ISSN 2782-2001
Системы анализа и обработки данных
том 84, № 4, 2021, с. 7–18

http://journals.nstu.ru/vestnik
Analysis and data processing systems
Vol. 84, No. 4, 2021, pp. 7–18

ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

INFORMATICS, COMPPUTER ENGINEERING AND MANAGEMENT

УДК 004.8

DOI: 10.17212/2782-2001-2021-4-7-18

Метод поиска и разметки артефактов на изображениях с использованием алгоритмов детекции и сегментации*

А.М. КИТЕНКО

199178, РФ, г. Санкт-Петербург, 14-я линия В. О., Федеральное государственное бюджетное учреждение науки «Санкт-Петербургский федеральный исследовательский центр Российской академии наук» (СПб ФИЦ РАН)

kitenko.andrey@gmail.com

В работе исследуется возможность применения нейронных сетей для попиксельного выделения целевых артефактов на разных типах документов. Многочисленные типы нейронных сетей часто применяются для обработки документов – от анализов текста до выделения определенных зон, где может содержаться искомая информация. Однако сегодня нет совершенных систем по обработке документов, которые могут работать автономно, компенсируя человеческие ошибки, появляющиеся в процессе работы из-за стресса, усталости и многих других причин. В настоящей работе акцент сделан на поиске и выделении целевых артефактов на чертежах в условиях малого количества первоначальных данных. Предложенный метод поиска и выделения артефактов на изображении состоит из двух основных частей: детектирования и семантической сегментации детектируемой области. В основе метода лежит обучение с учителем на размеченных данных для двух сверточных нейронных сетей. Первая сверточная сеть используется для детектирования области с артефактом, в данном примере за основу была взята YoloV4. Для семантической сегментации применяется архитектура U-Net, где основой является предобученная Efficientnetb0. При комбинации этих нейронных сетей были достигнуты хорошие результаты даже для выделения определенных рукописных текстов, без использования для этого специфики построения нейросетевых моделей для распознавания текстов. Данный метод может применяться для поиска и выделения артефактов в больших наборах данных, при этом сами артефакты могут быть разными по форме, цвету и типу, при этом они могут располагаться в разных местах изображения, иметь или не иметь пересечения с другими объектами.

Ключевые слова: искусственный интеллект, семантическая сегментация, компьютерное зрение, распознавание образов, нейронные сети, машинное обучение, глубокое обучение, детекция объектов

---

* *Статья получена 12 мая 2021 г.*

Представленные результаты исследований были выполнены в рамках бюджетной темы № 0073-2019-0005 (2019–2021).

ВВЕДЕНИЕ

Сегодня нейронные сети активно используются для автоматизации многих процессов в нашей повседневной жизни, тем самым избавляя людей от рутинной работы. Во всём мире в больших количествах используются разные типы документов – от текстовых файлов до чертежей, которые необходимо обрабатывать для поиска интересующей информации. На данный момент многие документы хранятся в электронном формате, это позволяет нам с использованием специализированного программного обеспечения существенно быстрее обрабатывать документы, а иногда и полностью автоматизированно работать с ними, тем самым уменьшая вероятность человеческой ошибки, связанной с усталостью, нежеланием работать и многими другими обстоятельствами, которые могут повлиять на работу человека.

В настоящей работе был опробован метод поиска подписей и рукописного текста на изображениях с чертежами с использованием комбинации сверточных нейронных сетей. Целью настоящей работы было найти общий подход для успешной работы комбинации двух сверточных нейронных сетей.

Основная проблема поиска и выделения подписей и рукописного текста на чертежах заключается в том, что данные артефакты на чертежах достаточно близки по форме и цвету к остальной информации, которая присутствует на чертеже. Поэтому сверточным нейронным сетям сложно выделить общие признаки для артефактов относительно остальной информации, которая присутствует на изображении. Также проблемой является входное разрешение изображения. Входной чертеж может иметь разрешение (730 × 1540) или (12 630 × 14 620) пикселей. Таким образом, артефакты могут быть одинаковыми по общей форме (например, подпись), однако сильно отличаться по размеру.

Для решения данной задачи рассматривались разные подходы для обучения сверточных нейронных сетей с использованием разных обучающих выборок – от ручной разметки данных до автоматической вставки целевых артефактов в определенную область чертежа.

1. МЕТОД АНАЛИЗА ИЗОБРАЖЕНИЙ

В алгоритме поиска артефактов было задействовано два типа нейронных сетей. Первая нейронная сеть использовалась для детекции артефактов, вторая нейронная сеть – для семантической сегментации детектируемой области. На рис. 1 показана работа алгоритма поиска артефактов.

