ПОЖАРОВЗРЫВОБЕЗОПАСНОСТЬ/FIRE AND EXPLOSION SAFETY. 2023. Т. 32. № 5. С. 40–48
POZHAROVZRYVOBEZOPASNOST/FIRE AND EXPLOSION SAFETY. 2023; 32(5):40-48

НАУЧНАЯ СТАТЬЯ/RESEARCH PAPER

УДК 004.896

https://doi.org/10.22227/0869-7493.2023.32.05.40-48

Обнаружение очагов возгорания на технологических объектах с использованием сверточной нейронной сети

Андрей Александрович Евсиков, Илья Вадимович Самарин ✉

Российский государственный университет нефти и газа (национальный исследовательский университет) имени И.М. Губкина, г. Москва, Россия

АННОТАЦИЯ

Введение. Рассматривается решение задачи обнаружения очагов возгорания на технологических объектах в автоматическом режиме. Для этого выбран подход по созданию сверточной нейронной сети, способной работать на видеопотоке в реальном времени.
Цели и задачи. Целью данной работы является создание нейронной сети, способной обнаруживать пламя и дым на изображении с камер видеонаблюдения. Задачи исследования: выбор оптимальной архитектуры нейронной сети в соответствии с последними исследованиями в этой области; ускорение работы выбранной архитектуры с помощью методов квантования и прореживания фильтров
Методы. Рассматриваются различные архитектуры сверточных нейронных сетей, выполняющих задачу обнаружения объектов на изображении. Сравниваются их быстродействие и качество работы. Изучается архитектура YOLOv5, ее целевая функция, методы обучения и способы ускорения работы.
Результаты и их обсуждение. Показаны результаты обучения сверточной нейронной сети архитектуры YOLOv5 для задачи обнаружения пламени и дыма, а изменение результатов при применении методов ускорения нейронной сети. Определено, что использование таких методов ускорения, как квантование и фильтрация фильтров, позволяет значительно увеличить скорость работы нейронной сети, почти не потеряв в точности работы.
Выводы. Определена архитектура нейронной сети для обнаружения очага возгорания. На основе выбранной архитектуры обучена нейронная сеть, способная обнаруживать пламя и дым на изображении. Скорость ее работы дает возможность обрабатывать видеопоток в реальном времени без использования графического ускорителя.

Ключевые слова: компьютерное зрение; обнаружение пожаров; машинное обучение; обнаружение объектов в реальном времени; методы квантования и прореживания фильтров

Для цитирования: *Евсиков А.А., Самарин И.В.* Обнаружение очагов возгорания на технологических объектах с использованием сверточной нейронной сети // Пожаровзрывобезопасность/Fire and Explosion Safety. 2023. Т. 32. № 5. С. 40–48. DOI: 10.22227/0869-7493.2023.32.05.40-48

✉ *Самарин Илья Вадимович,* e-mail: ivs@gubkin.ru

Detection of fires at technological facilities using convolutional neural network

Andrey A. Evsikov, Ilya V. Samarin ✉

National University of Oil and Gas “Gubkin University”, Moscow, Russian Federation

ABSTRACT

Introduction. The article considers the solution of the problem of detecting fires at technological facilities in automatic mode. To solve this problem, an approach was chosen to create a convolutional neural network capable of operating on a real-time video stream.
Aims and Purposes. The aim of this work is to create a neural network capable of detecting flames and smoke in the image from CCTV cameras. The purposes of the work include: selection of the optimal architecture of the neural network in accordance with the latest research in this field; speeding up the chosen architecture using quantization and filter thinning techniques.
Methods. Different architectures of convolutional neural networks performing the task of detecting objects in an image are considered. Their performance and quality of work are compared. The YOLOv5 architecture, its target function, training methods and ways of speeding up work are considered.
Results and discussion. The paper shows the results of training a convolutional neural network of the YOLOv5 architecture for the task of flame and smoke detection, as well as how the results change when applying neural

network acceleration methods. It was determined that the use of such acceleration methods such as quantization and filter cleaning can significantly increase up the speed of the neural network, while almost no loss in accuracy of operation.

Conclusions. As a result of the conducted work, the architecture of the neural network was determined for performing the task of detecting a fire source. Based on the chosen architecture, a neural network was trained to detect flames and smoke in the image. The speed of its work allows to process video stream in real-time without using graphic accelerator.

Keywords: computer vision; fire detection; machine learning; real-time object detection; quantization and filter thinning methods

For citation: Evsikov A.A., Samarin I.V. Detection of fires at technological facilities using convolutional neural network. *Pozharovzryvobezopasnost/Fire and Explosion Safety*. 2023; 32(5):40-48. DOI: 10.22227/0869-7493.2023.32.05.40-48 (rus).

