УДК 004.89

ИСПОЛЬЗОВАНИЕ СВЕРТОЧНЫХ ВЕЙВЛЕТ-БЛОКОВ В ЗАДАЧЕ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

В. А. ВОРОБЕЙ[1)]***, А. Э. МАЛЕВИЧ***[1)]

[1)]*Белорусский государственный университет, пр. Независимости, 4, 220030, г. Минск, Беларусь*

Аннотация. На примере задачи классификации изображений и вейвлет-семейства CDF-9/7 показано, как можно внедрить дискретное вейвлет-преобразование в модель компьютерного зрения, сохранив возможность ее обучения методом обратного распространения ошибки. Предложен и успешно встроен в ряд моделей нейронных сетей сверточный вейвлет-блок, который сочетает в себе обработку признаков входного сигнала на нескольких уровнях вейвлет-разложения и позволяет уменьшить исходный размер модели на 30–40 %, обеспечивая при этом сопоставимое качество. Продемонстрирована возможность эффективно выполнять дискретное вейвлет-преобразование на графическом процессоре при использовании лифтинг-схемы. Реализация вейвлет-преобразования построена на поэлементных операциях сложения и умножения, что позволяет при необходимости экспортировать обученную модель в требуемый формат для запуска на новых данных без дополнительных сложностей. В качестве базовых моделей использованы архитектуры ResNetV2-50, MobileNetV2 и EfficientNetV2-B0. Для проведения экспериментов подготовлен набор данных на основе подвыборки категорий датасета LSUN.

Ключевые слова: нейронные сети; глубокое обучение; вейвлеты; дискретное вейвлет-преобразование; классификация изображений.

CONVOLUTIONAL WAVELET BLOCKS IN IMAGE CLASSIFICATION

U. A. VARABEI[a]***, A. E. MALEVICH***[a]

[a]*Belarusian State University, 4 Niezaliezhnasci Avenue, Minsk 220030, Belarus*
Corresponding author: U. A. Varabei (v.vorobey.edu@gmail.com)

Abstract. In this paper, based on an image classification problem and wavelet family CDF-9/7, it is shown how to incorporate discrete wavelet transform into a computer vision model, while maintaining the ability of its training with the backpropagation method. A convolutional wavelet block, that extracts features at different levels of decomposition of the incoming signal, is proposed and successfully integrated into a set of neural network models. The blocks implemented allow to reduce the original model size by 30–40 %, while maintaining comparable quality in terms of metric. An effective method for evaluation of discrete wavelet transform on graphics processing unit with lifting scheme is presented. The implementation of wavelet blocks uses element-wise operations of additions and multiplications, thus allowing a simple export of

Образец цитирования:
Воробей ВА, Малевич АЭ. Использование сверточных вейвлет-блоков в задаче классификации изображений. *Журнал Белорусского государственного университета. Математика. Информатика.* 2024;2:93–103.
EDN: LSIEVN

For citation:
Varabei UA, Malevich AE. Convolutional wavelet blocks in image classification. *Journal of the Belarusian State University. Mathematics and Informatics.* 2024;2:93–103. Russian.
EDN: LSIEVN

Авторы:
Владислав Александрович Воробей – аспирант кафедры дифференциальных уравнений и системного анализа механико-математического факультета. Научный руководитель – А. Э. Малевич.
Александр Эрнестович Малевич – кандидат физико-математических наук, доцент; доцент кафедры дифференциальных уравнений и системного анализа механико-математического факультета.

Authors:
Uladzislau A. Varabei, postgraduate student at the department of differential equations and system analysis, faculty of mechanics and mathematics.
v.vorobey.edu@gmail.com
Alexander E. Malevich, PhD (physics and mathematics), docent; associate professor at the department of differential equations and system analysis, faculty of mechanics and mathematics.
malevich@bsu.by

a trained model into one of desired formats for running on new data. ResNetV2-50, MobileNetV2 and EfficientNetV2-B0 architectures are used as the basis models. A new dataset, which is based on a set of categories of LSUN dataset, is constructed for conducting experiments.

Keywords: neural networks; deep learning; wavelets; discrete wavelet transform; image classification.

Введение

Задача классификации изображений остается одной из наиболее востребованных в компьютерном зрении. Задачи данного класса хорошо решаются с помощью моделей нейронных сетей, таких как VGG [1], ResNet [2], EfficientNet [3] и др. Как правило, для повышения качества работы моделей в них приходится увеличивать число параметров, что ведет к росту потребления памяти и создает определенные сложности как при обучении, так и при работе финальной версии модели.

Необходимость использования компактных моделей возникает при их встраивании непосредственно в оборудование (медицинские приборы, сенсоры, устройства с камерами и т. п.). Проблема с объемом памяти также актуальна для веб-сервисов, при работе с которыми модель скачивается на устройство пользователя. Еще одним примером могут быть мобильные приложения, использующие большое количество моделей (например, видеоредакторы). Во всех перечисленных случаях особенно важно уменьшить размер исходных моделей, сохранив достаточный уровень качества. Для этого можно использовать прунинг [4], обучение с типами данных пониженной точности (float8, float16) [5], а также методы полуконтролируемого обучения [6].

Часто для обучения работе с новыми данными веса моделей инициализируются случайными значениями из некоторых распределений (за исключением дообучения уже готовой модели на похожем наборе данных). В этом есть как плюсы, так и минусы. Случайная инициализация хороша тем, что модель может сама выучить параметры так, чтобы минимизировать ошибку на тренировочном наборе данных. Однако учить все параметры одновременно очень сложно, особенно когда их количество доходит до нескольких сотен тысяч или даже миллионов. В настоящей работе предлагается упростить модели задачу извлечения признаков из данных, добавив в нее вейвлеты. Они хорошо подходят для анализа данных, поскольку извлекают информацию о низких и высоких частотах входного сигнала на разных уровнях его разложения. Похожая идея упрощения обучения моделей использовалась в работе [7]. В ней авторы добавляли вейвлеты в несколько первых сверточных блоков для извлечения простейших признаков из входных изображений, это позволяло более глубоким слоям учить более сложные признаки, что приводило к увеличению значения метрик. В публикации [8] на примере задачи классификации изображений было показано, что при добавлении вейвлетов модели показывают более высокие результаты, а также лучше работают на данных, в которых присутствует много шума. В работе [9] благодаря способности вейвлетов анализировать сигнал на разных уровнях авторам удалось успешно решить задачу определения подделок человеческих лиц.

