Разработка облачной платформы для сбора, анализа и хранения видеоданных

С.О. Степаненко[1], П.Ю. Якимов[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. На сегодняшний день анализ видеоданных является крайне актуальным. Объем видео постоянно растет и есть необходимость обрабатывать его. Очень часто есть набор источников видео, на которых непрерывно идет запись видео. В данной статье представлена реализация платформы, которая агрегирует видео с различных источников и обрабатывает его с помощью различных сервисов, основанных на искусственных нейронных сетях и методах машинного обучения.

1. Введение

Задача обработки видео является все более востребованной. Видеоускорители становятся все мощнее и обработка большого количества данных таких как видео становится быстрее. Но количество источников видеоданных, как правило, большое, и задачи обработки видео разные. Примером таких задач может служить поиск объектов на видеопотоке, подсчет людей и так далее. Такие задачи часто решаются с помощью искусственных нейронных сетей и методов машинного обучения. Такой подход позволяет решать задачу классификации с высокой точностью. Минус такого подхода – нужно много вычислительных ресурсов. При использовании нейронных сетей для обработки видео имеется один существенный недостаток: необходимость большого количества вычислительных ресурсов. На видеокарте NVIDIA GeForce 2080 TI скорость обработки видео реализацией алгоритма обнаружения объектов YOLO составила 170 кадров в секунду [1]. Из-за этого один сервис в большинстве случаев будет использовать целое вычислительное устройство и запустить другой сервис обработки не получится. Решение такой проблемы – использование большего количества вычислительных устройств для запуска разных сервисов. В этой статье представлена платформа, с помощью которой можно производить запись видео с различных устройств записи видео, просматривать видео и производить обработку с помощью сторонних программ.

2. Функционал и архитектура

Возможности представленной платформы можно разделить на 2 группы: функционал для взаимодействия с видео и функционал для обработки с помощью различных программ. Пользователь может взаимодействовать с камерами, используя поток rtsp для получения видео. Это видео может быть сохранено в систему для дальнейшего использования. Сохраненное видео

можно просматривать. Обработка видео происходит на удаленных узлах в сети. Данные узлы имеют общего посредника, который взаимодействует с основной частью системы и способен контролировать работу программы, которая обрабатывает видео. Программа для обработки видео включает в себя 2 ограничения: на вход передается файл с видео и после работы программы сохраняется видеофайл для отправки его на основную часть. Большая часть нейронных сетей для работы с видео выполняет эти требования.

Система состоит из 3 частей. Слой интерфейса, который отвечает за показ данных и предоставление пользователю элементов управления системой. Слой сервера, на котором происходит основное взаимодействие между частями системы. Данный слой позволяет работать с камерами, видео, сервисами по обработке видео. Слой, на котором видео обрабатывается, может быть запущен удаленно и вызван из основного узла, если пользователь начал обработку видео. Архитектура системы представлена на рисунке 1.