✉ *Ilya Vadimovich Samarin*, e-mail: ivs@gubkin.ru

Введение

Своевременное обнаружение и локализация очага возгорания является одной из важнейших задач в рамках обеспечения комплексной безопасности технологического объекта. Если в небольших закрытых помещениях данная задача решается дымовыми и тепловыми пожарными извещателями, то для обнаружения возгорания на объемных установках, расположенных вне помещений, требуется применение иных технических средств.

Наибольшее распространение получили системы обнаружения огня, основанные на тепловых инфракрасных (ИК) камерах [1, 2] и стандартных камерах видеонаблюдения общего назначения.

Существует несколько основных подходов для обнаружения возгорания по изображению с камер. В работах [3–5] для выявления пламени используется анализ цветового спектра изображения. Другим наиболее популярным подходом служит использование сверточных нейронных сетей (СНС), выполняющих детекцию (обнаружение) объектов на изображении. Архитектуры СНС для данных целей активно развиваются в последние годы, что делает это направление наиболее интересным для исследований.

Цель настоящей работы — создание нейронной сети, способной обнаруживать пламя и дым на изображении с камер видеонаблюдения. В задачи входят:

- выбор оптимальной архитектуры нейронной сети в соответствии с последними исследованиями в этой области;
- ускорение работы выбранной архитектуры с помощью таких методов облегчения нейронной сети, как quantization (квантование) и filter pruning (прореживание фильтров).

Материалы и методы

В последние годы разработано множество архитектур нейронных сетей для обнаружения объектов на изображении. Существует два основных типа, на которые можно разделить такие архитектуры.

Первый тип — двухэтапные (two-stage) детекторы. К этому типу относятся такие архитектуры, как R-CNN [6], Mask R-CNN [7], Pyramid Networks [8]. Принцип работы данных архитектур заключается в определении области, где потенциально находится нужный объект, после чего на втором этапе по этой области производится классификация и уточнение границ объекта. В двухэтапных архитектурах в последнее время экспериментируют с использованием трансформеров, например Swin [9], в качестве backbone на первом этапе вместо сверточных архитектур.

Второй тип — одноэтапные архитектуры. Сюда входят YOLO [10], SSD [11], RetinaNet [12]. В них реализован принцип использования единой сверточной сети и определение положения и класса объекта происходит в один этап.

Как мера качества работы нейронных сетей для обнаружения объектов чаще всего применяется численная метрика *mAP* (mean average precision) на наборе данных COCO (Common Objects in Context) [13]. Эта метрика, несмотря на поиски более репрезентативных альтернатив [14], остается основной в рассматриваемой области. Иногда метрику могут использовать в качестве функции потерь для обучения нейронных сетей [15].

Метрика *mAP* представляет собой среднее по классам значение метрики *AP* (Average Precision — средняя точность). Рассчитывается по формуле:

$$mAP = \frac{1}{k}\sum_{i=1}^{k} AP_k, \quad (1)$$

где k — количество определяемых нейронных сетью классов.

Метрика *AP* — это площадь под Precision-Recall кривой. Данная кривая (рис. 1) строится по точкам в координатах точности (Precision) и полноты (Recall), где каждая точка соответствует значению этих метрик для каждого граничного значения степени уверенности (Confidence) от 0 до 1.

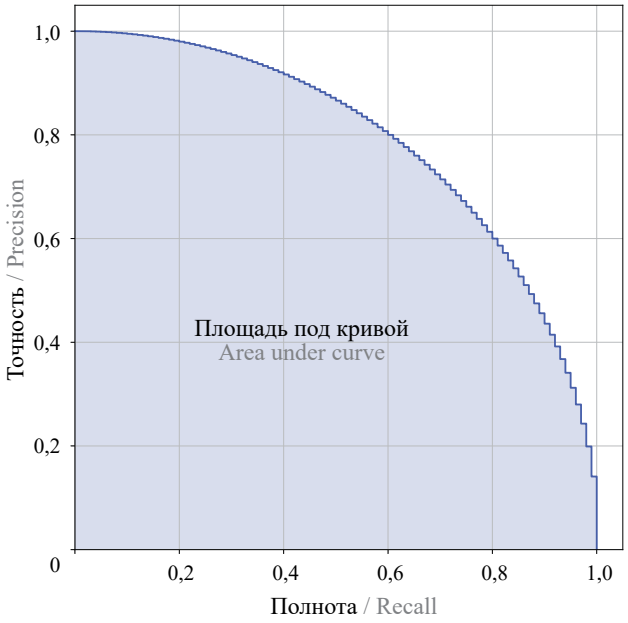

Рис. 1. Площадь под кривой точности – полноты
Fig. 1. Area under Precision – Recall curve

При расчете precision и recall за обнаруженный считается тот объект, у которого показатель *IoU* (Intersection over Union) и Confidence выше граничных.

