УДК 004.01

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ВОССТАНОВЛЕНИЯ ИЗОБРАЖЕНИЙ ПОСЛЕ СЖАТИЯ С ПОТЕРЯМИ

© 2018 В.Ф. Барабанов[1], Н.И. Гребенникова[1], С.Л. Кенин[2], Д.А. Юров[1]

[1]Воронежский государственный технический университет, г. Воронеж, Россия
[2]Atos, г. Воронеж, Россия

Аннотация: предлагается технология восстановления изображений, основанная на применении нейросетевого обучения и позволяющая удалять артефакты сжатия с изображений, обработанных кодеком JPEG, применяющим алгоритмы сжатия с потерями. Нейронные сети не программируются, в отличие от традиционных программных средств, они обучаются. Возможность обучения - одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных, частично искажённых данных. Предложена структура и рассмотрены функции программного средства для восстановления изображений при помощи глубинных нейронных сетей. На базе предложенной структуры разработана библиотека методов и приложение, позволяющее подготовить нейронную сеть к эксплуатации и протестировать ее на несинтетических экземплярах сжатых изображений. Значительное внимание уделено рассмотрению наиболее релевантных архитектур нейронных сетей для данной задачи и библиотек, упрощающих их реализацию. Получены и проанализированы результаты работы приложения

Ключевые слова: автоэнкодер, глубокое обучение, сверточная нейронная сеть, Keras

Введение

В связи с всё возрастающим объемом накопленных графических данных при снижении темпов увеличения объемов хранилищ данных становится все более острой проблема восстановления изображений после сжатия с потерями или иных повреждений, в том числе физических. Изображение, сохраненное на удаленных веб-серверах, использующих технологии сжатия графики для сохранения места на накопителях, после нескольких итераций зачастую бывает полностью отличным от оригинала или даже совершенно неразличимым.

Технологии восстановления изображения, примененные между итерациями либо непосредственно к файлу-источнику, могли бы позволить избежать потери данных, при этом сохраняя преимущества форматов сжатия с потерями, таких как JPEG, PNG и GIF.

Результаты обработки изображений подобным методом могут быть использованы в различных профессиональных предметных областях, вплоть до астрофотографии и криминалистики; в широком смысле область применения данных алгоритмов не ограничена ничем.

В рамках данной работы спроектировано и реализовано приложение, позволяющее частично восстанавливать исходное изображение на основе изображения, уже затронутого артефактами компрессии. До разработки нового программного средства следует рассмотреть уже существующие аналоги и проанализировать их достоинства и недостатки.

Обзор современных программных средств для восстановления изображений

Программные продукты для восстановления изображений сложно классифицировать ввиду недостатка информации о проприетарных компонентах и отсутствия строгой классификации. Тем не менее можно выделить некоторые проекты, в рамках которых была частично или полностью реализована данная функциональность [1, 2, 3]:

- Magic Pony Technology – приложение, разработанное одноименной компанией с целью увеличения разрешения входных изображений; компания использует машинное обучение, подготовленная нейросеть не просто интерполирует пиксели, а добавляет недостающие детали;

- AKVIS Noise Buster - программа для подавления цифрового шума на изображении, разработанная компанией AKVIS. AKVIS Noise Buster подавляет шумы матрицы цифровой камеры и шумы, появляющиеся при сканировании фотоснимка, уменьшает зернистость и устраняет неоднородные цветовые пятна на изображении, сохраняя детали и резкость гра-

ниц. Программа убирает как яркостной шум, так и цветовой (хроматический). Первый проявляется в виде искажающих элементов, отличающихся по яркости (например, неровности на коже), второй — в виде маленьких пятен, имеющих различие в цвете (пятна красного или синего цвета);

- Topaz DeJPEG – программный продукт, разработанный компанией Topaz Labs и предназначенный для улучшения качества изображений в формате JPEG. Topaz DeJPEG устраняет артефакты сжатия JPEG, которые возникают в результате сохранения изображения в формате JPEG, что позволяет восстановить детали и цвет изображения. Основной алгоритм DeJPEG исследует все изображения и удаляет все артефакты, сохраняя детали и улучшая естественные качества изображения. Разработчик заявлет, что данный продукт гарантирует качество изображения, соответствующее формату RAW при сохранении преимуществ формата JPEG.