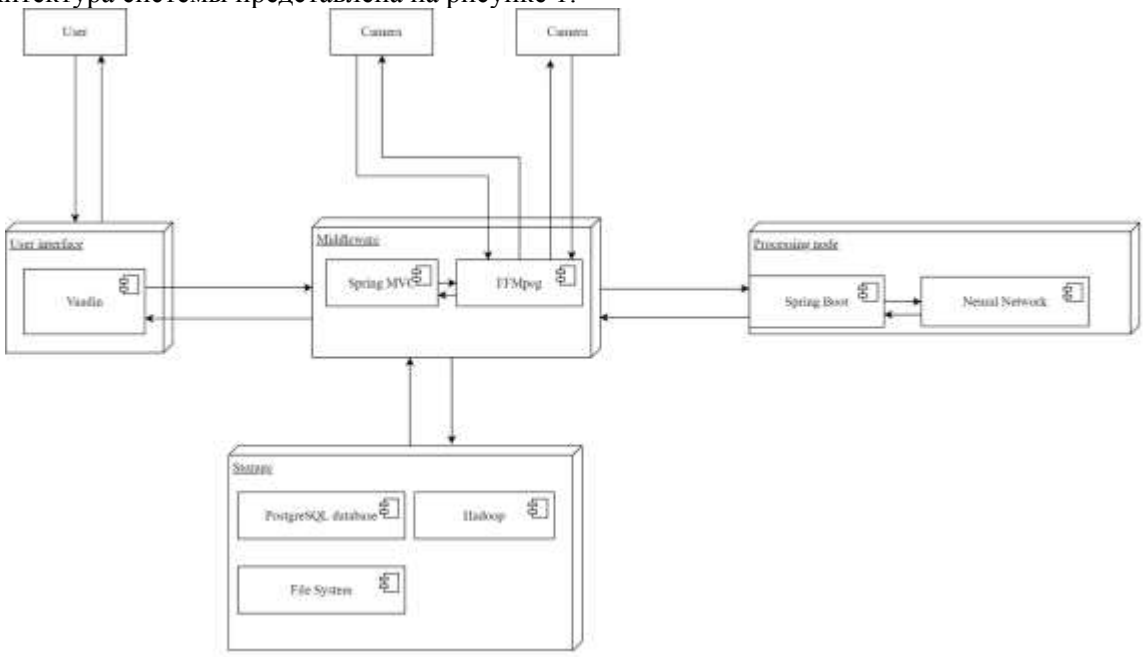

Рисунок 1. Архитектура системы.

Пользователь взаимодействует с пользовательским интерфейсом. Пользовательский интерфейс реализован на фреймворке vaadin. Данный фреймворк позволяет создавать пользовательский интерфейс в веб на языке java. Пользователю доступен личный аккаунт, с которого он имеет доступ к своим камерам. Ему доступна запись видео с подключенных камер, подключение новых камер, запуск обработки сохраненных видео на доступных обработчиках. На рисунках 2, 3 представлен внешний вид пользовательского интерфейса.

Основной слой системы использует Spring MVC, как фреймворк для создания серверной части, и FFMpeg, как программу для взаимодействия с камерами и видео. В данной работе используется реализация FFMpeg и OpenCV на языке java под названием JavaCv [2]. Данная библиотека имеет API для работы с FFMpeg и OpenCV. Данные хранятся в PostgreSQL и файловой системе. Файловая система HDFS позволяет хранить файлы больших размеров с дублированием на распределенные узлы, что уменьшает вероятность потери файлов. Подход MapReduce позволяет обрабатывать видео по частям. Работа с видео в Hadoop может сильно повысить скорость обработки [3]. Взаимодействие с узлами обработки производится через REST API.

Пользователь может просматривать поток видео в прямом эфире. Это реализуется через протокол HLS [4], который отлично подходит для вещания видео, где не требуется очень низкая

задержка. Данный протокол разбивает видео на сегменты, которые можно получить из браузера, чтобы просмотреть. Данные сегменты имеют одинаковую длину и после просмотра сегмента включается ледующий. В случае просмотра видео данный протокол отлично подходит, так как задержка в таком случае может быть. Диаграмма последовательности для просмотра видео в прямом эфире представлена на рисунке 4.