IoU (Intersection over Union) рассчитывается как отношение площади пересечения размеченной вручную области с объектом на изображении с областью, найденной нейронной сетью, и общей площади, занимаемой этими двумя областями:

$$IoU = \frac{area_{true} \cap area_{pred}}{area_{true} \cup area_{pred}}. \quad (2)$$

У *mAP* существует несколько разновидностей, которые отличаются выбранным граничным значением *IoU*. Наиболее часто можно встретить два вида, где за граничное значение взяты 0,5 и 0,95. Такие метрики обозначаются как $mAP_{0,5}$ и $mAP_{0,95}$ соответственно. Для оценки качества работы нейронной сети на наборе данных COCO рассчитываются значения *mAP* для *IoU* от 0,5 до 0,95 с шагом 0,05, после чего по этим значения берется среднее.

Еще одна важная характеристика нейронной сети — скорость обработки изображения. Есть несколько абсолютных метрик, характеризующих скорость работы. Первая — количество вычислительных операций с плавающей точкой (Floating Point Operations, FLOPs) для обработки одного изображения. Другой косвенной метрикой производительности служит количество параметров (весов) нейронной сети. Однако оба эти параметра не являются прямой характеристикой скорости обработки, так как конечная скорость зависит от множества нюансов, в первую очередь таких, как формат модели нейронной сети, архитектура вычислителя и архитектура нейронной сети. Поэтому для оценки непосредственно скорости работы используют метрику количества изображений (кадров) в секунду (Frames Per Second, FPS), которое модель может обработать на определенном вычислителе, или обратную ей величину — время обработки одного изображения.

Недостаток этой метрики заключается в ее относительности и зависимости от конкретного вычислителя. С другой стороны, она позволяет точно определить, насколько именно отличаются разные модели по скорости работы.

В настоящей работе выбор архитектуры основывается на показателях качества обработки, скорости вычисления и удобства модификации архитектуры для более тонкой настройки под конкретную задачу.

Сегодня наивысшими показателями *mAP* на наборе данных COCO обладают архитектуры YOLOR-D6 [16] с показателем 0,573 и YOLOv6-L6 [17] с показателем 0,572. Однако у них высокая точность достигается за счет больших моделей, что приводит к низкой скорости обработки порядка 46 FPS на вычислителе Nvidia Tesla V100. В то же время YOLOv4 [18] с *mAP* 0,554 имеет показатель 161 FPS на том же вычислителе. При этом в репозитории, реализующем немного усовершенствованную архитектуру во фреймворке PyTorch на языке Python, где модель названа YOLOv5, есть наиболее гибкие возможности для изменения архитектуры под конкретную задачу. Поэтому в данной работе используется именно эта архитектура.

По своей структуре архитектура YOLOv5 устроена следующим образом (рис. 2).

YOLOv5, как и большинство архитектур семейства YOLO, состоит из трех основных блоков: позвоночник (Backbone), шея (Neck) и голова (Head).

Backbone включает CHC Darknet53, впервые использованную в YOLOv3 [19] с применением метода Cross Stage Partial (CSP). Данный блок извлекает карты признаков (feature maps) из исходного изображения.

Neck состоит из блока пространственной объединяющей пирамиды (Spatial Pyramid Pooling — SPP) и сети агрегации путей Path Aggregation Network (PANet). Эта часть выполняет агрегацию признаков из Backbone для выявления наиболее важных из них.

Head — блок, реализующий определение объектов (их позицию, размер, класс и Confidence). Принцип работы блока заключается в разделении изображения на квадратную сетку, для каждой ячейки которой на базе признаков из предыдущих слоев устанавливается наличие объектов каждого класса. Особенностью семейства архитектуры YOLO служит использование якорей (anchor). Якорь — это