Все вышеперечисленные продукты либо являются внутренними разработками различных компаний для собственных нужд, либо предоставляются на коммерческой основе и с закрытым исходным кодом, при этом только MagicPony Technology демонстрирует возможность удаления артефактов сжатия JPEG, так что разработка современного программного продукта для восстановлений качества изображений с наглядным интерфейсом, открытым исходным кодом и публичным доступом, является актуальной задачей, поэтому были выдвинуты предполагаемые требования для эффективной разработки приложения.

Требования к приложению для удаления артефактов компрессии

Приложение для удаления артефактов сжатия должно обладать следующими характеристиками:

- иметь возможность как использовать обученную нейронную сеть, так и тренировать ее заново для специфических классов изображений;

- иметь возможность быстрой настройки гиперпараметров сети и сохранения сетей с различными весами после обучения;

- иметь возможность использовать обучающие и валидирующие выборки данных с минимальными ограничениями;

- быть доступным, то есть программное обеспечение должно находиться в свободном доступе с открытым исходным кодом, что позволит привлечь большее количество исследователей-энтузиастов.

Для разработки программного продукта использована IDE PyCharm 2017.3, язык разработки – Python, библиотеки разработки – Keras [4], TensorFlow [5]; также была предложена диаграмма использования приложения и его модульная структура.

Состав, структура и функции программного средства

Созданная программа предназначена для восстановления изображений при помощи глубинных нейронных сетей и позволяет удалять артефакты сжатия с изображений.

На рис. 1 представлена схема работы приложения, на которой приведены основные функции программного обеспечения.

Рис. 1. Схема работы программного средства

Для его функционирования пользователю необходимо подготовить данные для обучения – наборы изображений в формате PNG. Далее входные данные упаковываются в Numpy-массивы, которые в дальнейшем также сохраняются в файловой системе для большего удобства в использовании. Последующее обучение сети скрыто от пользователя, мониторинг процесса обучения осуществляется только с консоли приложения либо в инструменте мониторинга TensorBoard. Результатом обучения сети являются значения весов, которые впоследствии также сохраняются в файловой системе для будущих запусков либо использования в отдельных клиентских приложениях.

На рис. 2 приведена модульная структура приложения. Приложение использует базовые принципы ООП, такие как инкапсуляция и полиморфизм. В использовании более сложных парадигм или паттернов проектирования в данном случае нет необходимости, так как конечный продукт представляет собой компактный проект в среде PyCharm, архитектура которого, тем не менее, может быть усложнена без значительных затрат.

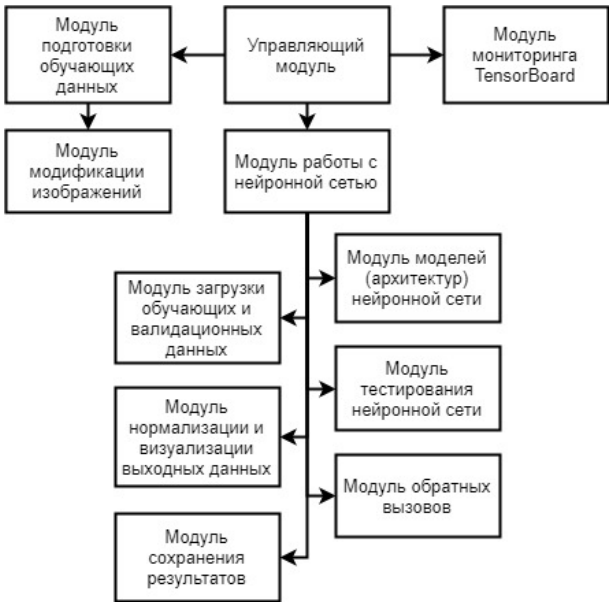

Рис. 2. Модульная структура приложения

В текущей реализации программного средства действия пользователя, требуемые от пользователя, ограничены поиском тестовых данных и их подготовкой для работы, в дальнейшем приложение работает в автоматическом режиме.

Управляющий модуль организует взаимодействие остальных модулей, позволяя абстрагироваться от деталей реализации конкретных модулей при их применении.

Модуль подготовки обучающих данных сканирует заданные директории на наличие изображений, преобразует их в формат массива Numpy, сохраняет его в файловой системе и возвращает управление основному процессу. Вспомогательный модуль модификации изображений добавляет артефакты сжатия JPEG к обучающим данным в автоматическом режиме. Так как для эффективного глубокого обучения требуются максимально большие объемы входных данных, обучение на «естественных» экземплярах не представляется возможным. В рамках данной работы использовалась обучающая выборка в 200000 изображений формата PNG размером 64*64 пикселя и валидационная выборка из 20000 изображений размером 64*64 пикселя.