На вход алгоритма подается RGB-изображение или в оттенках серого, которое проходит этап предобработки. Входное разрешение изображения масштабируется на среднее значение всей обучающей выборки, таким образом нормализуется пространственная информация на изображение относительно всей обучающей выборки. После изменения масштаба изображение делится на квадраты с шагом перекрытия, равным половине квадрата. Разрешение каждого квадрата составляет 1504 × 1504 пикселей. Таким образом после этапа предобработки в детектор поступает четырехмерный тензор [B, Y, X, C], где

- B – размер партии (batch size);
- Y – высота изображения в пикселях;

- X – ширина изображения в пикселях;
- C – цветное изображение или в оттенках серого.

На выходе из детектора формируется список значений с предсказанными координатами артефактов. На следующем этапе осуществляется копирование участков изображений по полученным координатам из детектора. Далее снова формируется четырехмерный тензор [B, Y, X, C], однако изображения масштабируются на меньшее ближайшее кратное значение 32, после этого изображения нормализуются от нуля до единицы.

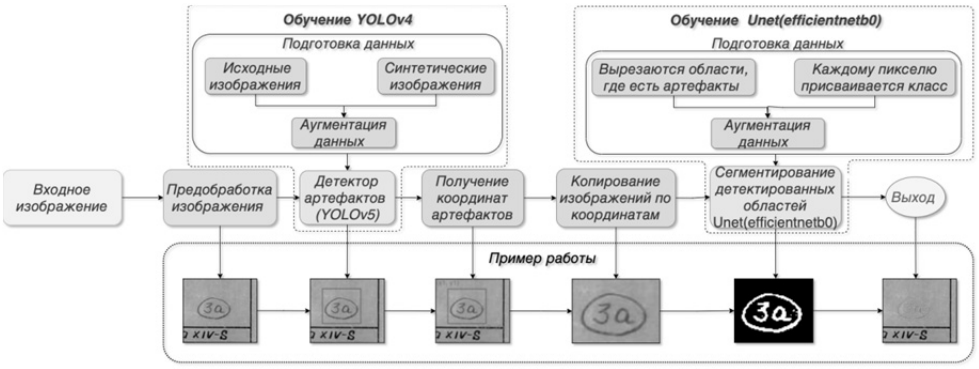

Рис. 1. Алгоритм анализа изображения

Fig. 1. An image analysis algorithm

Сформированный четырехмерный тензор подается в нейронную сеть для семантической сегментации, после последовательной обработки на выходе получается четырехмерный тензор [B, Y, X, N], где

- B – размер партии (batch size);
- Y – высота изображения в пикселях;
- X – ширина изображения в пикселях;
- N – количество классов обучаемой модели.

Все числа в выходном тензоре нормированы от нуля до единицы и могут быть интерпретированы как вероятности принадлежности к тому или иному классу, которые присутствуют в обучающей выборке. Таким образом, используя пороговую величину для выходного тензора нейронной сети, можно получить бинарную маску, где значения меньше пороговой величины будут равны нулю, иначе единице.

На заключительном этапе разрешения полученных масок масштабируются к оригинальным значениям копированных изображений. Таким образом по данным предсказаниям выделяются артефакты на оригинальном изображение.

На первом этапе обработки изображения используется нейронная сеть для детекции артефактов, в настоящей работе использовалась YoloV4 [1], так как данная архитектура сети показывает великолепные результаты по скорости и точности на датасете Microsoft COCO [2] относительно таких архитектур, как Google TensorFlow EfficientDet [3], FaceBook Detectron RetinaNet [4], MaskRCNN [5] и многие другие. Архитектура нейронной сети была изменена, чтобы учесть особенности размеров входных изображений. В первых трех блоках сверточных слоев параметр *stride* был изменен с 2 на 4, так как в основном детектируемые объекты имеют средние и большие размеры относи-

тельно входного изображения. Эти изменения критично не повлияли на оригинальную архитектуру, поэтому можно считать, что используется оригинальная архитектура YoloV4. Также был применен встроенный метод для подбора размеров якорных боксов с применением алгоритма k-means++ [6], чтобы они максимально соответствовали искомым артефактам.

На втором этапе обработки изображения использовалась нейронная сеть для семантической сегментации изображения. На первом этапе сегментации происходит уменьшение пространственного разрешения входного изображения с увеличением количества фильтрационных слоев, затем выполняется постепенное увеличение изображения, которое дополняется информацией со слоев декодера, с применением операции конкатенации [7].