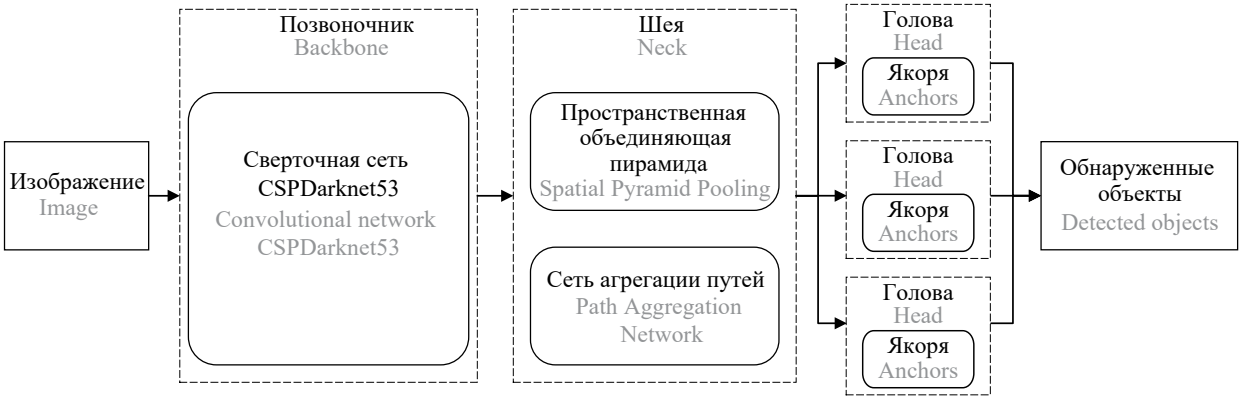

Рис. 2. Схема архитектуры YOLOv5
Fig. 2. YOLOv5 architecture diagram

заранее заданный размер объектов на изображении. В голове их применяется несколько для разных форм объектов. В каждой ячейке определяется, насколько близок найденный объект по форме к каждому якорю, после чего габариты этого объекта корректируются от самого близкого якоря. Кроме того, могут использоваться параллельно несколько голов, с разным набором якорей, например, исходно в YOLOv5 3 головы для мелких объектов, средних и крупных.

Head всегда определяет для каждой ячейки большое количество объектов. Чтобы отфильтровать только нужные объекты, кроме Confidence, применяется метод NMS (non-max suppression). Данный метод применяется с целью убрать сильно пересекающиеся объекты одного класса с высоким значением Confidence и оставить объект с самым высоким значением.

В качестве функции активации в YOLOv5 используется Swish [20] (уравнение (3)). Эта функция — составная, включает сигмоиды и линейную функцию. Swish является сглаженной, немонотонной, ограниченной снизу и неограниченной сверху функцией:

$$f(x) = x \cdot \text{sigmoid}(x) = \frac{x}{1 + e^{-x}}. \quad (3)$$

Наиболее распространенными алгоритмами оптимизации для обучения нейронных сетей служат стохастический градиентный спуск (Stochastic gradient descent — SGD) [21] и метод адаптивного определения момента (Adaptive moment estimation — Adam) [22, 23].

Принцип SGD заключается в обновлении весов модели на каждом шагу эпохи (epoch). Эпохой называется один полный проход по обучающей выборке. Обновление весов происходит в соответствии с градиентами каждого веса, рассчитанными в соответствии с функцией потерь (loss function) на одной партии (batch) данных. Данные разбиваются на партии, чтобы в обучение за один раз подавалось столько информации, сколько помещается в памяти вычислителя. Для YOLOv5 функция потерь представляет собой сумму трех составляющих:

$$Loss = L_{cls} + L_{obj} + L_{loc}, \quad (4)$$

где L_{cls} — составляющая, отвечающая за штраф по классу, представляет собой функцию типа Binary Cross Entropy Loss (BCELoss);

L_{obj} — составляющая, отвечающая за штраф по наличию или отсутствию нужного объекта, функция BCELoss;

L_{loc} — составляющая, отвечающая за штраф по позиции найденного объекта, считается как IoU.

Adam является модификацией SGD, основное отличие которого состоит в учитывании градиентов с предыдущих шагов обучения при обновлении весов. Это изменение позволило повысить сходимость обучения, снизив зашумленность в градиентах.

Обучение с использованием Adam осуществляется по следующему алгоритму.

На начальном этапе возможно два варианта. Если используется предобученная (pretrained) готовая модель и выполняется ее дообучение (fine-tuning) на новых данных, то обучение начинается с этими начальными весами. Если обучение происходит с нуля, например, в случае, когда архитектура исходной модели изменена, то веса инициализируются случайным образом.

После этого отмечается итеративный проход по обучающей выборке. На каждом шагу t происходит вычисление по следующим формулам.