Модуль мониторинга TensorBoard представляет собой отдельный процесс, запускающий специализированное серверное приложение на базе системы Continious Integration Jenkins. Так как приложение для мониторинга должно предоставлять данные в удобном для восприятия виде, также запускается предпочитаемый пользователем браузер, в окне которого отображаются веб-инструменты мониторинга процесса обучения. Пример работы TensorBoard представлен на рис. 3 (график измерения точности – отличие среднеквадратичного отклонения в процессе обучения от среднеквадратичного отклонения на валидационнной выборке) и рис. 4 (динамика среднеквадратичного отклонения на протяжении эпох обучения).

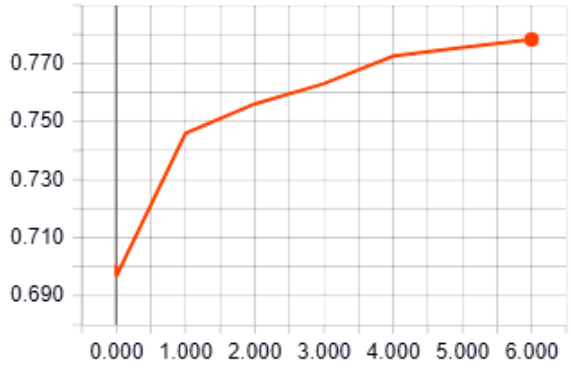

Рис. 3. График точности в TensorBoard

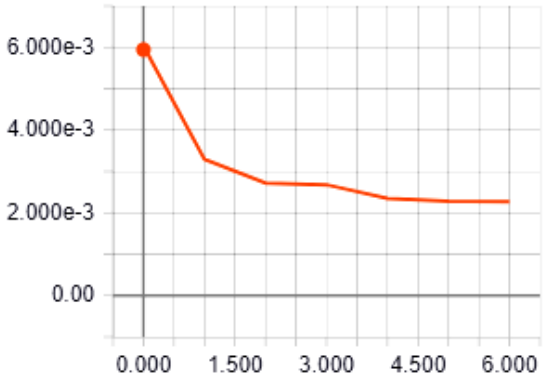

Рис. 4. График изменения среднеквадратичного отклонения

Для линейного нейрона список параметров значительно уже, однако не стоит считать, что линейный нейрон имеет малые вычислительные способности.

Модуль моделей/архитектур нейронной сети хранит подготовленные архитектуры, которые были протестированы на различных тестовых данных и показали свою эффективность.

В рамках данной работы было рассмотрено два вида архитектур нейронных сетей, относящиеся к глубокому обучению: автоэнкодеры и сверточные сети.

Автоэнкодер - алгоритм обучения без учителя, выходной вектор которого равен входному вектору признаков. Одной из самых распространенных архитектур автоэнкодера является нейронная сеть прямого распространения без обратных связей, содержащая входной, скрытый и выходной слои. В отличие от персептрона выходной слой автоэнкодера должен содержать столько же нейронов, сколько и входной слой. Данные на входном слое сжимаются на скрытом слое и восстанавливаются на выходном слое, таким образом, выделяются «скрытые признаки». К сожалению, автоэнкодеры неэффективны в решении слабо формализуемых задач, рассматриваемых в работе: изображения практически невозможно разделить на классы. Тем не менее, данная архитектура показала себя эффективной при работе с группами монохромных символов – более простая и производительная по сравнения со сверточной сетью. Архитектура автоэнкодера представлена на рис. 5.

