О задаче разработки автоматического конвейера запуска моделей машинного обучения

М.С. Мосева

Московский технический университет связи и информатики, Москва

Аннотация: В работе рассматривается разработка системы выполнения моделей машинного обучения. Разрабатываемая система представляет собой набор микро-сервисов, способных найти применение в различных областях производства. Рассматриваются технологии для реализации и перспективы разрабатываемого продукта. В работе используются современные технологии сегментации и распознавания объектов на кадрах, а также технологии, позволяющие выстроить инфраструктуру для данной системы, и технологии разработки программного обеспечения.
Ключевые слова: машинное обучение, компьютерное зрение, микросервисная архитектура, распознавание образов.

Введение

Технологии не стоят на месте — за последнее время был сделан большой рывок в сфере машинного обучения. Совсем недавно алгоритмы машинного обучения могли только определять, что на фотографии еда или питомец, и даже такие возможности уже восхищали людей. На данный момент развитие машинных алгоритмов ушло гораздо дальше [1].

Новые технологии находят свое применение в таких сферах жизни и таких отраслях, как здравоохранение, автомобильная промышленность, банковская нефть и газ, а также финансы и транспорт. Есть несколько областей, в том числе одна из самых быстро развивающихся областей на сегодняшний день - компьютерное зрение (от англ. Computer Vision). Современное машиностроение и робототехника требуют всё более детализированных результатов выполнения анализа окружающей среды, получаемых с камер и других приборов.

Компьютерное зрение — это область информатики, которая пытается воспроизвести зрительную систему человека и предоставить компьютерам возможность идентифицировать и обрабатывать объекты в изображениях и

видео так же, как это делают люди. До недавнего времени компьютерное зрение работало только в ограниченном объеме [2].

Но благодаря достижениям в области искусственного интеллекта и инновациям в области глубинного обучения и нейронных сетей, в последние годы эта область смогла совершить большие скачки и смогла превзойти людей в некоторых задачах, связанных с обнаружением и определением объектов.

Одним из движущих факторов роста компьютерного зрения является объем данных, которые мы генерируем сегодня, и которые затем используются для обучения и улучшения компьютерного зрения.

Наряду с огромным объемом визуальных данных (более 3 миллиардов изображений ежедневно обмениваются онлайн), теперь доступна вычислительная мощность, необходимая для анализа данных [3]. По мере роста области компьютерного зрения с новым оборудованием и алгоритмами растут и показатели точности идентификации объектов. Менее чем за десятилетие современные системы достигли высокой точности, при быстрой реакции на визуальные входы.

В таких отраслях, как розничная торговля, страхование, производство и им подобных, компьютерное зрение позволяет повысить уровень удовлетворённости клиентов.

Для упрощения использования различных моделей машинного обучения было принято решение разработать информационную систему, позволяющую взаимодействовать с моделью машинного обучения, как с сервисом.

Проектирование архитектуры

HTTP и REST API

Рассмотрим, как веб-приложения взаимодействуют в Интернете при помощи модели "Запрос - Ответ". Когда пользователь составляет запрос в

поисковой системе "Google" и нажимает кнопку «найти», веб-приложение формирует и отправляет запрос для сервера. После этого, сервер возвращает ответ, в котором содержится результат поиска.

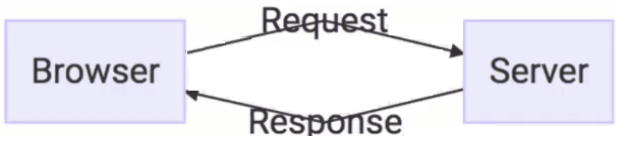

Рисунок 1. – Схема взаимодействия браузера и сервера

Каждый HTTP-запрос содержит:

- Заголовки - при помощи которых описывается сообщение, его метаданные, например, кем было отправлено сообщение, а также информацию, в каком формате представлено сообщение.
- Тело запроса - само содержимое запроса, которое было отправлено. Оно не является обязательным.

У HTTP протокола есть множество методов для отправки запроса, но рассмотрим 4 основных:

- GET - запрашивает представление указанного ресурса. При этом, тело запроса пустое.
- POST - используется для отправки тела сообщения и последующего создания записи на сервере.
- PUT - обновляет переданными в теле запроса данными, данные, уже имеющиеся на сервере.
- DELETE - удаляет указанные данные на сервере.

Так же у HTTP есть статусы ответов. Можно разделить их на 5 группы:

- 1xx – информационные;
- 2xx – положительные;
- 3xx - указывающие на то, что произошло перенаправление;
- 4xx - ошибки клиента.
- 5xx - ошибки сервера.

REST — это архитектурный подход к созданию приложений, которые работают на основе HTTP протокола [4]. Тело запроса в основном представлено в виде JSON-документа.

Docker

Теперь необходимо подготовить среду для работы модели. Так как среда должна быть легко переносимой и платформонезависимой - самое лучшее решение — это воспользоваться механизмом контейнеризации, предоставляемой компанией «Docker» [5].