Для семантической сегментации детектируемой области использовалась кастомизированная архитектура U-Net [9] с предобученнной Efficientnetb0 [10] для извлечения искомых признаков артефактов, на рис. 2 представлена схема архитектуры U-Net.

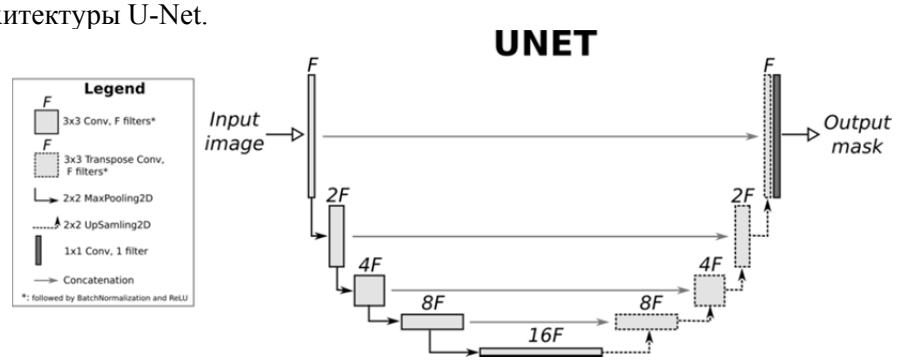

Рис. 2. Пример архитектуры U-Net [8]

Fig. 2. An example of the U-Net architecture [8]

Таким образом, архитектура сети состоит из сужающегося пути (левая часть архитектуры), где каждый блок соответствует архитектуре сети Efficientnetb0. Каждый блок на пути расширения (правая часть архитектуры) состоит из двух последовательных комбинаций (Conv2D [11], BatchNormalization [12], Relu [13]), в конце применяется UpSampling2D [14]. На рис. 3 представлена схема блока расширения.

Рис. 3. Блок расширения в U-Net(Efficientnetb0)

Fig. 3. An expansion unit in the U-Net(Efficientnetb0)

Efficientnetb0 относится к масштабированным нейронным сетям и занимает начальное положение в иерархии Efficientnet. Это означает, что данная сеть самая маленькая в своем семействе, и относительно данной сети происходит масштабирование до сетей Efficientnetb1 – Efficientnetb7. До появления семейства нейронных сетей Efficientnet существовали другие семейства сверточных нейронных сетей (Res-Net [15], Densenet [16] и многие другие), при масштабировании которых изменяли глубину, ширину или разрешение для увеличения точности. Однако до появления семейства Efficientnet способы масштабирования сетей не были хорошо изучены. Таким образом, в статье,

посвященной Efficientnet [10], на основе эмпирического исследования был представлен эффективный метод комбинированного масштабирования для нейронных сетей по глубине, ширине и разрешению.

При разработке архитектуры Efficientnetb0 авторы использовали метод многоцелевого поиска нейронной архитектуры [17], который при оптимизации был нацелен на точность и FLOPS. Основным структурным блоком в Efficientnetb0 является мобильное инвертированное узкое место MBConv [17, 18], к которому также была добавлена оптимизация сжатия и возбуждения.

2. ПОДГОТОВКА ОБУЧАЮЩИХ ВЫБОРОК И ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ

Подготовка обучающих выборок была разделена на пять основных этапов, которые отображены на рис. 4. На первом этапе подготовки обучающих выборок был выполнен анализ 179 целевых изображений, на которых присутствуют искомые артефакты. В настоящей работе большую значимость имеет местоположение артефактов. После проведения анализа целевых изображений был сделан вывод, что искомые артефакты в основном присутствуют рядом с таблицей чертежа, поэтому при формировании обучающей выборки важно учитывать данное свойство целевых изображений. В процессе анализа дополнительно были учтены формы, размеры и цвета артефактов для генерации синтезированных данных.