В формуле (5) представлен расчет вектора градиентов в нейронной сети:

$$g_t = \nabla_\theta f_t(\theta_{t-1}), \quad (5)$$

где g_t — вектор градиент нейронной сети;

θ — вектор обучаемых параметров нейронной сети;

$f_t(\theta_{t-1})$ — функция, характеризующая нейросеть.

Далее по формулам (6)–(9) [22] рассчитываются значения моментов для шага t. Изначально векторы m и v нулевые:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t; \quad (6)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2; \quad (7)$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}; \quad (8)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \quad (9)$$

где β_1 и β_2 — коэффициенты момента;

$\hat{m}_t$ и $\hat{v}_t$ — скорректированные на смещение векторы моментов.

В итоге по формуле (10) [22] рассчитываются новые значения весов:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \quad (10)$$

где α — коэффициент скорости обучения (learning rate);

ε — крайне малый добавочный коэффициент, предотвращающий деление на ноль.

Обычно для алгоритма Adam устанавливаются следующие значения: $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$ и $\varepsilon = 10^{-8}$.

Существуют различные методы дополнительного ускорения модели для того, чтобы она могла работать на ограниченных вычислительных ресурсах. В данной работе рассмотрены два метода — квантование (quantization) и прореживание фильтров (filter pruning).

Квантование является процессом перевода значений весов нейронной сети на меньшую битность. Например, 32-битные числа с плавающей точкой можно перевести в 8-битные целые числа, что значительно снижает требования к памяти и вычислительной мощности, но может также привести к потере некоторой точности. Квантование может проводиться как во время обучения, так и после него. Метод не всегда дает положительный результат, так как на некоторых вычислителях операции с плавающей точкой оптимизированы лучше, чем целочисленные.

Прореживание фильтров заключается в удалении из сверточных слоев модели фильтров, наименьшим образом влияющих на результат работы нейронной сети. Так же, как и квантование, оно может проводиться как в процессе обучения, так и после него. Процент удаляемых фильтров может определяться как автоматически во время обучения, так и вручную после обучения.

В случае обоих методов их использование во время обучения дает меньшую потерю точности. Таким образом, если обучение модели с нуля целесообразно, то стоит применять их во время обучения, и только в ином случае после.

Результаты и их обсуждение

В качестве базовой обучена стандартная самая маленькая YOLOv5-small с 800 тыс. обучаемых параметров.

Обучение производилось со значением batch, равным 64, в течение 100 эпох. Данные обучения собраны из открытых источников и вручную размечены. Этот набор разделен на обучающую, валидационную и тестовую выборки в соотношении 80 % / 20 % / 20 %.

В процессе исследования стандартная конфигурация backbone была облегчена путем перебора различных комбинаций параметров нейронной сети. Удалось получить конфигурацию, которая работает быстрее, чем исходная сеть, в 2 раза и дает неболь-

Сравнение метрик разных версий обученных нейронных сетей
Comparison of metrics of different versions trained neural networks

Нейронная сеть / Neural network	Метрики / Metrics				
	Валидационный $mAP_{0,5}$ / Validation $mAP_{0.5}$	Валидационный $mAP_{0,5\text{-}0,95}$ / Validation $mAP_{0.5\text{-}0.95}$	Тестовый $mAP_{0,5}$ / Test $mAP_{0.5}$	Тестовый $mAP_{0,5\text{-}0,95}$ / Test $mAP_{0.5\text{-}0.95}$	Время обработки изображения, мс / Image processing time, ms
Исходная / Baseline	0,782	0,553	0,781	0,549	10,2
Облегченная / Lightweight	0,776	0,548	0,784	0,559	5,2
Облегченная после прореживания фильтров / Lightweight after filter pruning	0,773	0,538	0,780	0,557	4,4
Облегченная после квантования / Lightweight after quantization	0,775	0,540	0,781	0,556	4,6
Облегченная после прореживания фильтров и квантования / Lightweight after filter pruning and quantization	0,778	0,536	0,776	0,554	3,9

Рис. 3. Результат обработки нейронной сетью сгенерированного изображения
Fig. 3. Result of neural network processing of generated image

Рис. 4. Результат обработки нейронной сетью фотографии
Fig. 4. Result of neural network processing of photo

шой прирост в качестве работы, который объяснятся меньшим переобучением.

Результаты обучения для стандартной, облегченной и после применения квантования и прореживания представлены в таблице. Время работы на одном изображении приведено для процессора Intel Core i5-6300U.

К облегченной нейронной сети применено прореживание фильтров. В ходе экспериментов наилучший результат отмечен при показателе прореживания 20 %. Прореживание фильтров реализовывалось с помощью библиотеки NNCF (Neural Network Compression Framework). Время обработки упало до 4,4 мс.