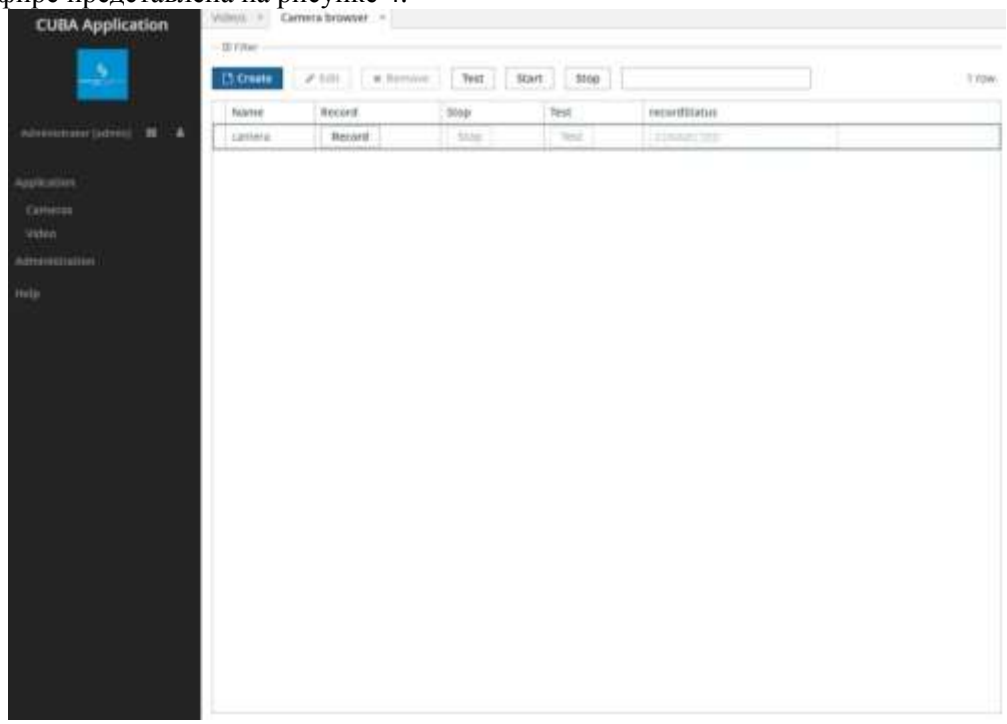

Рисунок 2. Пользовательский интерфейс.

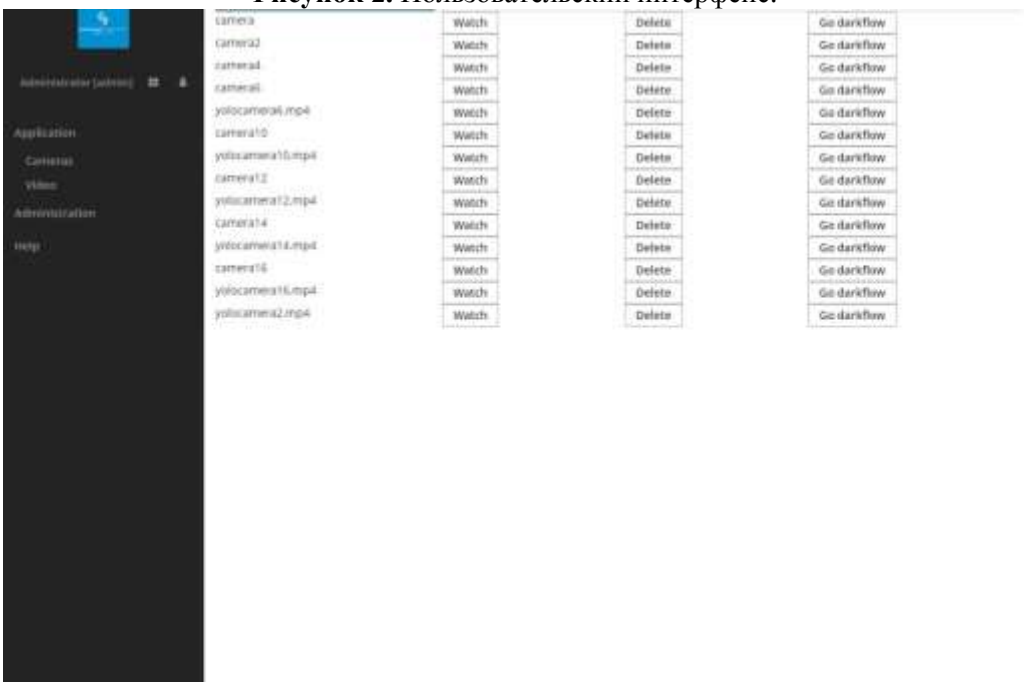

Рисунок 3. Пользовательский интерфейс.