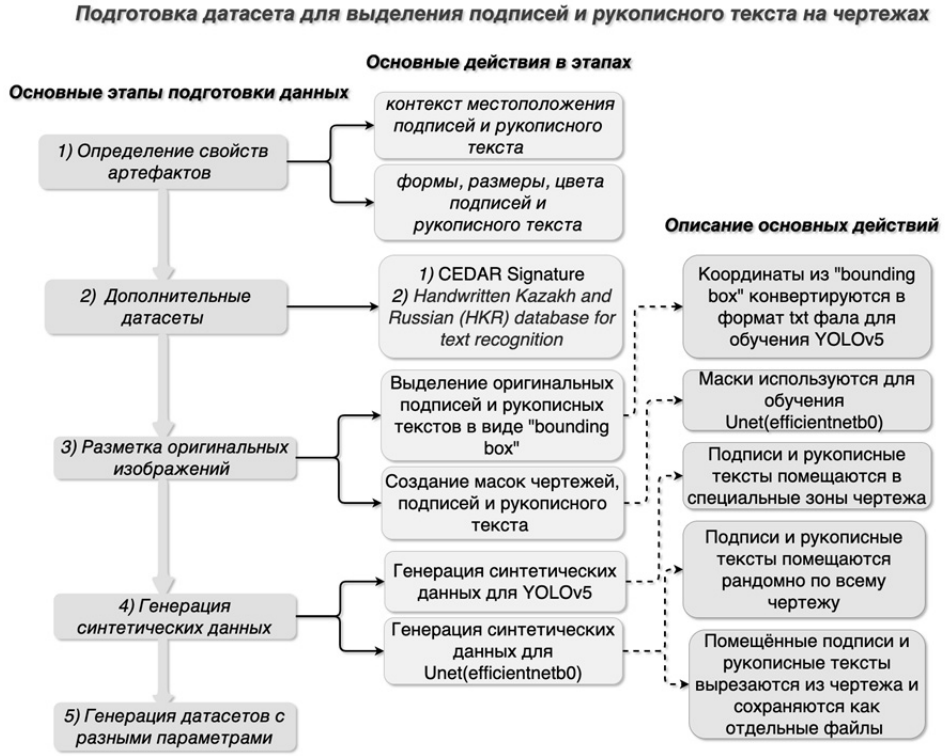

Рис. 4. Этапы подготовки обучающих выборок

Fig. 4. Stages of training samples preparation

На втором этапе подготовки обучающих выборок был проведен поиск дополнительных изображений и датасетов, которые могли бы использоваться при синтезировании новых обучающих выборок. В результате поиска были дополнительно скачаны 200 изображений чертежей, в которых присутствовал разный контекст информации – от текста до сложных форм деталей. Также были скачаны датасеты с подписями [19] и русскоязычными словами [20]. В результате для синтезирования новых датасетов использовались:

- 379 изображений чертежей;
- 4424 изображения подписей;
- 256 308 изображений русскоязычных слов и словосочетаний.

На третьем этапе подготовки обучающих выборок выполнялась разметка подписей и рукописного текста на изображениях чертежах. Для разметки использовался бесплатный веб-инструмент для аннотации изображений открытым исходным кодом CVAT [21]. Этот инструмент позволяет экспортировать аннотированные изображения во множество форматов, которые могут использоваться для обучения нейронных сетей. На первом этапе разметки выделялись зоны в виде прямоугольных областей, на которых присутствовали подписи или рукописные тексты. Затем размеченные области экспортировались в двух форматах:

- текстовый файл формата YOLO [22];
- сегментационные маски [22].

Полученные текстовые файлы использовались в дальнейшем при синтезировании новых обучающих выборок. В результате разметки были получены точные координаты подписей и рукописных надписей на чертежах, появилась возможность точно выделять границы артефактов посредством использования порога. Таким образом, темные пиксели относились к артефактам, а светлые – к фону.

На четвертом этапе подготовки обучающих выборок была выполнена генерация синтетических данных, которые удовлетворяли результатам анализа с первого этапа подготовки. Генерация синтетических данных проводилась отдельно для модели детекции и сегментации. Для детекции одним из важнейших факторов являлось местоположение артефакта, поэтому при генерации изображений данный параметр учитывался и артефакт помещался в специальную область чертежа, однако также были сформированы обучающие выборки, где артефакт помещался рандомно для сравнения точности работы алгоритма. При генерации синтетических данных в цикле перебирались все 379 изображений, далее в каждое изображение помещались подписи и рукописные тексты из скачанных датасетов в разделе 2, подписи и тексты брались случайным образом. Полученные координаты артефактов при генерации были записаны в одноименные текстовые файлы, которые были получены ранее.