Квантование дало небольшой прирост скорости. Скорость работы на CPU составила 4,6 мс. Качество работы снизилось незначительно.

Квантование также осуществляется с помощью NNCF. Библиотека предоставляет возможность выполнять прореживание фильтров и квантование одновременно, в результате чего получается добиться большего ускорения. В этом случае скорость составила 3,9 мс.

Результаты работы нейронной сети на данных из тестовой выборки представлены на сгенерированном изображении пожара на технологическом объекте (рис. 3) и реальной фотографии пожара с заметным дымом (рис. 4).

Выводы

Определена оптимальная архитектура нейронной сети для обнаружения очага возгорания. Этой архитектурой стала YOLOv5, которая является оптимальным выбором по соотношению скорости, качества работы и удобства обучения и применения.

С целью ускорения работы детектора применены методы облегчения нейронной сети, квантования и прореживание фильтров. Их использование в комбинации с подбором оптимальной конфигурации YOLOv5 позволило ускорить нейронную сеть в 2,6 раза. У самой быстрой полученной сети был достигнут показатель $mAP_{0,5}$ на тестовой выборке 0,962. Время обработки одного изображения составило 3,9 мс на процессоре, что позволяет выполнять обработку видео с частотой 256 кадров с секунду.

СПИСОК ИСТОЧНИКОВ

1. *Mazur-Milecka M., Glowacka N., Kaczmarek M., Bujnowski A., Kaszynski M., RuminskiSmart J.* City and fire detection using thermal imaging // 14th International Conference on Human System Interaction (HSI). 2021. Pp. 1–7. DOI: 10.1109/HSI52170.2021.9538699
2. *Chacon M., Perez-Vargas F.* Thermal Video Analysis for Fire Detection Using Shape Regularity and Intensity Saturation Features // Third Mexican Conference MCPR. 2011. Pp. 118–126. DOI: 10.1007/978-3-642-21587-2_13
3. *Zaman T., Hasan M., Ahmed S., Ashfaq S.* Fire Detection Using Computer Vision // IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS). 2018. Pp. 356–359. DOI: 10.1109/MWSCAS.2018.8623842
4. *Manjunatha K., Mohana H., Vijaya P.* Implementation of Computer Vision Based Industrial Fire Safety Automation by Using Neuro-Fuzzy Algorithms // I.J. Information Technology and Computer Science. 2015. Pp. 14–27. DOI: 10.5815/ijitcs.2015.04.02

