Санкт–Петербургский государственный университет

Сермягин Никита Владимирович

Выпускная квалификационная работа

Отслеживание присутствия людей в видеопотоке

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2017 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Исследование и проектирование систем управления и обработки сигналов»

Научный руководитель:
кандидат технических наук,
доцент
Гришкин Валерий Михайлович

Рецензент:
кандидат физ.-мат. наук, доцент
Степенко Николай Анатольевич

Санкт-Петербург
2021 г.

Содержание

Введение **4**

Постановка задачи **5**

Обзор литературы **6**

Глава 1. Методы дектирования **7**

1.1. Метрики используемые в задаче детектирования 7

1.2. Детектирование на основе HOG и SVM 8

1.3. Детектирование на основе нейронных сетей 9

1.3.1 Faster R-CNN 10

1.3.2 YOLO 11

Глава 2. Детектирование на основе нейронной сети типа YOLO 13

2.1. Модель и архитектура YOLOv4-tiny 13

2.2. Используемые датасеты 17

2.3. Параметры тренировки 18

2.4. Реализация 19

2.5. Внесенные изменения 19

2.6. Результаты 21

Глава 3. Детектирование на основе детектора движения и сверточной нейронной сети **23**

3.1. Детектор движения 24

3.2. Архитектура нейронной сети 24

3.3. Отслеживание объектов 25

3.4. Используемые датасеты 26

3.5. Параметры тренировки 27

3.6. Реализация 28

3.7. Результаты 28

Глава 4. Система детектирования людей в интересующей зоне 29

4.1. Выбор метода для реализации системы 29

4.2. Реализация системы 29

Заключение **31**

Список литературы **31**

Приложение 1. Примеры работы модифицированной YOLOv4-tiny. **34**

Приложение 2. Этап инициализации. Выделение области слежения. **35**

Приложение 3. Этап инициализации. Выделение области детектирования. **36**

Приложение 4. Этап отслеживания появления людей в интересующей зоне. **37**

Введение

На сегодняшний день стали популярны и широко распространены алгоритмы цифрового анализа изображений. Они могут быть применены в самых различных сферах деятельности. Как в повседневной жизни человека, например, для создания автопилота транспортных средств или в системах видеонаблюдения, так и в более специализированных отраслях. Каждая отрасль имеет свои специфические особенности и требует особого подхода для решения своих конкретных задач.

В данной работе рассматривается задача детектирования людей в видеопотоке. Целью детектирования является отслеживание присутствия искомого объекта и определение его местоположения, в случае если присутствие было зафиксировано. Данная задача является одной из самых распространенных задач в компьютерном зрении и становится все более актуальной, так как может быть применена в различных областях и, в частности, широко используется в системах видеонаблюдения, например, для контроля за охраняемыми объектами, техникой безопасности, или для получения статистики по посещению торговых предприятий.

Основными сложностями в данной задаче являются большая разнообразность внешности людей и влияние на них окружения (перекрытие другими объектами, плохое освещение, плохая видимость из-за погоды и др.). Также с ростом количества видеокамер, особенно значимой становится вычислительная сложность алгоритма.

Постановка задачи

Конечной целью данной работы является реализация алгоритма для осуществления видеоконтроля интересующей зоны, а именно — отслеживание появления в ней людей. Для этого, соответственно, необходимо решить задачу обнаружения и определения местоположения людей на кадре из видеопотока. Видео берется с камеры, которая является статичной и ведет съемку круглосуточно. Также, от алгоритма требуется высокая скорость работы, чтобы обеспечить обработку последовательности изображений из видеопотока в реальном времени. Для достижения цели были поставлены следующие задачи:

- Провести анализ методов для детектирования объектов;
- Разработать и реализовать алгоритм для детектирования людей;
- Провести экспериментальное исследование реализованного алгоритма;
- На базе разработанного алгоритма реализовать систему для отслеживания появления людей в интересующей области;

Обзор литературы

В процессе работы над ВКР были рассмотрены различные методы применяемые для детектирования объектов или людей. В работе [1] описывается алгоритм детектирования людей на изображении, основанный на гистограмме направленных градиентов (HOG) и методе опорных векторов (SVM). Также были рассмотрены методы, основанные на применении сверточных нейронных сетей. Так, в работе [3] описывается метод, основанный на предположении регионов с их дальнейшей классификацией, а в [4] и [5] о введенных в него модификациях. В [2] подробно описывается модель и архитектура одной из самых быстрых нейронных сетей, разработанных для детектирования объектов на изображении, однако, обладающей меньшей точностью чем сети, основанные на предположении регионов. Но этот недостаток был исправлен и в работе [7] описывается новая версия данной нейронной сети, которая в настоящее время остается одной из самых эффективных в задаче детектирования объектов на изображении.

