УДК 004.89:004.912

Силичева О.В.

студент

Научный руководитель: Саркисова И.О., к.т.н

ФГБОУ ВО МГТУ «СТАНКИН»

Россия, г. Москва

ДЕТЕКЦИЯ ПАТТЕРНОВ РИСКА ВОЗГОРАНИЙ НА АГЗС ОБУЧЕННОЙ НЕЙРОСЕТЬЮ.

Аннотация. *В статье анализируется архитектура сверточных нейросетей, позволяющая производить детекцию паттернов риска возгораний без использования значительных вычислительных мощностей и дорогостоящего оборудования. Также рассматривается процесс обучения нейросети на собственных наборах данных.*

Ключевые слова: *нейросеть, детекция объектов, YOLO.*

Silicheva O.V.

student

Scientific supervisor: Sarkisova I.O., c.t.s.

MSTU «STANKIN»

Russia, Moscow

DETECTION OF FIRE RISK PATTERNS ON THE GAS STATION BY A TRAINED NEURAL NETWORK.

Annotation. *The article analyzes the architecture of convolutional neural networks, which allows the detection of fire risk patterns without the use of significant computing power and expensive equipment. The process of training a neural network on its own data sets is also considered.*

Keywords: *neural network, object detection, YOLO.*

Наиболее быстрым образом реагировать на риск возгораний позволяет использование детекции паттернов риска на видеопотоке с камер наружного слежения с помощью обученной нейросети. Аналитика

видеоряда с обычных камер должна включать в себя детекцию огня и дыма, проливаний топлива, а также самых частых паттернов небезопасного поведения (курение, использование зажигалки для освещения бензобака, трогание с места с заправочным пистолетом в баке транспортного средства).

Существует несколько наиболее популярных сверточных нейросетей, работающих в режиме реального времени, однако большинство из них имеют высокие технические требования, сложные в настройке и конфигурировании, а также требующие значительных финансовых затрат. Например, Google TensorFlow EfficientDet или FaceBook Pytorch/Detectron требуют большое количество мощных видеокарт с ядрами параллельных вычислений для обучения.

Однако эту проблему решает использование предобученных нейросетей класса YOLO (You Only Look Once). Работа нейросетей по распознаванию объектов обычно предполагала анализ изображения в два этапа: на первом осуществлялся поиск сегмента изображения/кадра, содержащий искомый объект, а уже на втором происходила сама классификация. Архитектура YOLO в свою очередь анализирует изображение единожды. Алгоритм YOLO основан на нейронных сетях, которые обучены на большом количестве изображений с размеченными объектами. На этапе обучения сеть анализирует характеристики объектов разных классов и далее использует эту информацию для обнаружения объектов на новых изображениях.

Процесс работы YOLO достаточно прост: алгоритм разбивает изображение на сетку (обычно 19x19 или 13x13 ячеек) и для каждой ячейки определяет вероятность наличия в ней объекта и его координаты. Также для каждого объекта определяется его класс (например, автомобиль, человек, собака и т.д.). (рис.1).

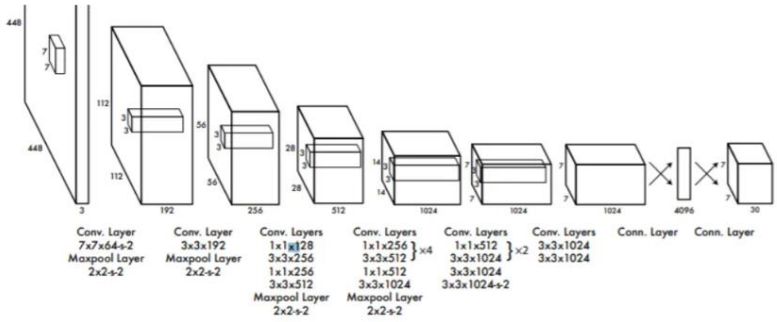

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Рис. 1. Архитектура YOLO

После этого происходит фильтрация результатов – удаляются объекты с низкой вероятностью и объединяются близкие объекты. На выходе получается список обнаруженных объектов с их координатами и классами.

Преимуществом YOLO является высокая скорость работы алгоритма. Он может обрабатывать изображения со скоростью до 45 кадров в секунду на современном GPU. Это делает его очень эффективным для использования в реальном времени – например, для распознавания объектов на видеопотоке.

Однако, как и любой другой алгоритм, YOLO не без недостатков. Например, он может иметь проблемы с обнаружением мелких объектов или с объектами, которые перекрываются другими объектами. Также YOLO может допускать ошибки при классификации объектов – например, если объект очень похож на объект другого класса.

Наиболее стабильной и проверенной версией нейросети YOLO на данный момент является четвертая. Однако, поскольку нейросеть YOLOv4 написана на C++, недостаточно просто установить библиотеку через консоль. Необходимо последовательно выполнить следующие задачи:

1. Установить Git.
2. Установить CMake.
3. Установить Visual Studio.
4. Обновить драйвер видеокарты.