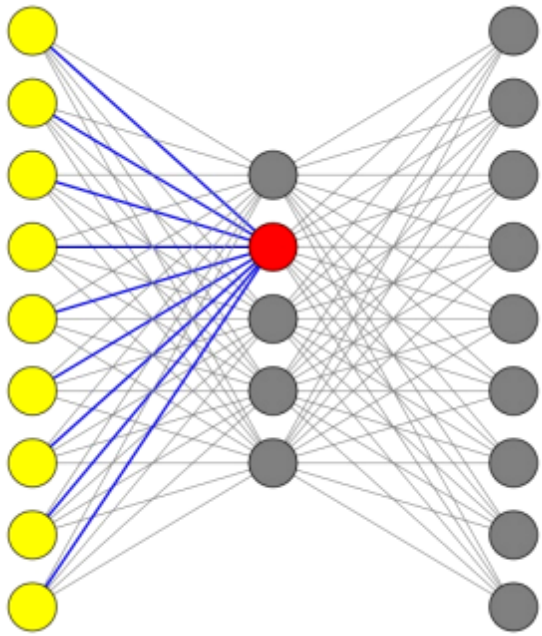

Рис. 5. Архитектура автоэнкодера

Автоэнкодер подходит для процесса обучения на данных небольшого размера, а при больших данных количество параметров (количество весов между входным и скрытым слоями, а также количество весов между скрытым и выходным слоями) существенно вырастет. Например, если при классификации изображений размером $m \times n = 8 \times 8$, мы хотим выделять $k = 50$ признаков, тогда входной и выходной слои имеют $m \times n = 8 \times 8 = 64$ нейронов, а скрытый слой имеет $k = 50$ нейронов, количество весов равно $2 \times m \times n \times k = 2 \times 8 \times 8 \times 50 = 6400$. Но если картинки имеют размер $m \times n = 100 \times 100$, то количество весов будет $2 \times 100 \times 100 \times 50 = 1\,000\,000$[6].

Чтобы размер нейронной сети не был большим и не зависел от размера обучаемых данных, можно использовать специальную структуру нейронной сети, называемую сверточной нейронной сетью (англ. Convolutional Neural Network – CNN). При обучении для каждого образца на вход сверточной нейронной сети попадают не все изображения $m \times n$, а только их часть $r \times c$, например 10×10, эта часть называется фильтром или ядром и она сдвигается по большой картинке. Такая сверточная нейронная сеть имеет фиксированное количество нейронов на входном и выходном слоях $r \times c = 10 \times 10 = 100$, а количество весов $2 \times r \times c \times k = 2 \times 10 \times 10 \times 50 = 10\,000$.

Для каждого большого изображения $m \times n$ выбираются маленькие изображения $r \times c$ для

обучения, их количество равно $(m - r + 1) \times (n - c + 1)$. Визуализация свертки представлена на рис. 6.

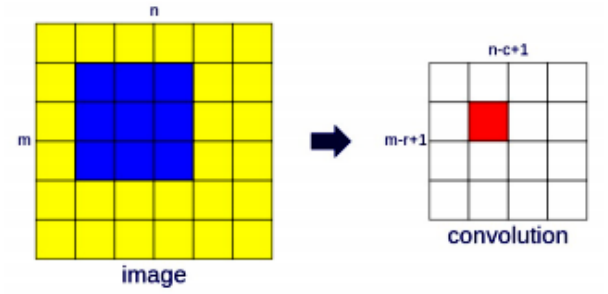

Рис. 6. Свертка

При использовании сверточной нейронной сети возникает проблема – большое количество выделенных признаков. Для одной большой картинки $m \times n$ будет $(m-r+1)\times(n-c+1)$ маленьких картинок $r\times c$, и количество выделенных признаков равно $k \times (m - r + 1) \times (n - c + 1) = 50 \times (100 - 10 + 1) \times (100 - 10 + 1) = 414\,050$. Использование такого огромного количества признаков для классификации оказалось неэффективно. Для уменьшения размера пространства признаков проводится субдискретизация (англ. pooling), разделив карту признаков, полученных от сверточной нейронной сети, на фиксированное количество частей p (рис. 7), на этом рисунке $p = pm \times pn = 2 \times 2 = 4$ и на каждой части вычисляется ее максимальное значение (англ. max pooling) или среднее значение (англ. mean pooling) [7].

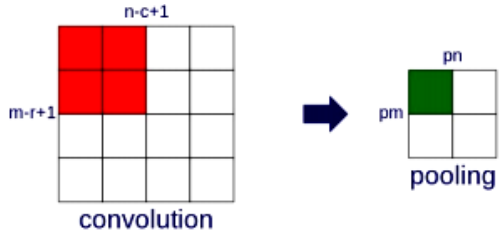

Рис. 7. Субдискретизация (субсэмплинг)

Структура работы сверточной сети, разработанной в рамках данной работы, представлена на рис. 8.