Отличительная особенность данного исследования – уменьшение размера модели с сохранением уровня качества при использовании только лишь вейвлетов. Полученную таким способом модель можно дополнительно сжимать стандартными методами, указанными выше.

Создание модели

Для проведения экспериментов были выбраны три достаточно компактные по размеру модели сверточных нейронных сетей, которые широко применяются в настоящее время, – ResNetV2-50 [10], MobileNetV2 [11] и EfficientNetV2-B0 [12]. Выбранные архитектуры за счет своей легковесности часто используются также в задачах сегментации и детекции для извлечения признаков разного уровня из изображений с дальнейшим объединением информации для решения поставленной задачи.

Дискретное вейвлет-преобразование. Различают непрерывное и дискретное вейвлет-преобразования. В данной работе был использован дискретный вариант, поскольку сигналы, передаваемые внутри классических сверточных нейронных сетей, также являются дискретными. Как правило, дискретное вейвлет-преобразование применяют к одномерным сигналам, в то время как при работе с изображениями внутри нейронных сетей сигналы имеют четырехмерную размерность: [b, c, h, w], где b – размер пакета (батча); c – количество каналов; h – высота растра; w – ширина растра. Для обобщения одномерного вейвлет-преобразования на двумерный случай поступают следующим образом: сначала выполняют преобразование построчно, а затем применяют его к каждому столбцу результата предыдущего шага. В итоге получается изображение, которое разделено на четыре части (каждая из них по ширине и высоте в 2 раза меньше исходного изображения) – LL, HL, LH, HH. Здесь L – медленно меняющаяся компонента;

H – быстро меняющаяся компонента. Схема данного метода представлена на рис. 1, где использованы следующие обозначения: X – входное изображение; DWT – дискретное вейвлет-преобразование. Далее описанный метод применяется отдельно к каждому каналу и отдельно к каждому изображению.

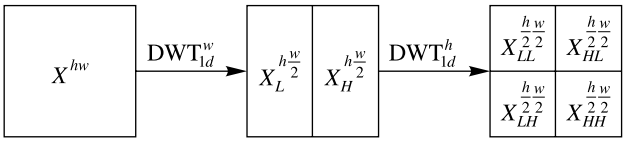

Рис. 1. Двумерное дискретное вейвлет-преобразование
Fig. 1. Two-dimensional discrete wavelet transform

Существует три метода выполнения дискретного вейвлет-преобразования:
1) полифазная матрица;
2) свертка с фильтрами синтеза и анализа;
3) лифтинг-схема.

У каждого из этих методов есть преимущества и недостатки. Алгоритм, основанный на полифазной матрице, требует перевода сигнала в полином Лорана и выполнения в дальнейшем символьных вычислений, что крайне плохо подходит для числовых пакетов. Метод, использующий свертку, достаточно просто реализуется стандартными средствами пакетов для глубокого обучения: для этого создаются фильтры, которыми можно инициализировать сверточные слои и при необходимости обучать их. Однако следует учитывать, что операция свертки даже в одномерном случае имеет нелинейную сложность относительно длины входного сигнала, а поскольку свертка выполняется по строкам и по столбцам, то исходная сложность алгоритма квадратично увеличивается. Лифтинг-схему можно реализовать с помощью векторных операций сложения, сдвигов сигнала на определенное количество позиций и умножения на число, поэтому она является весьма эффективной при небольшом количестве сложений. Данный алгоритм работает следующим образом.

Шаг 1. Сигнал делится на две части – четную (e) и нечетную (o).

Шаг 2. Четная часть сигнала модифицируется на i-м шаге предсказания (P): операцию можно представить в виде набора операций сложения, умножения на число, а также сдвигов сигнала. Далее полученный сигнал добавляется к текущей нечетной части сигнала.

Шаг 3. Нечетная часть сигнала модифицируется на i-м шаге обновления (U): операция аналогична операции на шаге 2, за исключением количества позиций сдвига, а также используемых констант в умножении. Полученный сигнал добавляется к текущей четной части сигнала.

Шаг 4. Шаги 2 и 3 повторяются несколько раз (зависит от типа вейвлет-семейства).

Шаг 5. Четная часть сигнала умножается, а нечетная часть сигнала делится на константу k.

Шаг 6. На выходе получаются две части сигнала – аппроксимация сигнала (s) и детали к ней (d).

Схема алгоритма представлена на рис. 2.

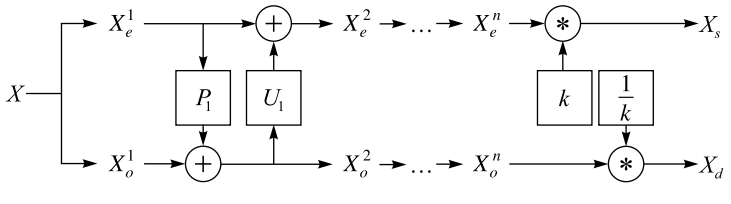

Рис. 2. Лифтинг-схема
Fig. 2. Lifting scheme

Следует отметить, что в описанном выше алгоритме этапы предсказания и обновления допускают реализацию с помощью операций свертки, однако их можно заменить на несколько операций сложения векторов, представляющих собой исходный масштабированный сигнал, сдвинутый на несколько позиций. Число элементов для сложения равно количеству элементов в соответствующей операции предсказания или обновления. Такой подход имеет смысл применять в том случае, если размеры фильтра в лифтинг-схемах невелики.

Существует большое множество вейвлет-семейств, таких как вейвлеты Хаара [13], Добеши [14] и др. В данной работе выбор сделан в пользу семейства CDF-9/7 [15], которое применяется, в частности, в формате сжатия изображений JPEG-2000. Лифтинг-схему для этого семейства можно найти в открытых источниках[1].

[1] *Getreuer P.* Wavelet CDF-9/7 implementation [Electronic resource]. URL: https://getreuer.info/posts/waveletcdf97/index.html (date of access: 21.10.2023).