5. Установить CUDA.
6. Установить CuDNN.
7. Установить OpenCV.
8. Сконфигурировать OpenCV через CMake.
9. Сборка OpenCV в Visual Studio.
10. Установка Darknet.

Таким образом создание необходимой конфигурации является довольно трудоемким, поскольку требуется особое соотношение версий используемых ПО, которое устанавливается только опытным путем. Исходя из этого целесообразно использовать пятую версию, которая хоть и не является официальной в полном смысле, вполне стабильна и проверена временем. Основное её преимущество в том, что она уже сконфигурирована и устанавливается в качестве python-пакета.

Для обучения необходимо собрать набор фотографий детектируемых паттернов, сегментировать эти изображения, выделив зоны паттерна, и начать процесс обучения, указав количество картинок на вход единовременно, количество эпох обучения, файл весов и количество потоков. С целью повышения качества распознавания на видео следует создавать наборы данных на базе видеозаписей детектируемых процессов. Для этого каждое такое видео длительностью до минуты минуты следует разделить на 400-600 фреймов.

Следующим этапом необходимо создать особую структуру папок: в папке «train_data» должно содержаться две подпапки – «images» и «labels», в каждой из которых содержится по папке «train» и «val». Названия могут быть иными, но рассмотренные считаются общепринятыми. Суть такой иерархии в том, что в папке с обучающими данными есть две категории данных, сами изображения и лейблы. Лейблы – это текстовые файлы, содержащие координаты рамки, выделяющей детектируемый объект и индекс названия этого объекта. Внутреннее разделение папок с

изображениями и лейблами на две подпапки разграничивает данные для обучения и для проверки. То есть нейросеть сначала обучается, а затем по валидационным данным проверяет на сколько хорошо, она обучилась.

Для того, чтобы подготовить изображения к обучению, следует соотнести каждый фрейм с лейблом и метаданными, содержащими в себе список лейблов в целом. Для этого можно использовать приложение «Make

Далее необходимо обучить нейросеть, стоит упомянуть, что в зависимости от желания и потребностей можно использовать модели разных размеров, зависимость скорости и качества распознавания от выбранного размера представлена на рисунке 2.

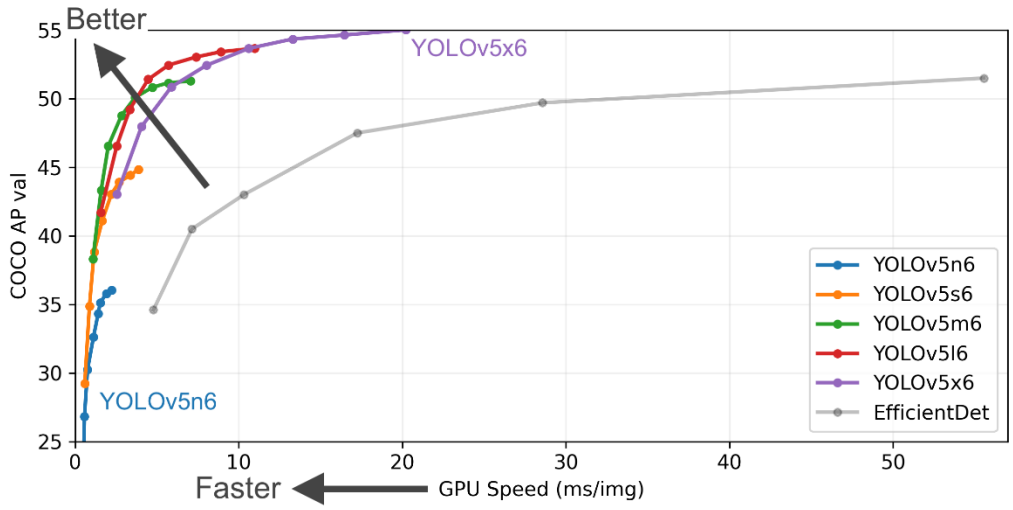

Рис. 2. Соотношение скорости и качества распознавания для всех размеров моделей

Для обучения необходимо указать такие параметры, как размер изображения, количество изображений обрабатываемых одновременно, количество эпох, путь к папке с обучающими данными и путь до весов, с которых следует начать или продолжить тренировку. После окончания обучения следует выбрать модель с наилучшими весами и на ее основании уже проводить детекцию видео.

Использованные источники:

1. Сайт «Введение в YOLO: обнаружение объектов в реальном времени» [Электронный ресурс] – Режим доступа: https://hashdork.com/ru/Yolo/, свободный. Дата обращения: 23.05.2022 г.

2. Сайт «yolov5» [Электронный ресурс] – Режим доступа: https://github.com/ultralytics/yolov5/, свободный. Дата обращения: 23.05.2022 г.

3. Сайт «YOLOv4 – самая точная real-time нейронная сеть на датасете Microsoft COCO» [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/503200//, свободный. Дата обращения: 23.05.2022 г.