При генерации синтетических данных для семантической сегментации изображений артефакты размещались случайным образом по всему чертежу, так как в этом случае главная цель – способность нейронной сети распознавать образ подписи в разных ситуациях. Как и при формировании синтетических данных для детекции, подписи и рукописные тексты брались случайным образом, однако перед помещением этих артефактов в область чертежа необ-

ходимо было извлечь маски для них. Точные маски, как и ранее, извлекались посредством порога, светлые пиксели относились к фону, темные – к артефактам. При вставке артефакта соблюдалось только одно правило: новый артефакт не должен пересекаться с другими артефактами, которые уже присутствовали на чертеже. Далее из полученных изображений вырезались прямоугольные области, на которых присутствовали артефакты. Таким образом было получено большое разнообразие данных, которые группировались по разным параметрам в следующем пункте.

На пятом этапе подготовки были сформированы разные обучающие выборки.

1. Сгенерированные данные для детекции.

- Первый датасет – данные помещались в специальную область (сгенерировано 3000 изображений).
- Второй датасет – данные помещались по всему чертежу, но только в свободные зоны (сгенерировано 1000 изображений).
- Третий датасет – данные помещались по всему чертежу и имели пересечения с другими объектами (сгенерировано 1000 изображений).
- Четвертый датасет – смешанные данные из первого, второго и третьего датасета (3000 изображений).

2. Сгенерированные данные для семантической сегментации.

- Первый датасет – данные помещались по всему чертежу, но только в свободные зоны (сгенерировано 1000 изображений).
- Второй датасет – данные помещались по всему чертежу и имели пересечения с другими объектами (сгенерировано 1000 изображений).
- Третий датасет – смешанные данные из первого и второго датасета (2000 изображений).

Обучение YoloV4 проводилось стандартными средствами, которые присутствуют в фреймворке Darknet [23]. Каждая сформированная обучающая выборка была поделена на тренировочные и валидационные части. В тренировочной присутствовало 80 % изображений, в валидационной – 20 % от общей части обучающей выборки.

В ходе проведенного исследования было выяснено, что наилучшие результаты модель показывает, если использовать для аугментации данных следующие средства:

- Saturation,
- Exposure,
- Hue,
- Mixup,
- Mosaic,
- Blur,
- Cutmix.

Далее модель обучалась на сформированных датасетах, результаты оценивались на валидации с использованием метрики mAP (mean average precision) [24].

Обучение U-Net(Efficientnetb0) осуществлялось с использованием высокоуровневого фреймворка Keras [25]. Обучающие выборки также были поделены на тренировочную и валидационные части в таких же пропорциях, как и

для обучения YoloV4. Как и при обучении YoloV4, была подобрана аугментация данных с использованием данных средств:

- Blur,
- ColorJitter,
- GaussNoise,
- HueSaturationValue.

В качестве алгоритма оптимизации нейронной сети использовался Adam [26] с шагом обучения 0.001 [27]. Для оценки потерь использовался критерий Жаккара [28]:

$$L(A,\ B) = 1 - \frac{A \cap B}{A \cup B}.$$

Затем модель U-Net(Efficientnetb0) обучалась на сформированных выборках, результаты оценивались на валидации с использованием метрики mIoU (Mean Intersection-Over-Union) [29].

3. ДЕМОНСТРАЦИЯ РАБОТЫ АЛГОРИТМА ПОИСКА АРТЕФАКТОВ

Для демонстрации работы алгоритма было взято изображение с одним артефактом. Из рис. 5 видно, что комбинированные нейронные сети успешно справились с поиском и попиксельным выделением артефакта.

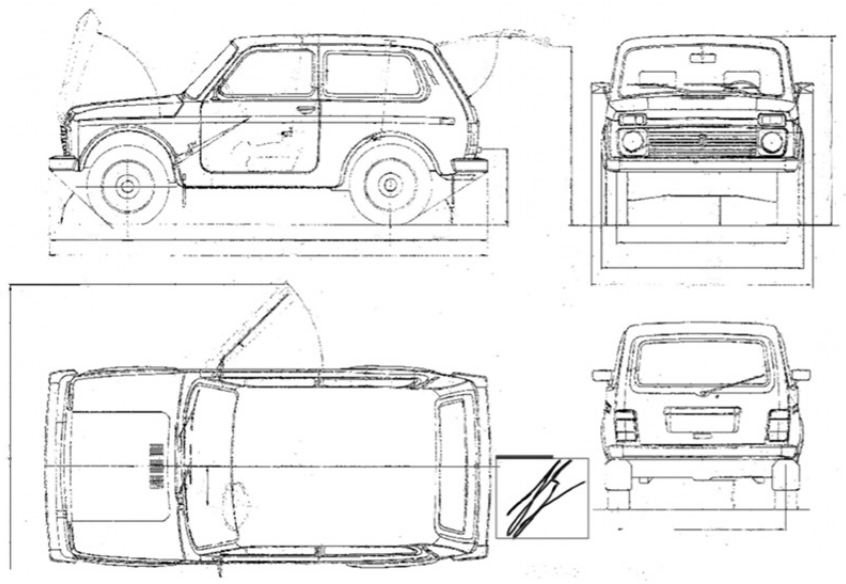

Рис. 5. Демонстрация работы алгоритма

Fig. 5. Demonstration of the algorithm

Результат детекции отображен в виде прямоугольной области. Результат семантической сегментации детектируемой области выделен толстой линией внутри прямоугольной области.