Модуль загрузки обучающих и валидационных данных служит оболочкой для доступа к данным, предоставляемым модулем подготовки обучающих данных, загружая их в оперативную память. Также данный модуль меняет последовательность изображений в массиве данных, позволяя избежать проблему перенасыщения сети.

Модуль тестирования нейронной сети является оболочкой для доступа к методу Evaluate библиотеки Keras, позволяя получить усредненные значения погрешности и среднеквадратичного отклонения на протяжении всего времени обучения сети.

Модуль обратных вызовов используется для вызова заданных действий после каждой эпохи/итерации обучения нейронной сети. В данном случае используются три стандартные функции: EarlyStopping, ModelCheckpoint и TensorBoard.

EarlyStopping используется для остановки обучения в случае, если изменение погрешности перестает уменьшаться на протяжении заданного количества эпох. В рамках данной работы количество эпох было ограничено двумя.

ModelCheckpoint обновляет значение весов нейронной сети в кэше Keras после каждой эпохи, кроме того, можно производить обновление только в случае, когда значение погрешности не превышает значение на предыдущей эпохе.

Модуль нормализации и визуализации выходных данных корректирует значения данных, полученных от нейронной сети для того, чтобы параметры RGB изображения не принимали отрицательные значения вследствие возникающих внутри сети шумов, что не позволяет отображать данные изображения. Визуализация производится при помощи библиотеки matplotlib, вызовы методов которой обернуты в методы, позволяющие сравнивать исходные, модифицированные и восстановленные последовательности изображений.

Модуль сохранения результатов обеспечивает сохранение весов нейронной сети в файловой системе по завершению работы.

Модульная структура в сочетании с использованием интегрированных сред разработки и библиотек позволяет значительно повысить скорость реализации приложения [8].

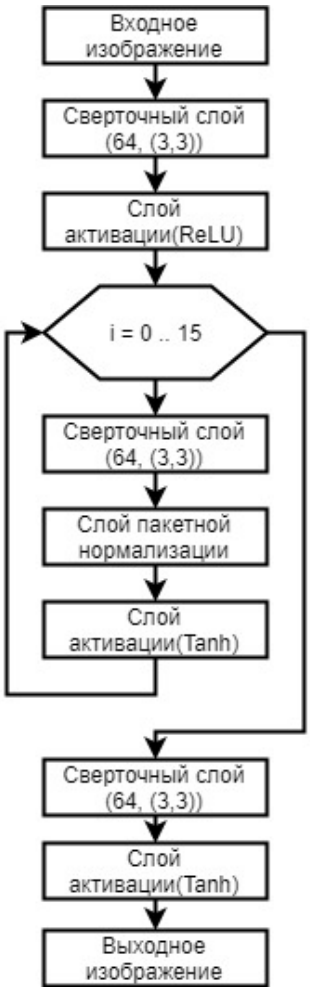

Рис. 8. Структура работы сверточной сети

Инструменты разработки программного средства

Python - интерпретируемый язык программирования. С одной стороны, это позволяет значительно упростить отладку программ, с другой - обусловливает сравнительно низкую скорость выполнения. В Python не надо заранее объявлять тип переменной, что очень удобно при разработке.

Python поддерживает автоматическую сборку мусора, что гарантирует отсутствие утечек памяти.

Программа, написанная на Python, будет функционировать одинаково вне зависимости от того, в какой операционной системе она запущена.

PyCharm – это интеллектуальная интегрированная среда разработки Python с полным набором средств для эффективной разработки. Выпускается в двух вариантах – бесплатная версия PyCharm Community Edition и поддерживающая больший набор возможностей PyCharm Professional Edition. PyCharm выполняет инспекцию кода, автодополнение, в том числе основываясь на информации, полученной во время исполнения кода, навигацию по коду, обеспечивает множество видов рефакторинга [9]. Community Edition также отличается открытостью исходного кода.

Keras - открытая нейросетевая библиотека, написанная на языке Python. Она представляет собой надстройку над фреймворками Deeplearning4j, TensorFlow и Theano и нацелена на оперативную работу с сетями глубинного обучения, при этом спроектирована так, чтобы быть компактной, модульной и расширяемой [4].

Она была создана как часть исследовательских усилий проекта ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System).

Keras предоставляет высокоуровневый, более интуитивный набор абстракций, который делает простым формирования нейронных сетей независимо от используемой библиотеки научных вычислений.