В работах [6], [8] и [13] рассказывается о методах и типовых архитектурах, позволяющих повысить эффективность алгоритмов и нейронных сетей, применяемых для решения задачи детектирования.

Для обучения нейронных сетей использовались различные датасеты. Для решения задачи классификации тренировка производилась на датасетах [15], [16] и [17], содержащих изображения людей, транспортных средств и других различных классов. А при решении задачи детектирования — на датасетах [9], [10] и [11], первые два из которых широко используются для оценки качества алгоритмов детектирования, а последний сосредоточен на задаче детектирования людей.

Для обучения моделей, основанных на архитектуре YOLOv4-tiny, использовался фреймворк с открытым исходным кодом [12]. Также в процессе работы возникла необходимость детектирования движения в видеопотоке, для этого использовался метод, описанный в [14], который показывает хорошие результаты в условиях изменяющегося фона и не требует большого числа вычислений.

Глава 1. Методы дектирования

Из существующих методов широко используются алгоритмы, основанные на гистограмме направленных градиентов (Histogram of Oriented Gradients, HOG) и методе опорных векторов (Support Vector Machine, SVM) [1]. Но такой подход плохо справляется со сложными ситуациями. Хорошие результаты в детектировании объектов показывают алгоритмы, основанные на нейронных сетях, например YOLO[2] или R-CNN [3]. Но для их применения, в основном, требуется большая вычислительная мощность.

1.1 Метрики используемые в задаче детектирования

Для оценки качества алгоритмов, решающих задачу детектирования используют следующие метрики:

FPS (frames per second) — количество сменяемых кадров за единицу времени.

$Precision = \frac{TP}{TP+FP}$

$Recall = \frac{TP}{TP+FN}$

где TP (true positive) — истинно положительное событие, объект был детектирован и правильно классифицирован; FP (false positive) — ложно положительное событие, объект не был достаточно хорошо заключен в обрамляющее окно или был детектирован несколько раз; FN (false negative) — объект не был детектирован, или детектирован с присвоением неправильного класса.

Precision показывает способность алгоритма отличать рассматриваемый класс от других классов а Recall — способность обнаруживать данный класс вообще.

Для оценки качества нахождения обрамляющих окон используется метрика IoU (Intersection over Union), которая измеряет перекрытие между двумя областями. Полученное IoU сравнивается с некоторым пороговым значением и используется для классификации предсказаний (на True

Positive, False Positive или False Negative).

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}$$

Для каждого класса определяется средняя точность (average precision, AP), полученная из Precision и Recall, и вводится понятие mAP (mean average precision) как среднее всех полученных AP.

1.2 Детектирование на основе HOG и SVM

Алгоритмы данного класса основаны на использовании HOG признаков изображения. Полученное с помощью HOG признаковое описание изображения может быть использовано методами машинного обучения, в частности SVM, для классификации изображений на содержащие и не содержащие искомый объект. Однако это накладывает ограничения на рассматриваемые изображения. Так как чаще всего такие алгоритмы классификации используют признаки фиксированной размерности, то и полученные признаковые описания должны быть одинаковой длины и, следовательно, рассматриваемые изображения также должны иметь одинаковый размер. Далее необходимо свести множество задач классификации к задаче детектирования и для этого чаще всего применяется метод бегущего окна: для поиска объектов некоторого заданного размера рассматриваются области одинакового размера расположенные с некоторым шагом по вертикали и по горизонтали. В случае детектирования объектов разных размеров используется многократное масштабирование изображения.

Таким образом, для детектирования объектов применяются следующие шаги:

1. На изображении с помощью метода бегущего окна последовательно рассматриваются области некоторого заданного размера.