Реализация сверточного вейвлет-блока. Сначала с помощью сверточного блока уменьшается количество каналов в сигнале исходного изображения. Затем несколько раз (параметр блока) применяется вейвлет-разложение сигнала: на вход всегда поступает наиболее медленно меняющаяся часть изображения (X_{LL}), для первой итерации используется выход сверточного блока. Далее выходной тензор после каждого шага разложения увеличивается в размерах так, чтобы размерность по ширине и высоте совпадала или с входом всего блока, или с выходом первого шага разложения (в этом случае не меняется размерность выхода первого шага преобразования), и объединяется вдоль оси каналов. Наконец, применяется еще один сверточный блок, который отвечает за то, чтобы вернуть необходимое количество каналов. Оно может или совпадать с входом целого вейвлет-блока (в этом случае выход блока можно использовать для поэлементного сложения с исходным сигналом и умножения на него), или быть фиксированным (в таком случае можно объединять выход блока с его входом с помощью операции конкатенации, но тогда нужно следить за совпадением размерностей по ширине и высоте). В данной работе структура входных и выходных сверточных блоков была следующей: свертка с ядрами размером 1 × 1, пакетная нормализация, активация Leaky ReLU. Стоит уточнить, что, хотя размер ядра часто указывают двумерным, на самом деле каждый фильтр в сверточном слое имеет трехмерную размерность: ширину, высоту и количество входных каналов. Поскольку вход текущего слоя напрямую зависит от выхода предыдущего слоя модели, то его размерность нельзя сделать параметром слоя. Также следует отметить, что данный размер ядра был выбран в целях ограничения параметров модели. Сопоставимый по количеству параметров вариант – использование сепарабельных сверток с ядрами размером 3 × 3, но были выбраны стандартные свертки, поскольку при такой реализации можно считать, что на входном тензоре вейвлет-блока каждый пиксел является взвешенной суммой только одного пиксела из всех каналов. То есть модель находит оптимальный с ее точки зрения попиксельный способ соединения карт признаков. В каждом вейвлет-блоке трижды применяется вейвлет-преобразование, это должно позволить собрать детали с нескольких уровней изображения. Общая схема блока представлена на рис. 3, где использованы следующие обозначения: InConvBlock – входной сверточный блок; OutConvBlock – выходной сверточный блок; Upscale_i – операция увеличения выхода изображения с помощью интерполяции в n раз на i-м шаге; DWT_i – операция дискретного вейвлет-преобразования на i-м шаге; Concat – операция конкатенации тензоров вдоль оси каналов; LL_i – один из выходов вейвлет-преобразования на i-м шаге.

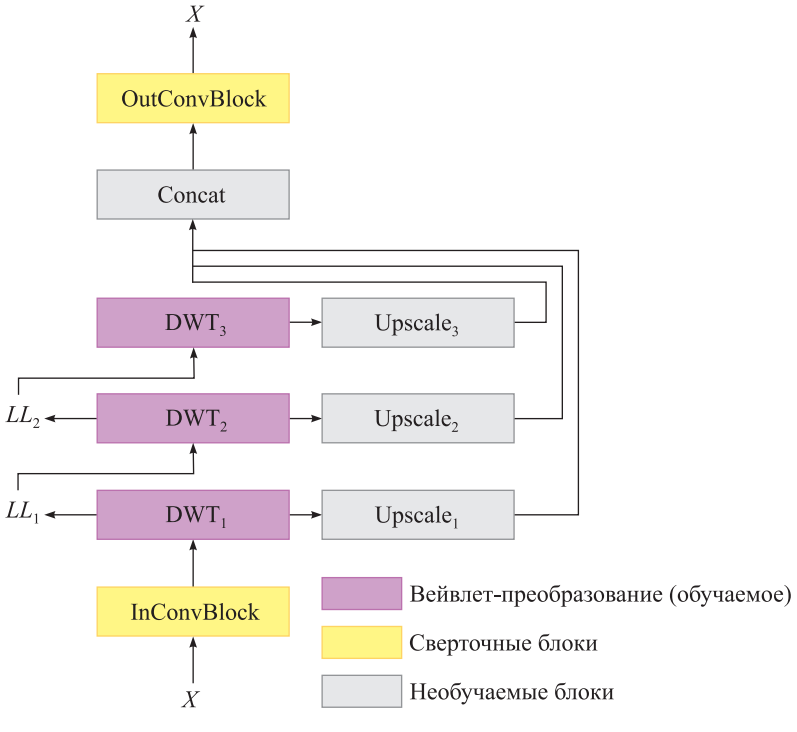

Рис. 3. Сверточный вейвлет-блок
Fig. 3. Convolutional wavelet block

Встраивание сверточного вейвлет-блока в модель. Есть множество вариантов того, как реализованный сверточный вейвлет-блок может быть встроен в сверточную нейронную сеть. В данной работе использованы два наиболее интересных способа.

Поскольку в большинстве моделей есть блоки, в которых происходит уменьшение входного разрешения тензора в 2 раза, то их возможности можно расширить, добавив дополнительную ветвь, которая будет состоять из реализованного сверточного вейвлет-блока. В результате входной сигнал будет анализироваться двумя независимыми извлекателями признаков. Следует отметить, что часто блоки уменьшения размерности состоят из большого количества сверточных слоев. Под таким блоком понимается часть модели, которая на выходе уменьшает размеры входного тензора в 2 раза вдоль осей, обозначающих ширину и высоту (количество каналов может как увеличиваться, так и уменьшаться), и при этом ее вход также имеет в 2 раза уменьшенные размеры относительно предыдущего шага. Во всех используемых в данной работе моделях можно выделить пять подобных блоков (вход первого из них представляет собой или исходное изображение, или выход DWT-блока, примененного к входной картинке). Далее складываются выходы исходного и реализованного вейвлет-блоков. В этом и заключается первый из предложенных способов встраивания сверточных вейвлет-блоков в модель. Он представлен на рис. 4, где использованы следующие обозначения: StrideBlock_i – блок уменьшения размерности на i-м шаге; CWBlock_i – сверточный вейвлет-блок на i-м шаге; WBlock – вейвлет-блок без сверточных слоев на входе и выходе; OutBlock – выходной блок без сверток, включающий в себя полносвязные, нормализационные и активационные слои, отвечающие за выход всей модели. Для удобства также указано, как меняется размерность входного тензора.

Второй способ встраивания сверточных вейвлет-блоков в модель состоит в том, чтобы добавлять дополнительную ветвь для анализа сигнала не к целому блоку уменьшения размерности, а к отдельным сверточным блокам, состоящим обычно из слоев свертки, нормализации и активации, каждый из которых может повторяться несколько раз. При таком подходе требуется добавить больше обучаемых параметров, но это должно улучшить возможности моделей по извлечению признаков из входных тензоров. На примере архитектуры ResNetV2-50 данный способ представлен на рис. 5, где использованы следующие обозначения: CWBlock – сверточный вейвлет-блок; Conv_i – сверточный слой на i-м шаге; BatchNorm_i – пакетная нормализация на i-м шаге; ReLU – слой активации.