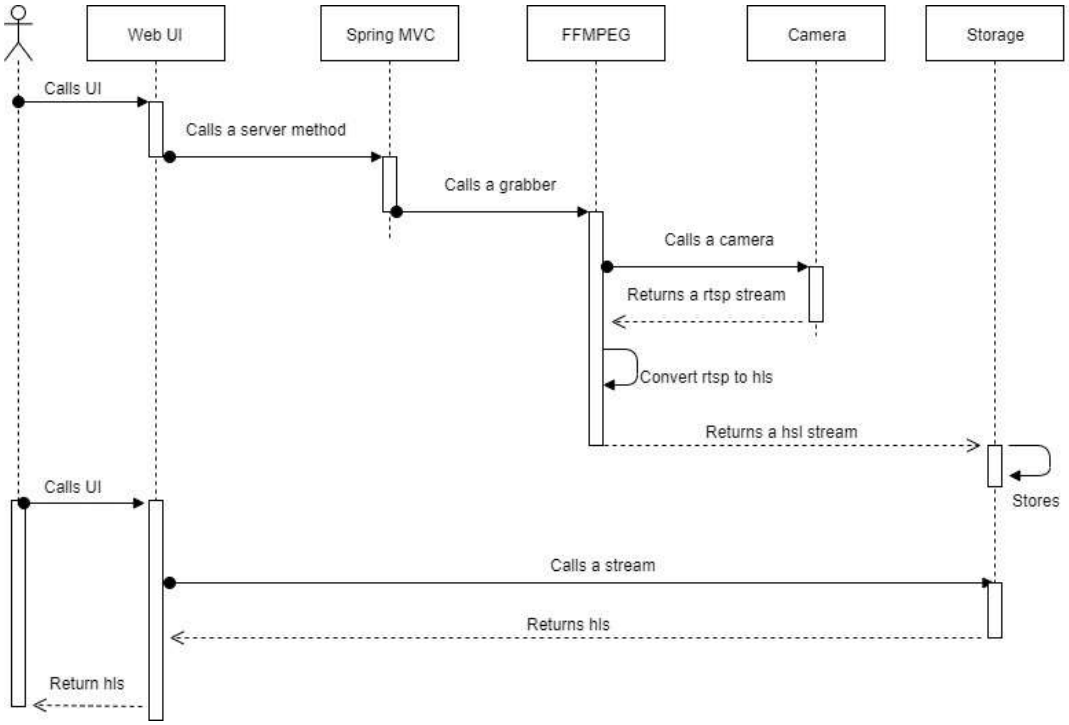

Рисунок 4. Диаграмма последовательностей при просмотре в прямом эфире.

На рисунке 5 показана последовательность вызовов, чтобы пользователь смог смотреть видео с источника. Сначала из интерфейса вызывается метод с сервера. Потом сервер обращается к FFMPEG, который конвертирует поток RTSP в HSL и сохраняет его. Затем из пользовательского интерфейса вызывается поток и показывается в браузере с помощью HTML5.

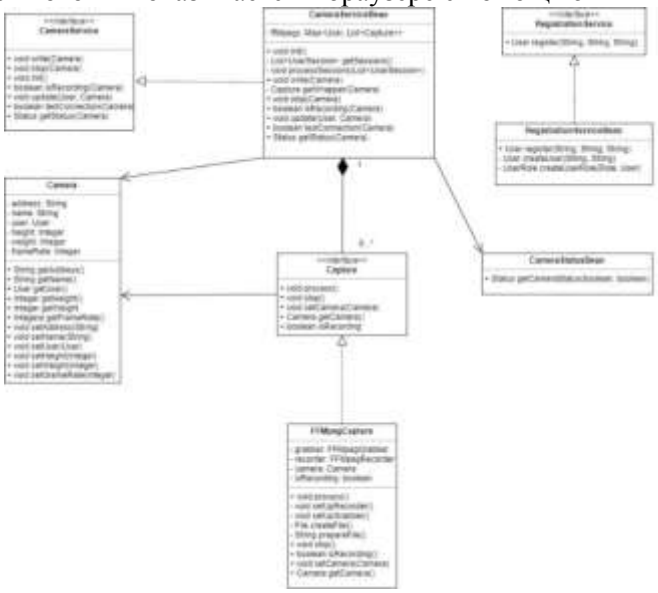

Рисунок 5. Диаграмма классов основной части системы.

