lvmin Z., Yi J., Xin L.** Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN [Elektron. Resurs] www.arxiv.org/abs/1706.03319 (Data Obrashcheniya 01.03.2021).

11. **Lvmin Z. e. a.** Two-stage Sketch Colorization [Elektron. Resurs] www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8944253 (Data Obrashcheniya 01.03.2021).

12. **Ozan Oktay e. a.** Attention U-Net: Learning Where to Look for the Pancreas [Elektron. Resurs] www.arxiv.org/pdf/1804.03999.pdf (Data Obrashcheniya 01.03.2021).

13. **Fei W. e. a.** Residual Attention Network for Image Classification [Elektron. Resurs] www.arxiv.org/pdf/1704.06904.pdf (Data Obrashcheniya 01.03.2021).

14. **Masashi A., Yuichi S., Yasuyuki T.** Do You Like Sclera? Sclera-region Detection and Colorization for Anime Character Line Drawings. Intern. J. Networked and Distributed Computing. 2019;7(3):113—120.

15. **Kataoka Y., Matsubara T., Uehara K.** Automatic Manga Colorization with Color Style by Generative Adversarial Nets. Proc. XVIII IEEE/ACIS Intern. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Kanazawa, 2017: 495—499.

16. **Junsoo L., Eungyeup K., Yunsung L.** Reference-based Sketch Image Colorization using Augmented-self Reference and Dense Semantic Correspondence www.arxiv.org/pdf/2005.05207 (Data Obrashcheniya 01.03.2021).

17. **Ren H., Li J., Gao N.** Automatic Sketch Colorization with Tandem Conditional Adversarial Networks. Proc. XI Intern. Symp. Computational Intelligence and Design. Hangzhou, 2018:11—15.

18. **Taebum K.** Anime Sketch Colorization Pair [Elektron. Resurs] www.kaggle.com/ktaebum/anime-sketch-colorization-pair (Data Obrashcheniya 01.03.2021).

19. **Yuxin W., Kaiming H.** Group Normlization [Elektron. Resurs] www.arxiv.org/pdf/1803.08494.pdf (Data Obrashcheniya 01.03.2021).

20. **Heusel M., Ramsauer H., Unterthiner T., Nessler B., Hochreiter S.** GANs Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium [Elektron. Resurs] www.arxiv.org/pdf/1706.08500.pdf (Data Obrashcheniya 01.03.2021).

Сведения об авторах:

Бартеньев Олег Васильевич — кандидат технических наук, доцент кафедры прикладной математики и искусственного интеллекта НИУ «МЭИ», e-mail: mdf4@mail.ru
Салахутдинов Эмиль Рашитович — аспирант кафедры прикладной математики и искусственного интеллекта НИУ «МЭИ», e-mail: SalakhutdinovER@gmail.com

Information about authors:

Bartenyev Oleg V. — Ph.D. (Techn.), Assistant Professor of Applied Mathematics and Artificial Intelligence Dept., NRU MPEI, e-mail: mdf4@mail.ru
Salakhutdinov Emil R. — Ph.D-student of Applied Mathematics and Artificial Intelligence Dept., NRU MPEI, e-mail: SalakhutdinovER@gmail.com

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов
Conflict of interests: the authors declare no conflict of interest

Статья поступила в редакцию: *10.05.2021*
The article received to the editor: *10.05.2021*