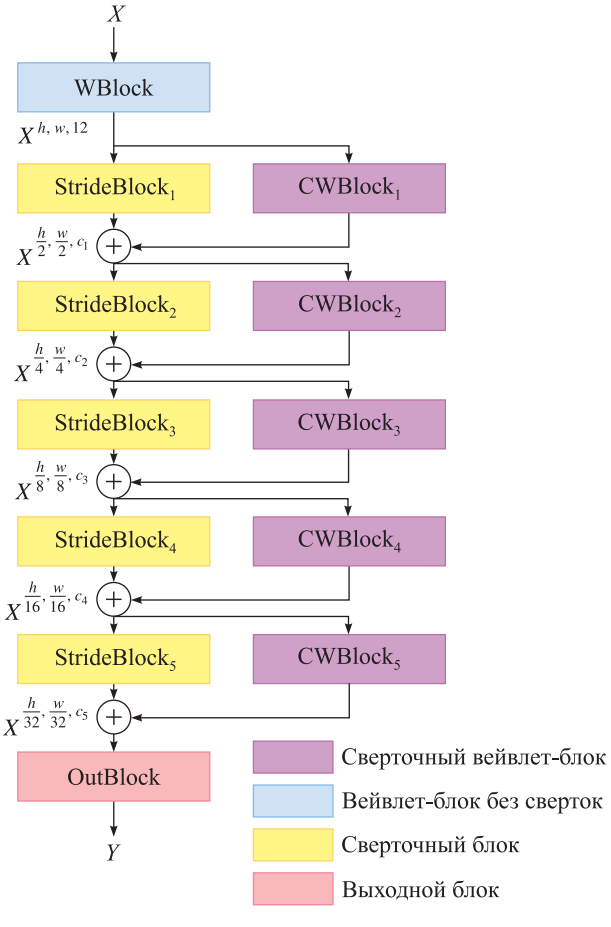

Рис. 4. Реализация первого способа встраивания сверточных вейвлет-блоков в модель

Fig. 4. Implementation of the first method of integrating convolutional wavelet blocks into the model

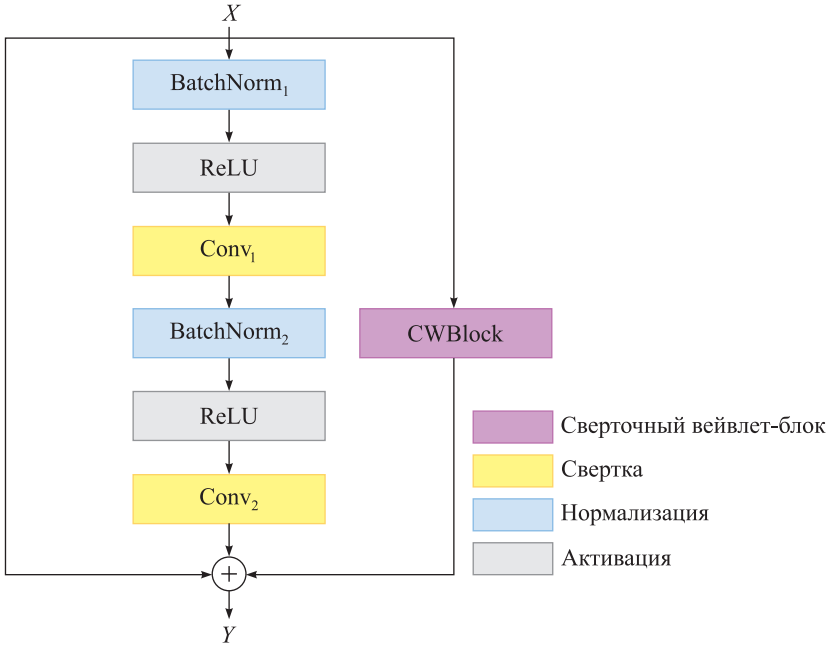

Рис. 5. Реализация второго способа встраивания сверточных вейвлет-блоков в модель (на примере архитектуры ResNetV2-50)

Fig. 5. Implementation of the second method of integrating convolutional wavelet blocks into the model (using the example of ResNetV2-50)

Модели, полученные первым из описанных способов, на рис. 7–10 и в таблице обозначены как «per_block», а модели, полученные вторым способом, – как «per_conv». В обоих случаях реализованный вейвлет-блок также применяется для анализа входного изображения, но без дополнительных входных и выходных сверток (только конкатенация выходов вейвлет-преобразования). В результате на вход первого сверточного блока поступит сигнал, у которого количество каналов составляет не 3, как у RGB-картинки, а 12 (по 3 уровня вейвлет-разложения для каждого из RGB-каналов, а также вход для каждого разложения).

Подготовка набора данных. Для проведения экспериментов было решено создать набор данных на основе датасета LSUN (Large-Scale Scene Understanding Challenge) [16], поскольку в нем представлено относительно большое количество категорий объектов с разным числом доступных изображений, а сами картинки имеют размер от 256 × 256 пк, что почти совпадает с размером входных изображений для исследуемых моделей при обучении на стандартном датасете ImageNet[2] – 224 × 224 пк. Для создаваемого набора данных были выбраны пять категорий: «автобус», «автомобиль», «мотоцикл», «корабль» и «самолет». Для обучения и валидации взяты первые 10 000 изображений из каждой категории, а для тестирования использованы по 10 000 картинок, начиная с 100 000-й. Во всех категориях количество данных позволяло это делать. При необходимости размер любой из них можно было бы увеличить, но 10 000 изображений каждой категории на обучение вместе с валидацией и 10 000 картинок на тестирование должно быть вполне достаточно для задачи классификации изображений.

Исходные наборы данных хранятся в формате базы данных LMDB (Lightning Memory-Mapped Database), поэтому для удобства работы каждое изображение извлекалось в отдельный файл перед запуском обучения моделей. Картинки сначала приводились к квадратной форме путем удаления пикселов вдоль большой стороны так, чтобы оставшаяся часть была в центре исходного изображения, а затем с помощью интерполяции методом Ланцоша уменьшались в размере до 256 × 256 пк. Поскольку часто целевой объект находится в центре изображений, удаление пикселов из картинок не должно навредить качеству модели, но может помочь избежать ситуации, при которой бóльшая часть изображений заполняется некоторыми искусственными значениями, а сам объект занимает меньшую часть картинки. Данная процедура применялась к изображениям всех датасетов: тренировочного, валидационного и тестового.