2. Для каждой рассматриваемой области с помощью HOG получают признаковое описание.

3. На основе полученного признакового описания с помощью SVM принимается решение, содержит ли данная область искомый объект.

Методы детектирования на основе HOG и SVM обладают высокой скоростью обработки изображений, однако работают не так точно как методы, основанные на нейронных сетях.

1.3 Детектирование на основе нейронных сетей

В задачах компьютерного зрения часто прибегают к использованию сверточных нейронных сетей, которые эффективно справляются с распознаванием образов. В таких сетях используются сверточные слои, в которых применяется операция свертки — фрагменты исходного слоя поэлементно умножаются на фильтр (ядро свертки), результат суммируется и записывается на соответствующую позицию выходного слоя, фильтр заранее не известен и определяется в процессе обучения. Помимо слоя активации между сверточными слоями часто применяют слои подвыборки (пулинг слои), которые не пересекающимся фрагментам исходного слоя ставят в соответствие одно значение, получаемое при помощи нелинейного преобразования, например функции максимума. Операция свертки направлена на выделение определенных признаков, а операция пулинга на их фильтрацию и, как следствие, уменьшение итогового количества полученных признаков, но с сохранением наиболее важной части информации.

Задачу детектирования объектов на изображении можно разделить на две подзадачи — регрессии (определение координат объекта) и классификации. А существующие алгоритмы, основанные на нейронных сетей, на два типа — двухстадийные и одностадийные.

Первые решают поставленную задачу в два этапа. Одна часть алгоритма генерирует области интереса, на которых потенциально расположены нужные нам объекты, а другая классифицирует, полученные предыдущей частью, фрагменты. К таким нейронным сетям относятся R-CNN, Fast R-CNN[4] и Faster R-CNN[5].

Вторые принимают на вход все изображение целиком и сразу выдают как координаты объектов, так и вероятности их принадлежности к

заданным классам. По такому принципу работают сети семейства YOLO.

1.3.1 Faster R-CNN

Faster R-CNN (Region-based Convolutional Network) — самая быстрая из перечисленных двухстадийных моделей, и является модификацией R-CNN и Fast R-CNN.

Схема работы модели:

1. Из входного изображения с помощью сверточной нейронной сети выделяется карта признаков.

2. На основе карты признаков отдельной нейронной сетью RPN (Region Proposal Network) определяются регионы, которые с высокой вероятностью содержат искомые объекты.

3. Затем карта признаков с полученными регионами с помощью специального RoI (Region of Interest) слоя сжимается и подается в нейронную сеть для классификации объектов в найденных регионах.

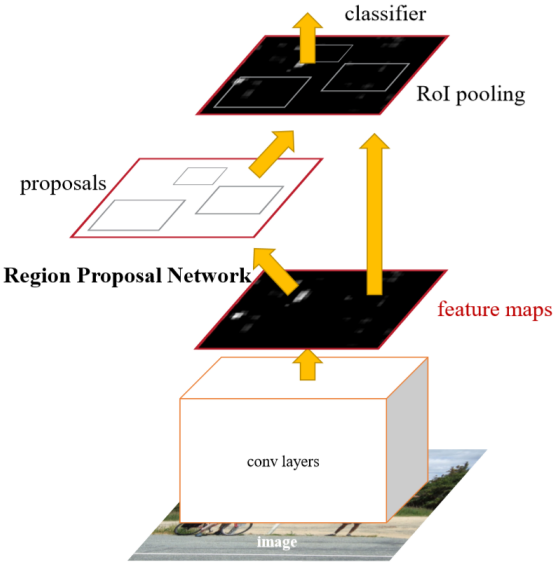


Рис. 1: Схема работы Faster R-CNN.

Данные системы, несмотря на свою большую точность обладают некоторыми недостатками, а именно, они не учитывают окружение искомых объектов и работают значительно медленнее одностадийных систем.

1.3.2 YOLO

YOLO (You Only Look Once) работает иначе, чем двухстадийные алгоритмы и сводит задачу детектирования к регрессии. Вместо генерации регионов кандидатов и последующей их оценки, она накладывает на изображение сетку размера $S \times S$, для каждой ячейки этой сетки модель предсказывает B обрамляющих окон, центр которых расположен внутри этой ячейки, вероятность, что эти окна содержат нужные нам объекты (confidence score), и C вероятностей классов. В итоге вероятности окон перемножаются с вероятностями классов, чтобы получить итоговый результат.