TensorFlow - открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов [5]. Применяется как для исследований, так и для разработки собственных продуктов Google. Основное API для работы с библиотекой реализовано для Python, также существуют реализации для C++, Haskell, Java и Go.

Применение инструментов разработки и библиотек позволяет вносить существенные изменения в алгоритм работы без значительных изменений кода приложения, позволяя направить освободившиеся ресурсы на решение задач обучения.

Обучение нейронной сети с использованием графического процессора

Библиотека машинного обучения TensorFlow может использовать два типа устройств – центральные процессоры (ЦП) и графические процессоры (ГП). Обучение на графических процессорах обеспечивает значительный прирост производительности, но в настоящий момент доступно только на ГП, использующих ядро NVIDIA.

Для обучения нейронной сети на ЦП достаточно установить пакет tensorflow при помощи пакетного менеджера pip. Стоит заметить, что Keras, так как является библиотекой более высокого уровня, не сможет работать без установленной TensorFlow (либо Theano, но в рамках данной работы этот случай не рассматривается).

Для обучения сети на ГП необходимо произвести следующие действия:

- проверить совместимость ГП с набором инструментов cuDNN [10];
- установить CUDAToolkit версии 8.0 [10], важно помнить, что для его работы необходимо наличие на компьютере IDE VisualStudio версии от 2015 и новее;
- установить cuDNN версии 5.1 [10].
- установить пакет tensorflow-gpu при помощи пакетного менеджера pip.

Если вышеперечисленные операции были произведены корректно, в дальнейшем ГП будет использоваться по умолчанию.

Использование ГП позволяет в разы увеличить скорость обучения. Для сравнения, в данной работе эпоха обучения на 200 000 изображениях с валидационной выборкой из 20 000 изображений занимает порядка 50-55 минут, в то время как эпоха с использованием ГП проходит за 12-15 минут (в качестве ГП использовался чип NVIDIA GTX 960, ЦП – Intel Core i5 6500K).

Непосредственно по завершению обучения можно производить тестирование нейронной сети.

Тестирование работоспособности программного средства

Начальной точкой работы с программным средством является загрузка обучающих (каталог train) и валидационных (каталог test) данных в виде изображений PNG размером 64*64 пикселя. После запуска средство упакует эти данные в массивы numpy, сохранит в файловой системе и произведет сжатые данные из исходных.

Изображения для анализа помещаются в директорию input (в рамках данной работы исходные изображения также были сохранены для визуального анализа эффективности работы алгоритма). Так как целью данной работы являлось изучение возможности восстановления потерянной при сжатии информации как таковой, алгоритм был реализован только для изображений размером 64*64 пикселя (тем не менее текущую реализацию теоретически возможно использовать для обработки изображений больших разрешений, предварительно разбив их на блоки).

По завершению обучения сети результаты работы программы будут отображены в окне SciView IDE PyCharm (рис. 9, 10).

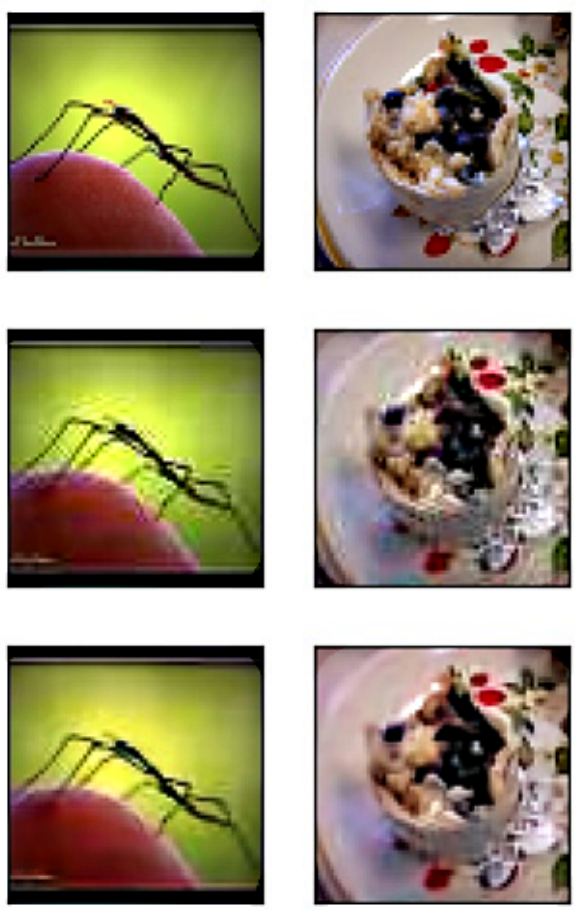

Рис. 9. Примеры обработанных сетью изображений (верхний ряд – исходное изображение, средний – сжатое, нижний – обработанное)