На рис. 6 представлено по восемь примеров из каждой категории, занимающей отдельную строку.

[2] ImageNet large scale visual recognition challenge (ILSVRC) [Electronic resource]. URL: https://www.image-net.org/challenges/LSVRC/ (date of access: 21.10.2023).

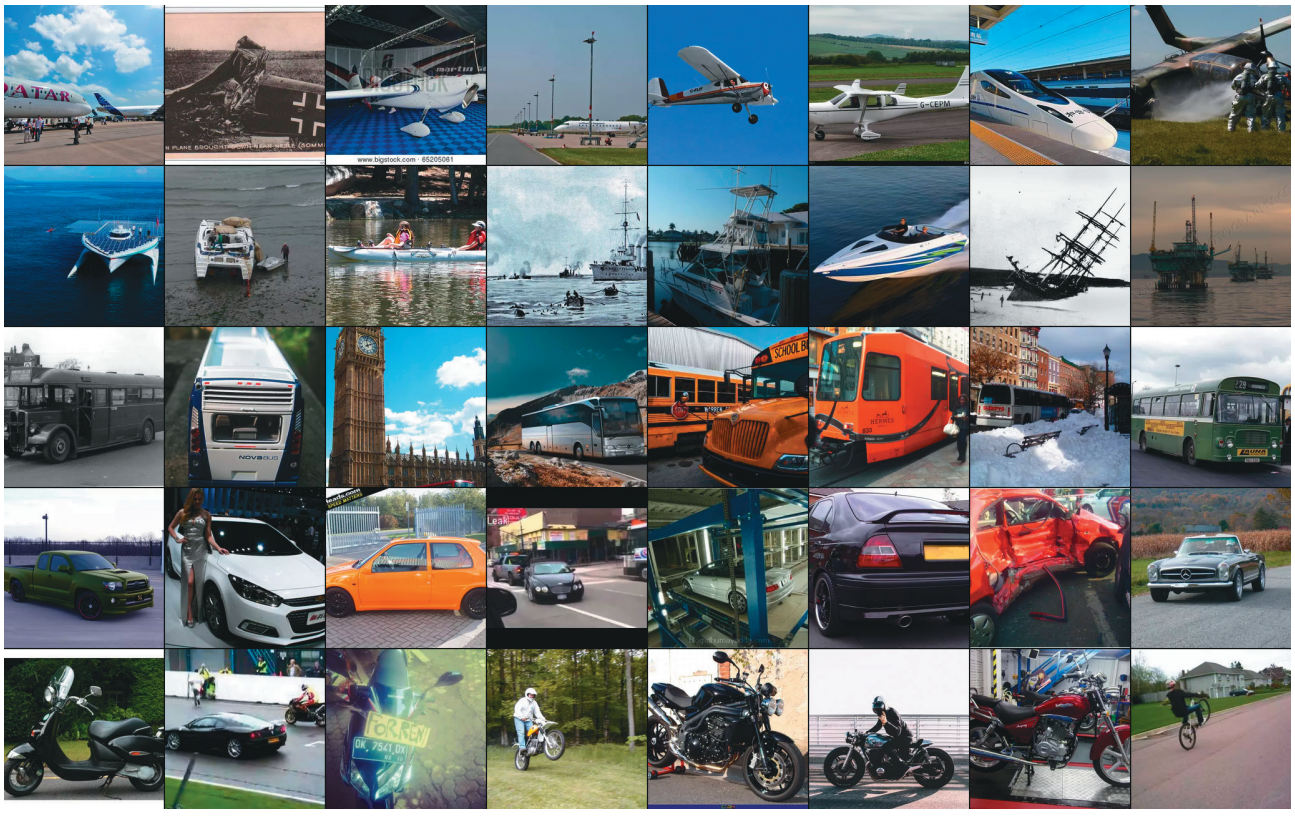

Рис. 6. Примеры изображений из подготовленного датасета
Fig. 6. Images examples from the prepared dataset

Схема обучения

Перед созданием итогового датасета для обучения данные из каждой категории делились на части по 85 % (обучение) и 15 % (валидация). Далее тренировочные данные объединялись в один общий набор и случайно перемешивались с зафиксированным параметром случайности (*seed*). Тестовые данные дополнительно никак не обрабатывались, поскольку изначально отбирались так, чтобы каждая категория была представлена одинаковым количеством изображений. Таким образом, тренировочный, валидационный и тестовый датасеты содержали 42 500, 7500 и 50 000 изображений соответственно. В качестве аугментаций использовалось вырезание случайной области исходного изображения, целевая ширина и высота которой определялись независимо друг от друга случайным образом в диапазоне [0,75; 1,0] от исходных размеров. Полученная картинка приводилась к исходному разрешению с помощью билинейной интерполяции. Также с вероятностью 0,5 применялось отражение изображения относительно вертикальной оси. При проверке качества модели на валидационной и тестовой выборках аугментации отключались, а слои пакетной нормализации переводились в тестовый режим (для нормализации использовались накопленные значения среднего и стандартного отклонения). В качестве препроцессинга для всех моделей посредством линейного отображения изображения переводились из диапазона [0; 255] в диапазон [–1,0; 1,0], а не [0; 1,0]. Данное действие обусловлено тем, что значения весов входного сверточного слоя инициализируются из равномерного распределения случайных чисел, которое симметрично относительно нуля (использовался метод Глорота). Такой подход к масштабированию входных изображений может помочь обучению первого слоя моделей, поскольку минимальное и максимальное значения входных изображений также будут симметричны относительно нуля, а максимальное по модулю значение не превысит 1,0. На вход моделей поступали изображения размером 256 × 256 пк. Для обучения моделей использовался оптимизатор Adam со скоростью обучения 0,001 и размером пакета (батча), равным 64. Поскольку классы в данных были полностью сбалансированы, то в качестве функции потерь выбрана категориальная кросс-энтропия, а в качестве метрики – точность (*accuracy*). Максимальная продолжительность обучения устанавливалась равной 100 эпохам с возможностью ранней остановки при отсутствии улучшения метрики на валидационном наборе в течение 15 эпох.