Подобный метод позволяет избежать проверки огромного числа регионов кандидатов, и обеспечивает нейронной сети возможность учитывать контекст при детектировании и распознавании объекта.

Схема работы алгоритма:

1. Исходное изображение масштабируется до заданного размера.

2. Полученное изображение пропускается через сверточную сеть и полученный вектор особенностей размерности $S * S * (B * 5 + C)$ преобразуется к тензору размерности $S \times S \times (B * 5 + C)$.

3. Полученные данные фильтруются, по заранее заданному пороговому значению (threshold), и дополнительно обрабатываются с помощью алгоритма подавления немаксимумов (Non-Max suppression) [6], чтобы избавиться от продублированных обрамляющих окон.

Модель YOLO обладает высокой скоростью обработки изображений, но имеет относительно малую точность. На основе подхода, предложенного в YOLO, было разработано большое количество нейронных сетей улучшающих изначальную модель. Однако на протяжении последних лет архитек-

тура нейронной сети YOLO также развивалась, и на данный момент она остается оптимальной по соотношению точности и скорости обработки.

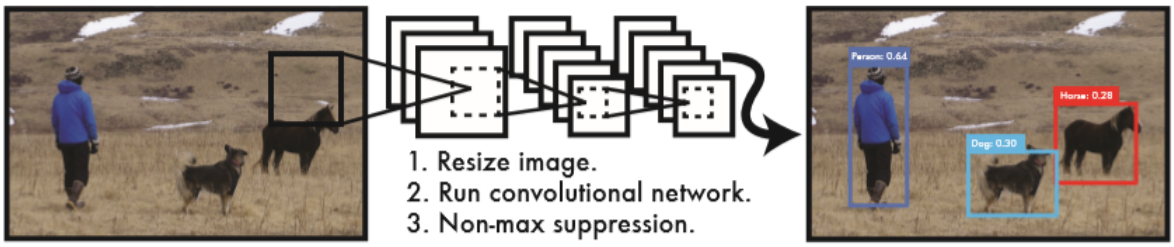


Рис. 2: Схема работы YOLO.

Глава 2. Детектирование на основе нейронной сети типа YOLO

Как уже было сказано ранее, на протяжении последних нескольких лет архитектура нейронной сети YOLO активно развивалась. Были созданы новые улучшенные версии и YOLOv4 [7] — одна из последних на данных момент. А YOLOv4-tiny модификация этой версии, которая позволяет использовать данную модель в реальном времени даже на относительно маломощных системах, тогда как полная версия способна на это только на мощном оборудовании. В отличии от YOLOv4 она имеет упрощенную архитектуру и меньшее количество выходных слоев. В ходе работы, в данную модель были внесены изменения, позволившие повысить точность предсказаний нейронной сети для рассматриваемой задачи.

2.1 Модель и архитектура YOLOv4-tiny

В общей сложности YOLOv4-tiny имеет 21 сверточный слой. В качестве backbone сети используется сеть CSPDarknet53-tiny, главной особенностью которой является использование CSP (Cross-Stage-Partial) блоков. Эти блоки оказались очень эффективными, несмотря на свою относительную простоту, и устроены таким образом, что половина выходного сигнала в них идет по основному пути (создавая больше семантической информации), а другая по обходному пути (сохраняя больше пространственной информации). Примеры CSP блоков представлены на рис. 3.