ЗАКЛЮЧЕНИЕ

Представленный метод поиска и попиксельного выделения артефактов с комбинированием двух нейронных сетей показал себя хорошо в реальных тестах. Применяя на первой стадии нейронную сеть для детекции и локализации искомых объектов, можно упростить задачу на второй стадии, где используется нейронная сеть для семантической сегментации, на вход которой подается участок с изображением, в котором вероятность нахождения артефакта велика, в отличие от остальных областей изображений.

Однако следует заметить, что предложенный алгоритм для решения проблемы большого отличия входных разрешений изображений не идеально подходит во всех условиях. Поэтому в данном случае необходимо доработать общий метод так, чтобы не менять разрешения входных изображений. Также следует доработать архитектуру нейронной сети так, чтобы использовать одну общую часть для извлечения признаков. Таким образом, скорость работы алгоритма уменьшится при повышении общей скорости работы нейронной сети, что может быть очень важным при работе с большим объемом данных.

СПИСОК ЛИТЕРАТУРЫ

1. *Bochkovskiy A.*, *Wang C.-Y., Liao H.-Y.M.* YOLOv4: Optimal speed and accuracy of object detection. – arXiv preprint arXiv:2004.10934 [cs, eess]. – 2020.

2. Microsoft COCO: common objects in context / T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C.L. Zitnick, P. Dollár. – arXiv preprint arXiv:1405.0312 [cs]. – 2015.

3. *Tan M., Pang R., Le Q.V.* EfficientDet: scalable and efficient object detection. – arXiv preprint arXiv:1911.09070 [cs, eess]. – 2020.

4. Focal loss for dense object detection / T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár. – arXiv preprint arXiv:1708.02002 [cs]. – 2018.

5. Mask R-CNN / K. He, G. Gkioxari, P. Dollár, R. Girshick. – arXiv preprint arXiv:1703.06870 [cs]. – 2018.

6. *Makarychev K.*, *Reddy A., Shan L.* Improved guarantees for k-means++ and k-means++ parallel. – arXiv preprint arXiv:2010.14487 [cs]. – 2020.

7. tf.keras.layers.Concatenate // TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Concatenate?hl=ru (accessed: 29.11.2021).

8. Image segmentation with Monte Carlo Dropout UNET and Keras // 42: A blog on A.I. – 2019. – 30 October. – URL: https://nchlis.github.io/2019_10_30/page.html (accessed: 29.11.2021).

9. *Ronneberger O.*, *Fischer P., Brox T.* U-Net: convolutional networks for biomedical image segmentation. – arXiv preprint arXiv:1505.04597 [cs]. – 2015.

10. *Tan M.*, *Le Q.V.* EfficientNet: rethinking model scaling for convolutional neural networks. – arXiv preprint arXiv:1905.11946 [cs, stat]. – 2020.

11. tf.keras.layers.Conv2D | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D (accessed: 29.11.2021).

12. tf.keras.layers.BatchNormalization | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization?hl=ru (accessed: 29.11.2021).

13. tf.keras.activations.relu | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu?hl=ru (accessed: 29.11.20211).

14. tf.keras.layers.UpSampling2D | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/UpSampling2D?hl=ru (accessed: 29.11.2021).

15. Deep Residual Learning for Image Recognition / K. He, X. Zhang, Sh.Ren, J. Sun. – arXiv preprint arXiv:1512.03385 [cs]. – 2015.

16. Densely connected convolutional networks / G. Huang, Z. Liu, Maaten L. van der, K.Q. Weinberger . – arXiv preprint arXiv:1608.06993 [cs]. – 2018.

17. MnasNet: platform-aware neural architecture search for mobile / M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le. – arXiv preprint arXiv:1807.11626 [cs]. – 2019.

18. MobileNetV2: inverted residuals and linear bottlenecks / M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. Chen. – arXiv preprint arXiv:1801.04381 [cs]. – 2019.