Модели были реализованы с помощью фреймворка TensorFlow. Обучение проходило на вычислительном стенде с графическим процессором Nvidia RTX 3090 (24 Гб).

Результаты

В таблице представлены результаты, показанные моделями на тестовом наборе данных. Можно видеть, что при грамотном использовании сверточных вейвлет-блоков общее количество параметров моделей незначительно увеличивается, но при этом ощутимо уменьшается количество параметров исходных моделей. В итоге, получив лишь небольшую потерю в качестве, можно добиться уменьшения объема модели вплоть до 40 %. Стоит отметить, что из протестированных моделей лучше всего себя показала архитектура MobileNetV2. Данный факт может быть связан с тем, что авторы этой модели изначально закладывали возможность контролировать ее сложность с помощью константы.

Для архитектуры ResNetV2-50 также были протестированы версии модели, в которых последний (пятый) блок уменьшения размерности не подвергался изменениям (обозначены как «nob5»). Использование такого варианта модели связано с тем, что именно в данном блоке содержится очень большое количество параметров, и при определенных способах встраивания вейвлет-преобразований их число существенно увеличивается. Результаты сходимости стандартной модели ResNetV2-50 на валидационном наборе представлены на рис. 7, а результаты сходимости модели без модификации пятого блока – на рис. 8.

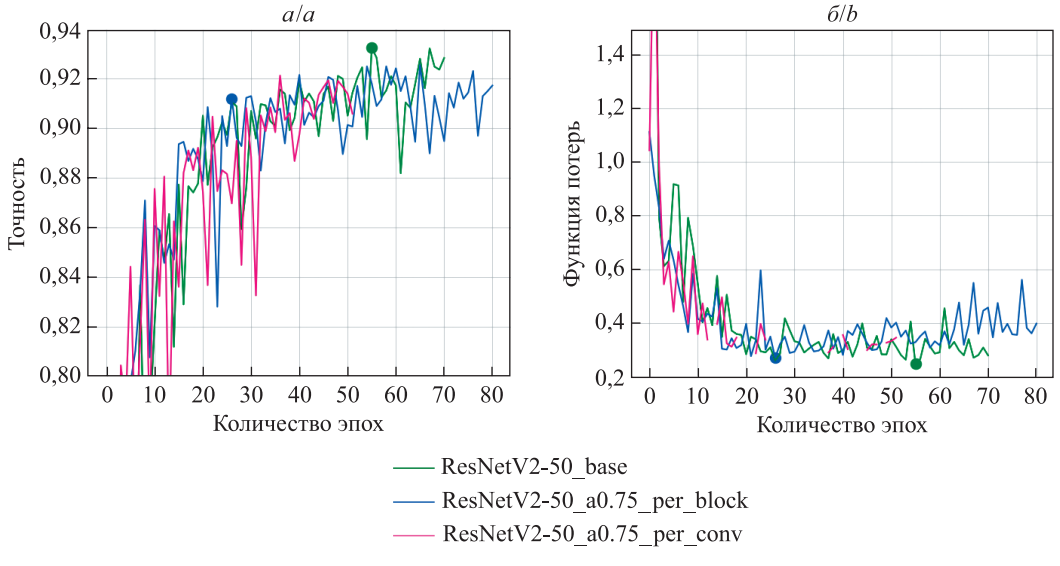

Рис. 7. Сходимость стандартной модели ResNetV2-50
(*а* – точность; *б* – функция потерь)

Fig. 7. Convergence of the standard ResNetV2-50 model
(*a* – valid accuracy; *b* – valid loss)

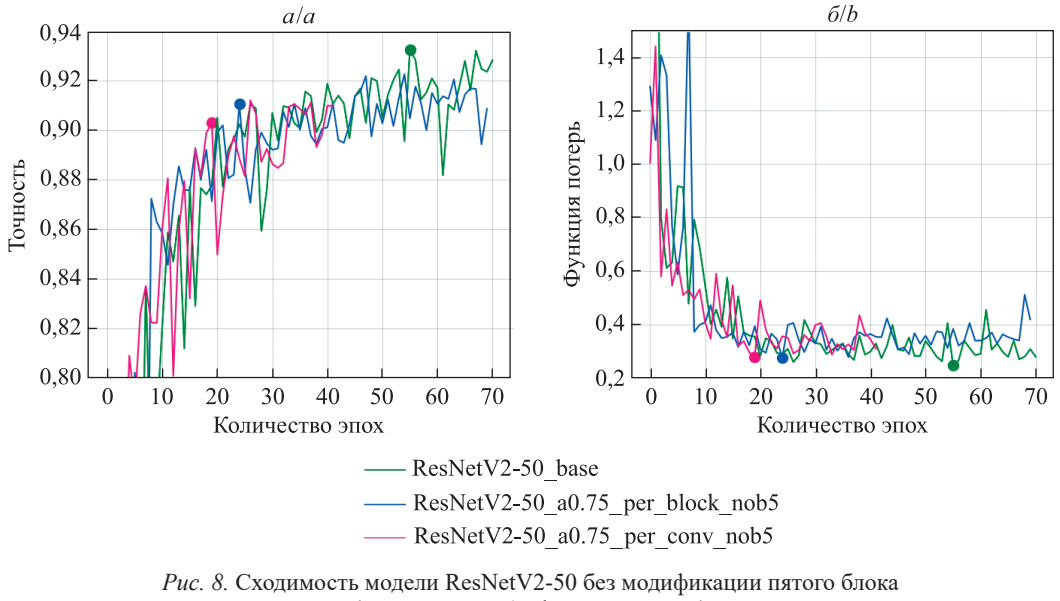

Рис. 8. Сходимость модели ResNetV2-50 без модификации пятого блока
(*а* – точность; *б* – функция потерь)

Fig. 8. Convergence of the ResNetV2-50 model without modification of the fifth block
(*a* – valid accuracy; *b* – valid loss)

Модель MobileNetV2 оказалась оптимальным вариантом для рассматриваемой задачи: исходная модель показала наилучшие результаты на тестовой выборке среди всех протестированных архитектур (она также имеет меньший размер), а лучшая модифицированная версия потеряла в качестве менее 0,3 % на тестовой выборке, но при этом ее размер удалось уменьшить на 33,7 %. Результаты сходимости моделей данного семейства на валидационной выборке представлены на рис. 9.