Основная часть системы создана на платформе cuba [5]. На рисунке 3 изображена диаграмма классов для основного узла системы.

Система реализована на платформе cuba. Данная платформа предоставляет возможность реализовывать логику в сервисах. В системе один основной сервис: “CameraService”. Он предоставляет методы для взаимодействия с камерами. Из пользовательского интерфейса доступна реализация этого интерфейса. Данный сервис является singleton. Второй сервис, который используется из пользовательского интерфейса: “RegistrationService”. Данный сервис имеет 1 метод, который регистрирует пользователя в системе. Сервис для взаимодействия с камерами вызывает объект, который является реализацией интерфейса “Capture”. Он предоставляет методы для работы с камерами. В системе существует одна реализация, которая использует FFMpeg для работы с камерами.

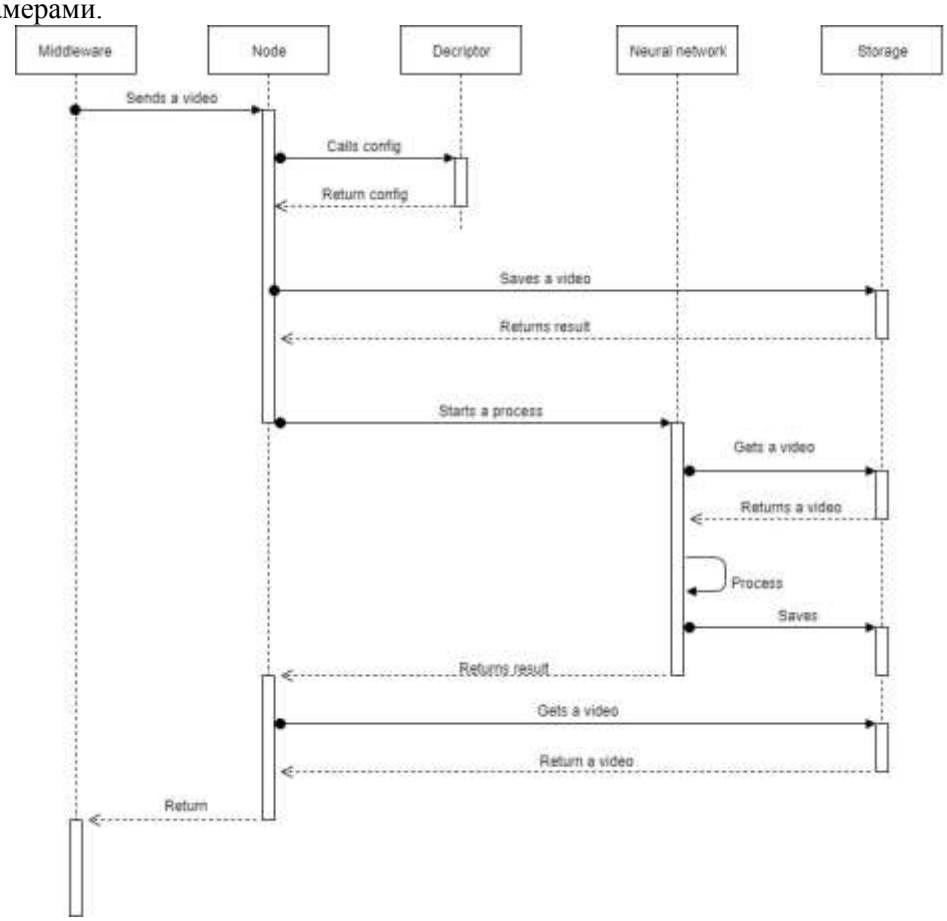

Рисунок 6. Диаграмма последовательностей обработки видео.