5. *Mondal M., Prasad V., Kumar R., Saha N., Saumadeep G., Ratna G., Mukhopadhyay A., Sourav S.* Automating Fire Detection and Suppression with Computer Vision: A Multi-Layered Filtering Approach to Enhanced Fire Safety and Rapid Response // Fire Technol. 2023. DOI: 10.1007/s10694-023-01392-w
6. *Girshick R., Donahue J., Darrell T., Malik J.* Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation // 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014. Pp. 580–587. DOI: 10.1109/CVPR.2014.81
7. *He K., Gkioxari G., Dollar P., Girshick R.* Mask R-CNN // 2017 IEEE International Conference on Computer Vision (ICCV). 2017. Pp. 2980–2988. DOI: 10.1109/ICCV.2017.322
8. *Lin T., Dollar P., Girshick R., He K., Hariharan B., Belongie S.* Feature Pyramid Networks for Object Detection // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. Pp. 936–944. DOI: 10.1109/CVPR.2017.106
9. *Liu Z. et al*. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows // 2021 IEEE/CVF International Conference on Computer Vision (ICCV). 2021. Pp. 9992–10002. DOI: 10.1109/ICCV48922.2021.00986
10. *Redmon J., Divvala S., Girshick R., Farhadi A.* You Only Look Once: Unified, Real-Time Object Detection // CVPR. 2016. Pp. 779–788. DOI: 10.1109/CVPR.2016.91
11. *Liu W. et al*. SSD: Single Shot MultiBox Detector // Computer Vision — ECCV Lecture Notes in Computer Science. 2016. Vol. 9905. DOI: 10.1007/978-3-319-46448-0_2
12. *Lin T., Goyal P., Girshick R., He K., Dollar P.* Focal Loss for Dense Object Detection // 2020 IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. Vol. 42. Issue 2. Pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826
13. *Lin T. et al.* Microsoft COCO: Common Objects in Context // ECCV 2014. Lecture Notes in Computer Science. 2014. Vol. 8693. DOI: 10.1007/978-3-319-10602-1_48
14. *Sobti A., Arora C., Balakrishnan M*. Object Detection in Real-Time Systems: Going Beyond Precision // 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). 2018. Pp. 1020–1028. DOI: 10.1109/WACV.2018.00117
15. *Henderson P., Ferrari V.* End-to-End Training of Object Class Detectors for Mean Average Precision // ACCV 2016. Lecture Notes in Computer Science. 2016. Vol. 10115. DOI: 10.1007/978-3-319-54193-8_13
16. *Li C., Li L., Geng Y., Jiang H., Cheng M., Zhang B., Ke Z., Xu X., Chu X.* YOLOv6 v3.0: A Full-Scale Reloading // arXiv: Computer Science. 2023. DOI: 10.48550/arXiv.2301.05586
17. *Wang C., Yeh I., Liao H.* You Only Learn One Representation: Unified Network for Multiple Tasks // Journal of Information Science and Engineering. 2021. Vol. 39. Issue 2. Pp. 691–709. DOI: 10.48550/arXiv.2105.04206
18. *Wang C., Bochkovskiy A., Liao H.* Scaled-YOLOv4: Scaling Cross Stage Partial Network // 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021. Pp. 13024–13033. DOI: 10.1109/CVPR46437.2021.01283
19. *Redmon J., Farhadi A.* YOLOv3: An Incremental Improvement // arXiv: Computer Science. 2018. DOI: 10.48550/arXiv.1804.02767
20. *Ramachandran P., Zoph B., Le Q.* Swish: a Self-Gated Activation Function // arXiv: Neural and Evolutionary Computing. 2017. DOI: 10.48550/arXiv.1710.05941
21. *Robbins H.* A Stochastic Approximation Method // Annals of Mathematical Statistics. 1951. Vol. 22. Pp. 400–407. DOI: 10.1214/aoms/1177729586
22. *Kingma D., Ba J.* Adam: A Method for Stochastic Optimization // arXiv: Computer Science. 2014. DOI: 10.48550/arXiv.1412.6980
23. *Zhang Z.* Improved Adam Optimizer for Deep Neural Networks // 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). 2018. Pp. 1–2. DOI: 10.1109/IWQoS.2018.8624183

REFERENCES

1. Mazur-Milecka M., Glowacka N., Kaczmarek M., Bujnowski A., Kaszynski M., RuminskiSmart J. City and fire detection using thermal imaging. *14th International Conference on Human System Interaction (HSI)*. 2021; 1-7. DOI: 10.1109/HSI52170.2021.9538699
2. Chacon M., Perez-Vargas F. Thermal Video Analysis for Fire Detection Using Shape Regularity and Intensity Saturation Features. *Third Mexican Conference MCPR*. 2011; 118-126. DOI: 10.1007/978-3-642-21587-2_13
3. Zaman T., Hasan M., Ahmed S., Ashfaq S. Fire Detection Using Computer Vision. *IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2018; 356-359. DOI: 10.1109/MWSCAS.2018.8623842