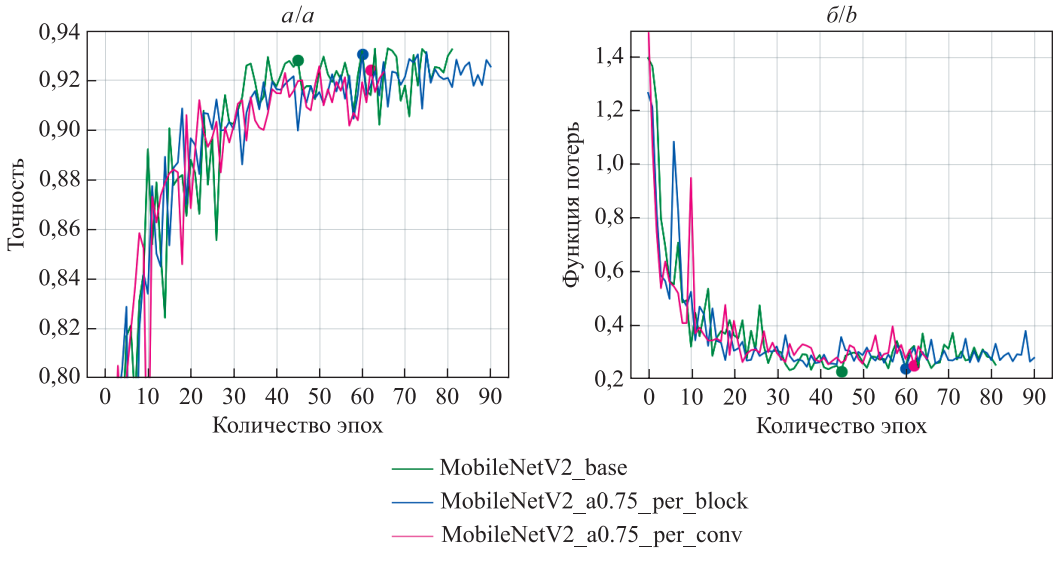

Рис. 9. Сходимость модели MobileNetV2
(*а* – точность; *б* – функция потерь)

Fig. 9. MobileNetV2 model convergence
(*a* – valid accuracy; *b* – valid loss)

Обучение модели EfficientNetV2-B0 проходило крайне тяжело: значение метрики на валидационном датасете не могло стабилизироваться, и с течением времени модель расходилась, причем данное поведение не зависело от того, использовались ли стандартные настройки обучения или настройки обучения, предложенные авторами архитектуры в работе [12]. Итоговые результаты заметно уступают результатам других протестированных семейств моделей. Графики сходимости модели EfficientNetV2-B0 на валидационном наборе данных представлены на рис. 10.

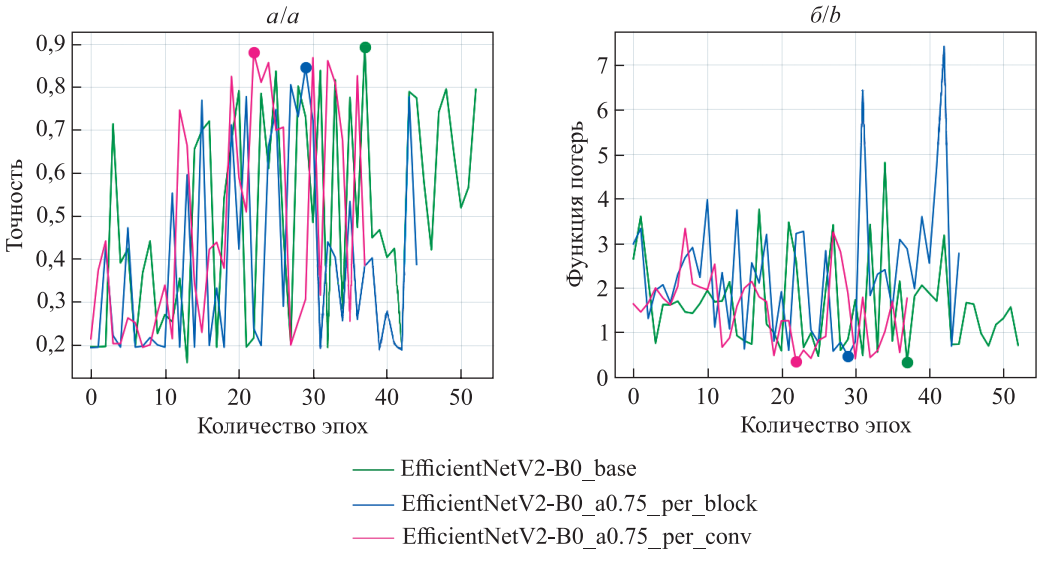

Рис. 10. Сходимость модели EfficientNetV2-B0
(*а* – точность; *б* – функция потерь)

Fig. 10. EfficientNetV2-B0 model convergence
(*a* – valid accuracy; *b* – valid loss)

Сравнение результатов

Comparison of results

Модель	Количество общих параметров	Количество DWT-параметров	Доля DWT-параметров	Размер, Мб	Метрика (*accuracy*)
ResNetV2-50_base	$23{,}56 \cdot 10^6$	0	0	90,1	0,934 9
ResNetV2-50_a0.75_per_block	$16{,}63 \cdot 10^6$	$3{,}22 \cdot 10^6$	0,194	60,7 (–32,6 %)	0,922 7 (–0,012 2)
ResNetV2-50_a0.75_per_conv	$38{,}15 \cdot 10^6$	$24{,}74 \cdot 10^6$	0,648	111,0 (+23,2 %)	0,922 9 (–0,012 0)
ResNetV2-50_a0.75_per_block_nob5	$20{,}08 \cdot 10^6$	$786{,}0 \cdot 10^3$	0,039	76,1 (–15,5 %)	0,928 5 (–0,006 4)
ResNetV2-50_a0.75_per_conv_nob5	$29{,}20 \cdot 10^6$	$9{,}89 \cdot 10^6$	0,339	97,8 (+8,5 %)	0,911 3 (–0,023 6)
MobileNetV2_base	$2{,}26 \cdot 10^6$	0	0	8,9	0,935 5
MobileNetV2_a0.75_per_block	$1{,}49 \cdot 10^6$	$94{,}0 \cdot 10^3$	0,063	5,9 (–33,7 %)	0,932 4 (–0,003 1)
MobileNetV2_a0.75_per_conv	$1{,}62 \cdot 10^6$	$221{,}0 \cdot 10^3$	0,137	6,3 (–29,2 %)	0,928 6 (–0,006 9)
EfficientNetV2-B0_base	$5{,}92 \cdot 10^6$	0	0	22,8	0,894 4
EfficientNetV2-B0_a0.75_per_block	$3{,}52 \cdot 10^6$	$42{,}0 \cdot 10^3$	0,012	13,7 (–39,9 %)	0,884 4 (–0,010 0)
EfficientNetV2-B0_a0.75_per_conv	$3{,}86 \cdot 10^6$	$380{,}0 \cdot 10^3$	0,098	15 (–34,2 %)	0,848 8 (–0,045 6)

Примечание. В графах «размер» и «метрика (*accuracy*)» в скобках указана величина изменения соответствующих значений относительно базовой модели.