3. Микросервис для обработки видео

Пользователю доступен механизм обработки записанных видео с камер. Обработка видео может решать любую задачу с видео, например обнаружение объектов на видео, подсчет объектов, отслеживание и.т.д. Такие обработчики могут быть подключены к системе. Они могут быть запущены на различных вычислительных устройствах, которые могут иметь мощное железо, если обработка является вычислительно сложной. Обработчики часто уже созданы и написаны не на языке java, и, чтобы они были запущены в системе, реализован микросервис, который предоставляет инструменты для запуска программ по обработке видео и способен взаимодействовать с остальной системой. Обработка видео и передача по сети происходит асинхронно, тем самым не нарушая работу других частей системы.

Микросервис, который обрабатывает видео, взаимодействует с системой через Rest API. Он находится в режиме ожидания, до того, как не поступит запрос на обработку. На рисунке 6 изображена диаграмма последовательностей обработки видео.

После отправки видео на сервер обработки используется дескриптор, в котором есть необходимая информация. Потом видео сохраняется и запускается процесс с нейронной сетью. Нейронная сеть получает видео и начинает обработку. Когда сеть завершит обработку, видео сохранится, и сервер отправит его назад на основной узел.

В качестве программ для обработки видео была взята реализация алгоритма обнаружения объектов на изображениях YOLO [6] и модифицированная реализация данного алгоритма, которая использует платформу TensorRT для увеличения скорости обработки видео на видеокартах NVIDIA.

YOLO [7] – это алгоритм классификации и детектирования объектов, использующий сверточные нейронные сети для решения задачи. Такой тип нейронных сетей обладает преимуществом перед другими типами сетей в случае обработки изображений, так как размер сети меньше, чем при использовании полно-связных аналогов. Существует не одна реализация YOLO, которая изначально была создана на сети darknet [6]. В статье была взята реализация, которая называется darkflow [7]. Данная реализация использует фреймворк tensorflow и написана на языке python.

TensorRT [8-9] – это платформа для ускорения искусственных нейронных сетей. Данная платформа разработана компанией NVIDIA. Данная платформа имеет инструменты для работы с многими популярными фреймворками для создания нейронных сетей. Эта платформа использует алгоритмы оптимизации графа сети и аппаратное ускорение, в случае, когда видеокарта способна работать с тензор ядрами. Чтобы получить аппаратное ускорение, нужно иметь видеокарту, которая поддерживает тензор ядра. При использовании аппаратного ускорения можно использовать числа с плавающей точкой половинной точности. Числа с плавающей точкой одинарной точности представляются как FP32, а с половинной FP16.

Применение платформы TensorRT с darkflow способно ускорить обработку видео примерно на 30% [1], не теряя при этом точности классификации.

4. Экспериментальные исследования

Разработанная система была запущена на 1 устройстве. Использовалась 1 камера для получения видео. Обработка видео произведена на 2 различных узлах. Обработано видео с помощью реализации YOLO darkflow и ее модификации, которая использует TensorRT. В таблице 1 представлены характеристики устройств, которые обрабатывали видео.

Таблица 1. Характеристики компьютеров.

Видеокарта	Процессор	Объем памяти, Гб
NVIDIA GeForce GTX 950	Intel Core i5-6500	8
NVIDIA GeForce GTX 2080 TI	Intel Core i7-9700K	64

После обработки пользователь может просмотреть обработанное видео на своей странице. На рисунке 7 показан кадр без обработки нейронной сетью. На рисунке 8 представлен кадр, обработанный нейронной сетью, которая находит людей.