19. Papers with Code – CEDAR Signature Dataset. – URL: https://paperswithcode.com/dataset/cedar-signature (accessed: 29.11.2021).

20. *Abdallah A., Hamada M., Nurseitov D.* Attention-based fully gated CNN-BGRU for Russian handwritten text // Journal of Imaging. – 2020. – Vol. 6, N 12. – P. 141.

21. GitHub – openvinotoolkit/cvat: powerful and efficient computer vision annotation tool (CVAT). – URL: https://github.com/openvinotoolkit/cvat (accessed: 29.11.2021).

22. *Pokhrel S.* Image data labelling and annotation – everything you need to know. – URL: https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1 (accessed: 29.11.2021).

23. GitHub – AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO – Neural Networks for Object Detection (Windows and Linux version of Darknet). – URL: https://github.com/AlexeyAB/darknet (accessed: 29.11.2021).

24. *Yohanandan S.* mAP (mean Average Precision) might confuse you! – URL: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 (accessed: 29.11.2021).

25. Module: tf.keras | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras?hl=ru (accessed: 29.11.2021).

26. tf.keras.optimizers.Adam | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam?hl=ru (accessed: 29.11.2021).

27. *Zulkifli H.* Understanding learning rates and how it improves performance in deep learning. – URL: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (accessed: 29.11.2021).

28. Segmentation models Python API – segmentation models 0.1.2 documentation. – URL: https://segmentation-models.readthedocs.io/en/latest/api.html#losses (accessed: 29.11.2021).

29. tf.keras.metrics.MeanIoU | TensorFlow Core v2.7.0. – URL: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/MeanIoU?hl=ru (accessed: 29.11.2021).

Китенко Андрей Максимович, младший научный сотрудник, аспирант, Федеральное государственное бюджетное учреждение науки «Санкт-Петербургский федеральный исследовательский центр Российской академии наук» (СПб ФИЦ РАН). Основное направление научных исследований – глубокое обучение, компьютерное зрение, распознавание образов. E-mail: kitenko.andrey@gmail.com

Kitenko Andrey M., junior researcher, postgraduate student, St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS). His research interests are currently focused on deep learning, computer vision, and pattern recognition. E-mail: kitenko.andrey@gmail.com

DOI: 10.17212/2782-2001-2021-4-7-18

A method of searching and marking artifacts in images applying detection and segmentation algorithms[*]

A. KITENKO

Saint Petersburg, St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), 4th lin. V.I., 199178, Russian Federation

kitenko.andrey@gmail.com

Abstract

The paper explores the possibility of using neural networks to single out target artifacts on different types of documents. Numerous types of neural networks are often used for document processing, from text analysis to the allocation of certain areas where the desired information may be contained. However, to date, there are no perfect document processing systems that can work autonomously, compensating for human errors that may appear in the process of work due to stress, fatigue and many other reasons. In this work, the emphasis is on the search and selection of target artifacts in drawings, in conditions of a small amount of initial data. The proposed method of searching and highlighting artifacts in the image consists of two main parts, detection and semantic segmentation of the detected area. The method is based on training with a teacher on marked-up data for two convolutional neural networks. The first convolutional network is used to detect an area with an artifact, in this example YoloV4 was taken as the basis. For semantic segmentation, the U-Net architecture is used, where the basis is the pre-trained Efficientnetb0. By combining these neural networks, good results were achieved, even for the selection of certain handwritten texts, without using the specifics of building neural network models for text recognition. This method can be used to search for and highlight artifacts in large datasets, while the artifacts themselves may be different in shape, color and type, and they may be located in different places of the image, have or not have intersection with other objects.

Keywords: artificial intelligence, semantic segmentation, computer vision, pattern recognition, neural networks, machine learning, deep learning, object detection

REFERENCES

1. Bochkovskiy A., Wang C.-Y., Liao H.-Y.M. *YOLOv4: optimal speed and accuracy of object detection*. arXiv:2004.10934 [cs, eess], 2020.

2. Lin T.-Y., Maire M., Belongie S., Bourdev L., Girshick R., Hays J., Perona P., Ramanan D., Zitnick C.L., Dollár P. *Microsoft COCO: common objects in context*. arXiv:1405.0312 [cs], 2015.

3. Tan M., Pang R., Le Q.V. *EfficientDet: scalable and efficient object detection*. arXiv:1911.09070 [cs, eess], 2020.

4. Lin T.-Y., Goyal P., Girshick R., He K., Dollár P. *Focal loss for dense object detection*. arXiv:1708.02002 [cs], 2018.