Заключение

В данной работе реализован алгоритм быстрого дискретного вейвлет-преобразования для семейства CDF-9/7 на основе соответствующей лифтинг-схемы. Полученный алгоритм распространен на случай четырехмерного входа для возможности работы с изображениями в сверточных нейронных сетях. Для реализации алгоритма использовались векторные операции сложения и умножения, благодаря чему его легко можно встроить в существующие модели компьютерного зрения, сохранив возможность их обучения методом обратного распространения ошибки. Константы, использованные в лифтинг-схеме, также можно сделать обучаемыми параметрами. На основе разработанного алгоритма реализован сверточный вейвлет-блок, который был успешно встроен в модели ResNetV2-50, MobileNetV2 и EfficientNetV2-B0. Исходные и модифицированные версии этих моделей проверены в задаче классификации изображений на наборе данных, созданном на основе датасета LSUN. Благодаря использованию вейвлет-блоков удалось масштабировать исходные модели за счет уменьшения количества фильтров в сверточных слоях, сохранив при этом сопоставимый уровень качества. При оптимальном встраивании вейвлет-блоков в модели разница в метриках составляет 0,3–1,2 %, однако полученные сети занимают на 30–40 % меньший объем памяти на диске. В лучшем случае (модель на основе MobileNetV2) удалось добиться сжатия размера весов с 8,9 до 5,9 Мб (–33,7 %), потеряв в качестве лишь 0,31 %.

Библиографические ссылки / References

1. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556v6 [Preprint]. 2015 [cited 2024 January 2]: [14 p.]. Available from: https://arxiv.org/abs/1409.1556v6.

2. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. arXiv:1512.03385 [Preprint]. 2015 [cited 2024 January 2]: [12 p.]. Available from: https://arxiv.org/abs/1512.03385.

3. Tan M, Le QV. EfficientNet: rethinking model scaling for convolutional neural networks. arXiv:1905.11946v5 [Preprint]. 2020 [cited 2024 January 2]: [11 p.]. Available from: https://arxiv.org/abs/1905.11946v5.

4. Cheng H, Zhang M, Shi JQ. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. arXiv:2308.06767 [Preprint]. 2023 [cited 2024 January 2]: [23 p.]. Available from: https://arxiv.org/abs/2308.06767.

5. Blake C, Orr D, Luschi C. Unit scaling: out-of-the-box low-precision training. arXiv:2303.11257v2 [Preprint]. 2023 [cited 2024 January 2]: [29 p.]. Available from: https://arxiv.org/abs/2303.11257v2.

6. Zhang Shuai, Guangdi Ma, Yang Weichen, Fang Zuo, Ablameyko SV. Car parking detection in images by using a semi-supervised modified YOLOv5 model. *Journal of the Belarusian State University. Mathematics and Informatics*. 2023;3:72–81. EDN: XVDRSN.

7. Singh A, Kingsbury N. Efficient convolutional network learning using parametric log based dual-tree wavelet ScatterNet. arXiv: 1708.09259 [Preprint]. 2017 [cited 2024 January 2]: [8 p.]. Available from: https://arxiv.org/abs/1708.09259.

8. Li Q, Shen L, Guo S, Lai Z. Wavelet integrated CNNs for noise-robust image classification. arXiv:2005.03337v2 [Preprint]. 2020 [cited 2024 January 2]: [17 p.]. Available from: https://arxiv.org/abs/2005.03337v2.

9. Wolter M, Blanke F, Heese R, Garcke J. Wavelet-packets for deepfake image analysis and detection. arXiv:2106.09369v4 [Preprint]. 2022 [cited 2024 January 2]: [29 p.]. Available from: https://arxiv.org/abs/2106.09369v4.

10. He K, Zhang X, Ren S, Sun J. Identity mappings in deep residual networks. arXiv:1603.05027v3 [Preprint]. 2016 [cited 2024 January 2]: [15 p.]. Available from: https://arxiv.org/abs/1603.05027v3.

11. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. MobileNetV2: interested residuals and linear bottlenecks. arXiv: 1801.04381v4 [Preprint]. 2019 [cited 2024 January 2]: [14 p.]. Available from: https://arxiv.org/abs/1801.04381v4.

12. Tan M, Le QV. EfficientNetV2: smaller models and faster training. arXiv:2104.00298v3 [Preprint]. 2021 [cited 2024 January 2]: [11 p.]. Available from: https://arxiv.org/abs/2104.00298v3.

13. Lepik Ü, Hein H. *Haar wavelets: with applications.* Cham: Springer; 2014. X, 207 p. (Hillermeier C, Schröder J, Weigand B, editors. Mathematical engineering). DOI: 10.1007/978-3-319-04295-4.

14. Daubechies I. *Ten lectures on wavelets.* Philadelphia: Society for Industrial and Applied Mathematics; 1992. XIX, 357 p. (CBMS-NSF regional conference series in applied mathematics; volume 61).

15. Cohen A, Daubechies I, Feauveau J-C. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics.* 1992;45(5):485–560. DOI: 10.1002/cpa.3160450502.

16. Yu F, Seff A, Zhang Y, Song S, Funkhouser T, Xiao J. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. arXiv:1506.03365v3 [Preprint]. 2016 [cited 2024 January 2]: [9 p.]. Available from: https://arxiv.org/abs/1506.03365v3.

Получена 07.01.2024 / исправлена 16.06.2024 / принята 16.06.2024.
Received 07.01.2024 / revised 16.06.2024 / accepted 16.06.2024.