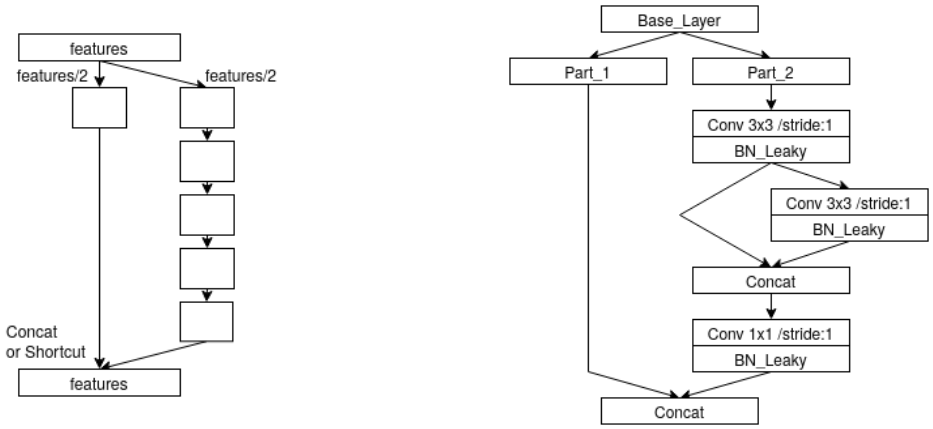


Рис. 3: Примеры простейшего CSP блока (слева) и CSP блока из yolov4-tiny (справа).

Для генерации карт признаков, относящихся к разным масштабам, YOLOv4-tiny использует структуру на основе FPN (Feature Pyramid Network) [8], которую можно разделить на три основные части: восходящий путь (bottom-up pathway), нисходящий путь (top-down pathway) и боковые соединения (lateral connections). Восходящий путь представляет из себя последовательность сверточных слоев с уменьшающейся размерностью, в случае yolov4-tiny эту функцию выполняет backbone сеть CSPDarknet53-tiny. Стоит отметить что карты признаков с разных уровней будут иметь свои преимущества и недостатки. Так, карты признаков с нижних уровней имеют высокое разрешение, но несут мало семантической информации, а карты признаков с верхних – наоборот. Таким образом, восходящий путь в определенных ситуациях может потерять важную информацию при извлечении признаков, например в случае зашумления небольшой, но важной части фоном, из-за сильно обобщенной информации на верхних уровнях.

В нисходящем пути каждая карта признаков вышележащего слоя увеличивается до размеров карты, соответствующего ей слоя в восходящем пути. Также, при помощи боковых соединений, карты признаков соответствующих слоев в восходящем и нисходящем путях складываются, при этом карты из восходящего пути дополнительно проходят через свертку с ядром 1x1, благодаря этому решается проблема затухания важной информации при последовательном переходе по слоям. Таким образом, структура FPN позволяет объединить достоинства как верхних, так и нижних уровней. Схема этого процесса изображена на рис. 4.

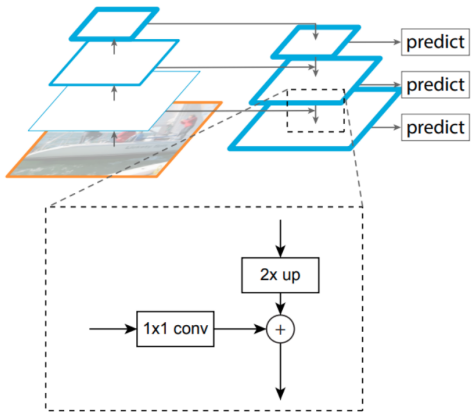


Рис. 4: Структура Feature Pyramid Network.

Для детектирования YOLOv4-tiny использует два выходных слоя, каждый из которых рассчитан на обнаружения объектов определенного размера. Более подробно архитектура сети представлена на рис. 5.