Docker — это отраслевой стандарт для контейнеров. Контейнеры — это стандартизированная единица программного обеспечения, которая позволяет разработчикам "упаковать" свое приложение со всем его окружением и зависимостями, решая проблему "это работает на моей машине".

Несколько контейнеров могут работать на одной машине и совместно использовать ядро ОС. Каждый из контейнеров работает как изолированный процесс в своём пространстве. Контейнеры занимают меньше места, чем виртуальные машины.

Виртуальные машины — это абстракция физического оборудования, позволяющая разделить один физический сервер на много отдельных. Делается это при помощи гипервизора. Каждая ВМ включает полную копию операционной системы, приложения, необходимые бинарные файлы и библиотеки, занимая десятки гигабайт.

Именно такой функционал и необходим для создания платформонезависимой архитектуры.

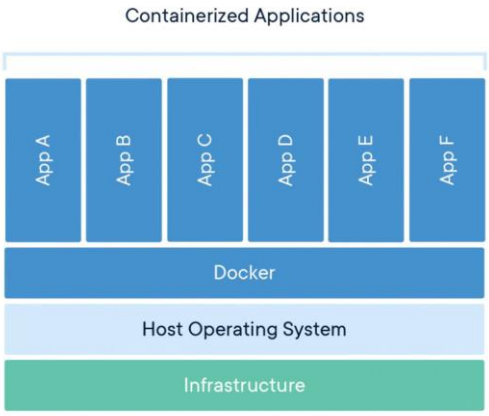

Рисунок 2. – Схематическое изображение сервера с Docker

На данном этапе были рассмотрены два главных компонента: веб-интерфейс и серверное приложение, далее необходимо организовать отправку от последнего компонента к модели.

Apache Kafka

Так как в проекте используется микросервисная архитектура, необходим инструмент, который поможет обеспечить потоковую передачу сообщений (кадров) между микросервисами масштабируемой и распределенной системы. Использование очередей — один из подходящих вариантов.

Очередь — это структура данных с определенным порядком хранения и доступа. Основные плюсы: обмен данными не зависит друг от друга, что позволяет добиться слабой связности; очереди позволяют экономить ресурсы; процессы отделены друг от друга и поэтому повышается отказоустойчивость системы; самый важный плюс — они позволяют добиться масштабируемости, так как позволяют распределить процесс обработки информации. Благодаря всем этим плюсам, появляется возможность контролировать скорость добавления сообщения и их дальнейшей обработки.

Apache Kafka — это одна из самых используемых платформ потоковой передачи сообщений, способная обрабатывать триллионы сообщений в день

[6]. Данная реализация используется такими компаниями, как airbnb, netflix, microsoft и многими другими.

Kafka выполняет три основных функции, позволяющие реализовать сценарии потоковой передачи сообщений:

- Запись и чтение потоков событий, включая непрерывный импорт/экспорт данных из других систем.
- Долговечное и надежное хранение потоков событий в течение необходимого времени.
- Обработка потоков событий по мере их возникновения или ретроспективно.

Все эти функции предоставляются в распределенном, высокомасштабируемом, эластичном, отказоустойчивом и безопасном виде. Kafka может быть развернута на пустом оборудовании, виртуальных машинах и контейнерах, как в локальной сети, так и в облаке.

В Kafka есть два основных понятия — потребитель и поставщик. Поставщик отправляет данные в очередь, а потребитель их вычитывает. Топики — это аналог самой очереди [7].

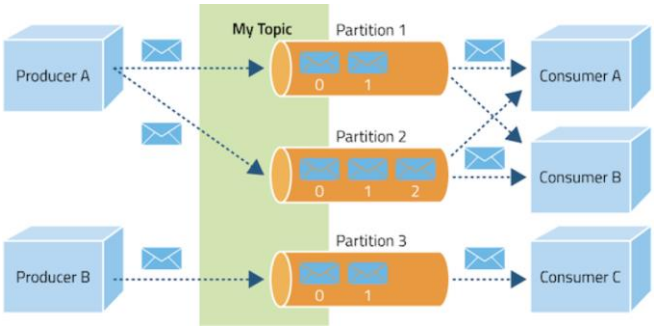

Рисунок 3. – Внутреннее устройство Kafka

Результатом проектирования будет следующая схема взаимодействия: в Web API приходит изображение или видео, которое необходимо обработать [8]. После этого оно конвертируется в формат, понятный модели, и отправляется в менеджер очередей Kafka для дальнейшего получения этого

сообщения в Detectron2, где к нему будет применена модель для определения объектов. В результате будет получено изображение с размеченными данными и отправлено обратно в очередь Kafka. На финальном этапе это изображение будет выведено пользователю в веб-интерфейсе. Дополнительно пользователю предоставляется возможность для обработки видеопотока с камеры с помощью модуля «Camera». Данный процесс также представлен на рисунке 4:

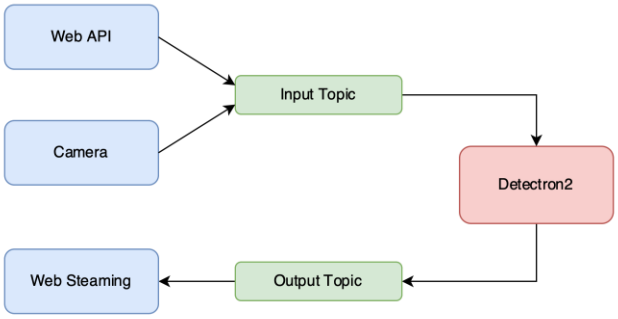

Рисунок 4. – Схема информационной системы

Алгоритм работы системы заключается в следующем [9]:

1) Взаимодействие с пользовательским интерфейсом, выбор модели и входных данных
2) Получение файла из запроса и открытие изображения при помощи библиотеки PIL. Конвертация цветов в RGB, сохранение в формате JPEG и конвертация в байтовый поток для отправки в очередь.
3) Получение сообщения с изображениями из входной очереди Kafka. Обработка изображения при помощи модели машинного обучение и отправка полученного результирующего изображения в выходную очередь Kafka.

Для упрощения развертывания разработанной системы был сформирован Docker-контейнер. Для этого был использован Dockerfile. За основу был взят специально подготовленный Docker-образ от Nvidia с

поддержкой Cuda для параллельных вычислений на GPU [10]. Помимо этого, производится базовая настройка операционной системы: загрузка необходимых пакетов, создание пользователя, настройка python, установка библиотек для работы приложения, в том числе, и библиотек для работы с машинным обучением и запуск созданного приложения.

Заключение

Результатом исследования является промышленная система машинного зрения, имеющая микро-сервисы: для отправки фотографий из веб-браузера, микро-сервис для потоковой передачи видео для обработки, микро-сервис для обработки потоков данных при помощи нейронной сети, микро-сервис для просмотра результатов выполнения.

Для реализации возможности распознавания образов было подробно рассмотрено применение библиотеки Detectron2 для сегментации и определения объектов. Также были рассмотрены инструменты: для распределённой потоковой передачи данных, контейнеризации и инструменты, позволяющие создавать веб-приложения.

Использование библиотеки Detectron2 позволило реализовать микро-сервис, способный принимать запросы из распределённой очереди и проводить анализ кадра, на предмет наличия определённых объектов.

В результате выполнения работы был выбрана микро-сервисная архитектура для проектирования системы машинного зрения. Это позволило создать автономные, независимые от друг друга, компоненты системы, отделив тем самым компонент по анализу изображения от пользовательского интерфейса и распределённой системы потоковой передачи данных. Так как компоненты небольшие и независимы друг от друга, такую систему легко масштабировать, то есть добавить другие компоненты, реализованные с использованием других языков программирования и технологий, не

затрагивая при этом уже существующие компоненты системы. Достаточно лишь подключиться к распределённой очереди для отправки данных.

Литература

1. Makridakis S. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms, Futures, 2017, Volume 90. URL: doi.org/10.1016/j.futures.2017.03.006.
2. Solem, J., Programming computer vision with Python. Beijing: O'Reilly, 2012, pp: 274.
3. Elgendy N., Elragal A. Big Data Analytics: A Literature Review Paper. Perner, P. (eds) Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2014. Lecture Notes in Computer Science, 2014, vol 8557. Springer, Cham. DOI: doi.org/10.1007/978-3-319-08976-8_16
4. Ayub Khan A., Laghari A., Awan S., Lyari, Karachi P. Machine Learning in Computer Vision: A Review. ICST Transactions on Scalable Information Systems, 2021, DOI: 10.4108/eai.21-4-2021.169418.
5. Neumann A., Laranjeiro N., Bernardino J. "An Analysis of Public REST Web Service APIs," in IEEE Transactions on Services Computing, vol. 14, no. 4, pp. 957-970, 2021, DOI: 10.1109/TSC.2018.2847344.
6. Treveil M., Omont N., Stenac C., Lefevre K., Phan D., Zentici J., Lavoillotte A., Miyazaki M., Heidmann L. Introducing MLOps. O'Reilly Media, Sebastopol, 2020, pp. 150.
7. Shapira G., Palino T., Sivaram R., Petty K. Kafka: The Definitive Guide, 2nd Edition. In: O'Reilly Media, Sebastopol, 2021, pp. 425.
8. Martín C., Langendoerfer P., Díaz M., Soltani Zarrin P., Rubio B. Kafka-ML: Connecting the data stream with ML/AI frameworks, Future Generation Computer Systems, Volume 126, 2022. URL: doi.org/10.1016/j.future.2021.07.037.
9. Vartak M., Subramanyam H., Lee W.-E., Viswanathan S., Husnoo S.,

Madden S., Zaharia M.. ModelDB: a system for machine learning model management. In Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA '16). Association for Computing Machinery, New York, NY, USA, Article 14, 2016. URL: doi.org/10.1145/2939502.2939516.

10. Bashari Rad B., Bhatti H., Ahmadi M. An Introduction to Docker and Analysis of its Performance. IJCSNS International Journal of Computer Science and Network Security, 2017, VOL.17, No.3, pp. 228-235.