5. He K., Gkioxari G. , Dollár P., Girshick R. *Mask R-CNN*. arXiv:1703.06870 [cs], 2018.

6. Makarychev K., Reddy A., Shan L. *Improved guarantees for k-means++ and k-means++ parallel*. arXiv:2010.14487 [cs], 2020.

7. tf.keras.layers.Concatenate | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Concatenate?hl=ru (accessed 29.11.2021).

8. Image segmentation with Monte Carlo Dropout UNET and Keras. *42: A blog on A.I*, 2019, 30 October. Available at: https://nchlis.github.io/2019_10_30/page.html (accessed 29.11.2021).

9. Ronneberger O., Fischer P., Brox T. *U-Net: convolutional networks for biomedical image segmentation*. arXiv:1505.04597 [cs], 2015.

[*] *Received 12 May 2021.*

The presented research results were carried out within the framework of budget topic No. 0073-2019-0005 (2019-2021).

10. Tan M., Le Q.V. *EfficientNet: rethinking model scaling for convolutional neural networks*. arXiv:1905.11946 [cs, stat], 2020.

11. tf.keras.layers.Conv2D | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D (accessed 29.11.2021).

12. tf.keras.layers.BatchNormalization | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization?hl=ru (accessed 29.11.2021).

13. tf.keras.activations.relu | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu?hl=ru (accessed 29.11.2021).

14. tf.keras.layers.UpSampling2D | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/UpSampling2D?hl=ru (accessed 29.11.2021).

15. He K., Zhang X., Ren Sh., Sun J. *Deep residual learning for image recognition*. arXiv:1512.03385 [cs], 2015.

16. Huang G., Liu Z., Maaten L. van der, Weinberger K.Q. *Densely connected convolutional networks*. arXiv:1608.06993 [cs], 2018.

17. Tan M., Chen B., Pang R., Vasudevan V., Sandler M., Howard A., Le Q.V. *MnasNet: platform-aware neural architecture search for mobile*. arXiv:1807.11626 [cs], 2019.

18. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L. *MobileNetV2: inverted residuals and linear bottlenecks*. arXiv:1801.04381 [cs], 2019.

19. *Papers with Code – CEDAR Signature Dataset*. Available at: https://paperswithcode.com/dataset/cedar-signature (accessed 29.11.2021).

20. Abdallah A., Hamada M., Nurseitov D. Attention-based fully gated CNN-BGRU for Russian handwritten text. *Journal of Imaging*, 2020, vol. 6, no. 12, p. 141.

21. *GitHub – openvinotoolkit/cvat: powerful and efficient computer vision annotation tool (CVAT)*. Available at: https://github.com/openvinotoolkit/cvat (accessed 29.11.2021).

22. Pokhrel S. *Image data labelling and annotation – everything you need to know*. Available at: https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1 (accessed 29.11.2021).

23. GitHub – AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO – Neural Networks for Object Detection (Windows and Linux version of Darknet). Available at: https://github.com/AlexeyAB/darknet (accessed 29.11.2021).

24. Yohanandan S. *mAP (mean Average Precision) might confuse you!* Available at: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 (accessed 29.11.2021).

25. Module: tf.keras | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras?hl=ru (accessed 29.11.2021).

26. tf.keras.optimizers.Adam | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam?hl=ru (accessed 29.11.2021).

27. Zulkifli H. *Understanding learning rates and how it improves performance in deep learning*. Available at: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (accessed 29.11.2021).

28. *Segmentation models Python API – segmentation models 0.1.2 documentation*. Available at: https://segmentation-models.readthedocs.io/en/latest/api.html#losses (accessed 29.11.2021).

29. tf.keras.metrics.MeanIoU | TensorFlow Core v2.7.0. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/MeanIoU?hl=ru (accessed 29.11.2021).

Для цитирования:

Китенко А.М. Метод поиска и разметки артефактов на изображениях с использованием алгоритмов детекции и сегментации // Системы анализа и обработки данных. – 2021. – № 4 (84). – С. 7–18. – DOI: 10.17212/2782-2001-2021-4-7-18.

For citation:

Kitenko A.M. Metod poiska i razmetki artefaktov na izobrazheniyakh s ispol'zovaniyem algoritmov detektsii i segmentatsii [A method of searching and marking artifacts in images applying detection and segmentation algorithms]. *Sistemy analiza i obrabotki dannykh = Analysis and Data Processing Systems*, 2021, no. 4 (84), pp. 7–18. DOI: 10.17212/2782-2001-2021-4-7-18.