Рис. 9 и 10 подтверждают работоспособность данного алгоритма. Стоит отметить, что в случае с изображениями с большим количеством небольших деталей заметно появление слабого размытия, в то время на изображениях с ярко выраженным фоном (рис. 9 слева и рис. 10 справа) конечный результат сопоставим с исходными данными.

Заключение

Разработанное приложение подтвердило на практике возможность восстановления исходных данных в изображениях, сжатых lossy-кодеком JPEG. Так как приложение разработано с применением модульного подхода, в дальнейшем оно может быть модифицировано для

обработки изображений большего разрешения. Важно заметить, что, несмотря на эффективность глубокого обучения, необходимость использования большого объема обучающих данных накладывает некоторые ограничения, так как для практического применения модель необходимо будет тренировать на гораздо более крупных массивах данных [11, 12]. Важно также учитывать то, что в случае отсутствия в обучающих и валидационных выборках изображений, сжатыми различными реализациями кодека JPEG с различными степенями сжатия, присутствует риск т.н. перенасыщения модели, что приведет к значительному снижению эффективности.

Рис. 10. Примеры обработанных сетью изображений (аналогично рис. 9)

Литература

1. Twitter pays up to $150M for Magic Pony Technology, which uses neural networks to improve images Режим доступа: https://techcrunch.com/2016/06/20/ (дата обращения 20.02.2018).
2. Tools to inspire imagination & creativity Режим доступа: http://akvis.com/en/noise-buster/index.php (дата обращения 20.02.2018).
3. Substantially increase JPEG image quality with Topaz DeJPEG Режим доступа: http://www.topazlabs.com/dejpeg (дата обращения 20.02.2018).
4. Keras: The Python Deep Learning library Режим доступа: http://keras.io (дата обращения 20.02.2018).
5. An open-source machine learning framework for everyone Режим доступа: https://www.tensorflow.org/(дата обращения 20.02.2018).
6. Ха Л. М. Свёрточная нейронная сеть для решения задачи классификации // Труды МФТИ. 2016. № 3. С. 91.
7. Rawat W., Wang Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review // MIT Press - Neural Computation. 2017. № 29. С. 2352.
8. Kalinin A., Podvalny S., Soldatov E. Synthesis of a complex control system on the basis of neural networks // Intelligent Engineering Systems Through Artificial Neural Networks. 2003. 13, pp. 157-162.
9. PyCharm — интеллектуальная Python IDE Режим доступа: https://jetbrains.ru/products/pycharm/ (дата обращения 20.02.2018).
10. Nvidia Developer Режим доступа: https://developer.nvidia.com/ (дата обращения 20.02.2018).
11. Программная реализация конструктора нейронных сетей / Е.А. Ганцева, В.Ф. Барабанов, Н.И. Гребенникова, Д.С. Болдырев // Вестник Воронежского государственного технического университета. 2017. Т. 13. № 6. С. 25-31.
12. Анализ результатов экспериментальных исследований с использованием конструктора нейронных сетей / Е.А. Ганцева, В.Ф. Барабанов, Н.И. Гребенникова, Д.С. Болдырев // Вестник Воронежского государственного технического университета. 2017. Т. 13. № 6. С. 38-44.

Поступила 22.02.2018; принята к публикации 14.05.2018

Информация об авторах

Барабанов Владимир Федорович - д-р техн. наук, профессор, Воронежский государственный технический университет (394026, Россия, г. Воронеж, Московский проспект, 14), e-mail: bvf@list.ru

Гребенникова Наталия Ивановна – канд. техн. наук, доцент, Воронежский государственный технический университет (394026, Россия, г. Воронеж, Московский проспект, 14), e-mail: g-naty@yandex.ru

Кенин Сергей Леонидович – канд. техн. наук, руководитель проектов, Atos (394026, Россия, г. Воронеж, пр. Труда, 65), e-mail: sergey.kenin@atos.net

Юров Дмитрий Анатольевич – магистрант, Воронежский государственный технический университет (394026, Россия, г. Воронеж, Московский проспект, 14), e-mail: yurov.dmitry.a@gmail.com