В таблицах 2, 3, 4 показаны значения FPS при обработке на узлах на процессоре, видеокарте и видеокарте вместе с TensorRT.

Обработка на процессоре очень медленная. В этом случае видеокарта способна обрабатывать видео намного быстрее. Обработка на видеокарте NVIDIA GeForce 2080 TI без использования TensorRT быстрее, чем на видеокарте NVIDIA GeForce 950, в 3.62 раза. Использование TensorRT

без аппаратного ускорения позволяет увеличить скорость обработки в 1.09 раза. Использование аппаратного ускорения увеличивает скорость обработки в 1.41 раза.

Рисунок 7. Кадр до обработки.

Рисунок 8. Кадр после обработки.

Таблица 2. FPS на процессоре.

Процессор	FPS
Intel Core i5-6500	2.86
Intel Core i7-9700K	4.21

Таблица 3. FPS на видеокарте.

Видеокарта	FPS
NVIDIA GeForce GTX 950	9.78
NVIDIA GeForce GTX 2080 TI	35.42

Таблица 4. FPS на видеокарте NVIDIA GeForce 2080 TI с TensorRT.

Режим	FPS
FP32	38.69
FP16	49.81

5. Заключение

В рамках данной работы была разработана система для видеонаблюдения. Система способно производить запись и хранение видео с различных камер. Данное видео может быть обработано различными нейронными сетями. Обработка видео может выполняться на удаленных узлах, которые используют мощные вычислительные устройства. Данная система может автоматизировать анализ видео с различных камер, которые постоянно записывают видео. Запущен сервис для обработки видео с реализацией алгоритма для обнаружения объектов YOLO и модификацией, которая использует TensorRT. Запуск произведен на 2 устройствах. На 1 компьютере использована реализация YOLO с TensorRT. TensorRT увеличила скорость обработки в 1.41 раза.

6. Благодарности

Работа выполнена в рамках государственного задания по теме FSSS-2020-0017 при частичной поддержке РФФИ: проект № 17-29-03112 офи_м и проект № 19-29-01235 мк».

7. Литература

[1] Stepanenko, S. Using high-performance deep learning platform to accelerate object detection / S. Stepanenko, P. Yakimov // Proceedings of Information Technology and Nanotechnology. Data Science – Samara, 2019. – Vol. 4. – P. 354-360.
[2] Репозиторий на сайте github.com [Электронный ресурс]. – Режим доступа: https://github.com/bytedeco/javacv (01.11.2019).
[3] Tan, H. An approach for fast and parallel video processing on Apache Hadoop clusters / H. Tan, L. Chen // IEEE International Conference on Multimedia and Expo (ICME), 2014. – P. 1-6.
[4] Документация hls [Электронный ресурс]. – Режим доступа: https://developer.apple.com/streaming/ (01.11.2019).
[5] Официальны сайт cuba [Электронный ресурс]. – Режим доступа: https://www.cuba-platform.ru/ (10.11.2019).
[6] YOLO: Real-Time Object Detection [Электронный ресурс]. – Режим доступа: https://pjreddie.com/darknet/yolo/ (01.11.2019).
[7] Redmon, J. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon, S. Divvala, R. Girshick, A. Farhadi // You Look Only Once, 2015. – 10 p.
[8] Официальный сайт TensorRT [Электронный ресурс]. – Режим доступа: https://developer.nvidia.com/tensorrt (01.11.2019).
[9] TensorRT integration speeds up tensorflow inference [Электронный ресурс]. – Режим доступа: https://devblogs.nvidia.com/tensorrt-integration-speeds-tensorflow-inference/ (01.11.2019).

Development of a cloud platform for gathering, storing and analysis of video data

S.O. Stepanenko[1], P.Y. Yakimov[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. Today video analysis is very relevant. Amount of video is being increased and there is a need to perform it. Very often there are a number of video data sources on which video is being continuously recorded. This article presents an implementation of a platform which aggregates video from various sources and processes it with use of different services which are based on artificial neural networks and machine learning methods.