4. Manjunatha K., Mohana H., Vijaya P. Implementation of Computer Vision Based Industrial Fire Safety Automation by Using Neuro-Fuzzy Algorithms. *I.J. Information Technology and Computer Science.* 2015; 14-27. DOI: 10.5815/ijitcs.2015.04.02
5. Mondal M., Prasad V., Kumar R., Saha N., Saumadeep G., Ratna G., Mukhopadhyay A., Sourav S. Automating Fire Detection and Suppression with Computer Vision: A Multi-Layered Filtering Approach to Enhanced Fire Safety and Rapid Response. *Fire Technol*. 2023. DOI: 10.1007/s10694-023-01392-w
6. Girshick R., Donahue J., Darrell T., Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition.* 2014; 580-587. DOI: 10.1109/CVPR.2014.81
7. He K., Gkioxari G., Dollar P., Girshick R. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017; 2980-2988. DOI: 10.1109/ICCV.2017.322
8. Lin T., Dollar P., Girshick R., He K., Hariharan B., Belongie S. Feature Pyramid Networks for Object Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017; 936-944. DOI: 10.1109/CVPR.2017.106
9. Liu Z. et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021; 9992-10002. DOI: 10.1109/ICCV48922.2021.00986
10. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *CVPR*. 2016; 779-788. DOI: 10.1109/CVPR.2016.91
11. Liu W. et al. SSD: Single Shot MultiBox Detector. *Computer Vision — ECCV Lecture Notes in Computer Science*. 2016; 9905. DOI: 10.1007/978-3-319-46448-0_2
12. Lin T., Goyal P., Girshick R., He K., Dollar P. Focal Loss for Dense Object Detection. *2020 IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020; 42(2):318-327. DOI: 10.1109/TPAMI.2018.2858826
13. Lin T. et al. Microsoft COCO: Common Objects in Context. *ECCV 2014. Lecture Notes in Computer Science.* 2014; 8693. DOI: 10.1007/978-3-319-10602-1_48
14. Sobti A., Arora C., Balakrishnan M. Object Detection in Real-Time Systems: Going Beyond Precision. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018; 1020-1028. DOI: 10.1109/WACV.2018.00117
15. Henderson P., Ferrari V. End-to-End Training of Object Class Detectors for Mean Average Precision. *ACCV 2016. Lecture Notes in Computer Science.* 2016; 10115. DOI: 10.1007/978-3-319-54193-8_13
16. Li C., Li L., Geng Y., Jiang H., Cheng M., Zhang B., Ke Z., Xu X., Chu X. YOLOv6 v3.0: A Full-Scale Reloading. *arXiv: Computer Science*. 2023. DOI: 10.48550/arXiv.2301.05586
17. Wang C., Yeh I., Liao H. You Only Learn One Representation: Unified Network for Multiple Tasks. *Journal of Information Science and Engineering*. 2021; 39(2):691-709. DOI: 10.48550/arXiv.2105.04206
18. Wang C., Bochkovskiy A., Liao H. Scaled-YOLOv4: Scaling Cross Stage Partial Network. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021; 13024-13033. DOI: 10.1109/CVPR46437.2021.01283
19. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. *arXiv: Computer Science*. 2018. DOI: 10.48550/arXiv.1804.02767
20. Ramachandran P., Zoph B., Le Q. Swish: a Self-Gated Activation Function. *arXiv: Neural and Evolutionary Computing*. 2017. DOI: 10.48550/arXiv.1710.05941
21. Robbins H. A Stochastic Approximation Method. *Annals of Mathematical Statistics*. 1951; 22:400-407. DOI: 10.1214/aoms/1177729586
22. Kingma D., Ba J. Adam: A Method for Stochastic Optimization. *arXiv: Computer Science*. 2014. DOI: 10.48550/arXiv.1412.6980
23. Zhang Z. Improved Adam Optimizer for Deep Neural Networks. *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS).* 2018; 1-2. DOI: 10.1109/IWQoS.2018.8624183

Поступила 24.07.2023, после доработки 05.09.2023;
принята к публикации 12.09.2023
Received July 24, 2023; Received in revised form September 5, 2023;
Accepted September 12, 2023

Информация об авторах

ЕВСИКОВ Андрей Александрович, аспирант, Российский государственный университет нефти и газа (национальный исследовательский университет) имени И.М. Губкина, Россия, 119991, г. Москва, Ленинский пр-т, 65, корп. 1; ORCID: 0009-0007-4974-7948; e-mail: andreyev4@gmail.com

САМАРИН Илья Вадимович, д-р техн. наук, доцент, заведующий кафедрой автоматизации технологических процессов, Российский государственный университет нефти и газа (национальный исследовательский университет) имени И.М. Губкина, Россия, 119991, г. Москва, Ленинский пр-т, 65, корп. 1; РИНЦ ID: 867674; ORCID: 0000-0003-2430-5311, e-mail: ivs@gubkin.ru

Information about the authors

Andrey A. EVSIKOV, Postgraduate Student, Gubkin Russian State University of Oil and Gas (National Research University), Leninskiy Avenue, 65, Bldg. 1, Moscow, 119991, Russian Federation; ORCID: 0009-0007-4974-7948; e-mail: andreyev4@gmail.com

Ilya V. SAMARIN, Dr. Sci. (Eng.), Docent, Head of Department of Automation of Technological Processes, Gubkin Russian State University of Oil and Gas (National Research University), Leninskiy Avenue, 65, Bldg. 1, Moscow, 119991, Russian Federation; ID RISC: 867674; ORCID: 0000-0003-2430-5311, e-mail: ivs@gubkin.ru

Вклад авторов:

Евсиков А.А. — *концепция исследования; проведение экспериментов; написание исходного текста.*

Самарин И.В. — *научное руководство; доработка текста; итоговые выводы.*

Авторы заявляют об отсутствии конфликта интересов.

Contribution of the authors:

Andrey A. Evsikov — *the concept of the study; conducting experiments; writing the source text.*

Ilya V. Samarin — *scientific guidance; revision of the text; final conclusions.*

The authors declare no conflicts of interests.