SOFTWARE IMPLEMENTATION OF APPLICATION USING NEURAL NETWORK TO RESTORE IMAGES AFTER LOSSY COMPRESSION

V.F. Barabanov[1], N.I. Grebennikova[1], S.L. Kenin[2], D.A. Yurov[1]

[1]Voronezh State Technical University, Voronezh, Russia
[2]Atos, Voronezh, Russia

Abstract: the paper proposes a technology of image restoration, based on the use of neural network training. It allows to remove compression artifacts from the images processed by the JPEG codec that uses compression algorithms with losses. Neural networks are not programmed, unlike traditional software, they are trained. Learning capability is one of the main advantages of neural networks over traditional algorithms. Technically, training consists in finding the coefficients of connections between neurons. In the learning process, the neural network is able to identify complex relationships between input and output data, as well as perform generalization. This means that if the training is successful, the network will be able to return the correct result based on the data that was not in the training sample, as well as incomplete, partially distorted data. The structure and functions of the software tool for image restoration using deep neural networks are proposed. On the basis of the proposed structure, a library of methods and an application were developed to prepare the neural network for operation and to test it on non-synthetic copies of compressed images. Considerable attention is paid to the consideration of the most relevant architectures of neural networks for this task and libraries that simplify their implementation. The results of the application were obtained and analyzed

Key words: autoencoder, deep learning, convolutional neural network, Keras

References

1. “Twitter pays up to $150M for Magic Pony Technology, which uses neural networks to improve images”, available at: https://techcrunch.com/2016/06/20/

2. “Tools to inspire imagination & creativity”, available at: http://akvis.com/en/noise-buster/index.php.

3. “Substantially increase JPEG image quality with Topaz DeJPEG”, available at: http://www.topazlabs.com/dejpeg

4. “Keras: The Python Deep Learning library”, available at: http://keras.io.

5. “An open-source machine learning framework for everyone”, available at: https://www.tensorflow.org/.

6. Ha L.M. “Convolutional neural network for solving classification problem”, *Proceedings of MIPT (Trudy MFTI)*, 2016, no. 3, pp. 91.

7. Rawat W., Wang Z. “Deep Convolutional Neural Networks for Image Classification: a Comprehensive Review”, *MIT Press - Neural Computation*, 2017, no. 29, pp. 2352.

8. Kalinin A., Podvalny S., Soldatov E. “Synthesis of a complex control system on the basis of neural networks”, *Intelligent Engineering Systems Through Artificial Neural Networks*, 2003, 13, pp. 157-162.

9. “PyCharm-intelligent Python IDE”, available at: https://jetbrains.ru/products/pycharm/.

10. “Nvidia Developer”, available at: https://developer.nvidia.com/.

11. Gantseva E.A., Barabanov V.F., Grebennikova N.I., Boldyrev D.S. “Software implementation of the neural network designer”, *The Bulletin of Voronezh State Technical University (Vestnik Voronezhskogo gosudarstvennogo tekhnicheskogo universiteta)*, 2017, vol. 13, no. 6, pp. 25-31.

12. Gantseva E.A., Barabanov V.F., Grebennikova N.I., Boldyrev D.S. “Analysis of the results of experimental studies using a neural network designer”, *The Bulletin of Voronezh State Technical University (Vestnik Voronezhskogo gosudarstvennogo tekhnicheskogo universiteta),* 2017, vol. 13, no. 6, pp. 38-44.

Submitted 22.02.2018; revised 14.05.2018

Information about the authors

Vladimir F. Barabanov, Dr. Sc. (Technical), Professor, Voronezh State Technical University (14 Moskovskiy prospekt, Voronezh 394026, Russia), e-mail: bvf@list.ru

Natalia I. Grebennikova, Cand. Sc. (Technical), Associate Professor, Voronezh State Technical University (14 Moskovskiy prospekt, Voronezh 394026, Russia), e-mail: g-naty@yandex.ru

Sergey L. Kenin, Cand. Sc. (Technical), Project Manager, Atos (65 Prospekt Truda, Voronezh 394026, Russia), e-mail: sergey.kenin@atos.net

Dmitriy A. Yurov, MA, Voronezh State Technical University (14 Moskovskiy prospekt, Voronezh 394026, Russia), e-mail: yurov.dmitry.a@gmail.com