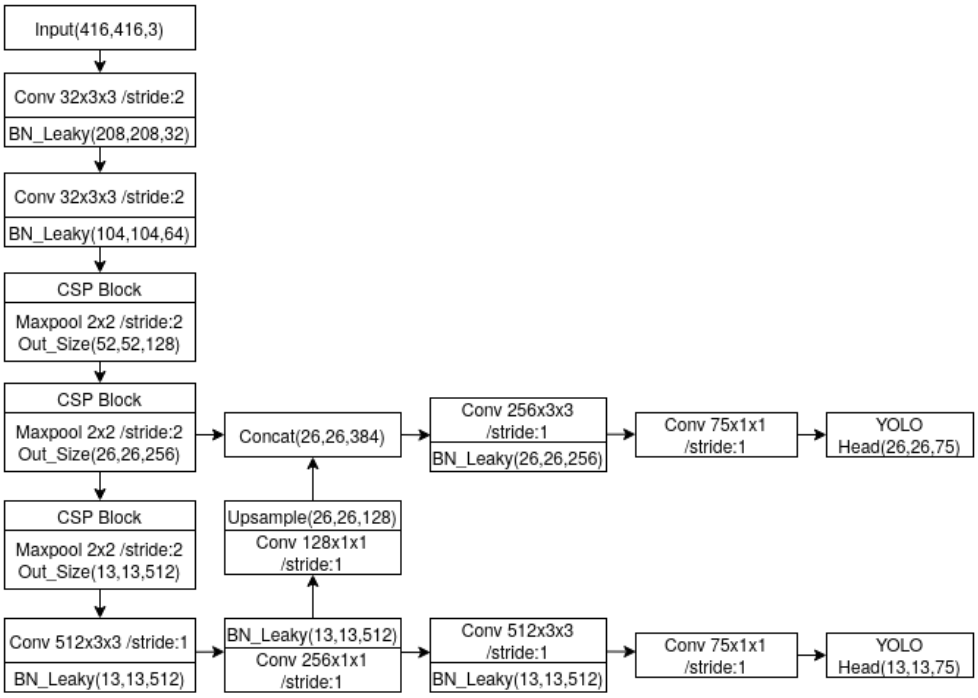


Рис. 5: Архитектура yolov4-tiny для детектирования 20 классов.

Процесс детектирования yolov4-tiny и yolov4 схож с предыдущими моделями. А именно, изображение разбивается на сетку определенного размера и каждая ячейка этой сетки использует некоторое фиксированное количество обрамляющих окон для обнаружения объектов. Соответственно, если центр объекта попадает в центр некоторой ячейки, то обрамляющие окна данной ячейки будут отвечать за детектирование этого объекта.

Далее, чтобы описать используемую в yolov4-tiny функцию потерь, необходимо ввести следующее обозначение:

$$C_{ij} = P_{ij} * IOU_{pr}^{gt}$$

где $P_{ij} = 1$ если объект находится в j-ом окне i-ой ячейки и $P_{i,j} = 0$ иначе, а IOU_{pr}^{gt} это значение IOU между искомым и предсказанным окном.

Тогда итоговое значение функции потерь складывается из следую-

щих значений:

$$loss = loss_1 + loss_2 + loss_3,$$

$$loss_1 = -\sum_{i=0}^{M*N}\sum_{j=0}^{B} W_{ij}^{obj}[\hat{C}_{ij}\log(C_{ij}) + (1-\hat{C}_{ij})\log(1-C_{ij})] -$$

$$\lambda_{noobj}\sum_{i=0}^{M*N}\sum_{j=0}^{B}(1-W_{ij}^{obj})[\hat{C}_{ij}\log(C_{ij}) + (1-\hat{C}_{ij})\log(1-C_{ij})],$$

$$loss_2 = -\sum_{i=0}^{M*N}\sum_{j=0}^{B} W_{ij}^{obj}\sum_{c\in classes}[\hat{p}_{ij}(c)\log(p_{ij}(c)) + (1-\hat{p}_{ij}(c))\log(1-p_{ij}(c))],$$

$$loss_3 = \sum_{i=0}^{M*N}\sum_{j=0}^{B} W_{ij}^{obj}[1 - IOU_{pr}^{gt} + \frac{\rho^2(b, b^{gt})}{d^2} + av],$$

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2,$$

$$a = \frac{v}{(1 - IOU_{pr}^{gt}) + v}$$

где:

M и N - ширина и высота сетки;

B - число обрамляющих окон в каждой ячейке;

$W_{ij} = 1$ если j-ое окно i-ой ячейки отвечает за обнаружение текущего объекта и $W_{ij} = 0$ иначе;

$\hat{C}_{ij}$ и C_{ij} - искомое и предсказанное значение соответственно;

λ_{noobj} - весовой параметр;

$\hat{p}_{ij}(c)$ и $p_{ij}(c)$ - искомая и предсказанная вероятность того, что объект в j-ом окне i-ой ячейки принадлежит к классу c;

w^{gt}, h^{gt}, b^{gt} - искомые ширина, высота и координаты центра обрамляющего окна;

w, h, b - предсказанные ширина, высота и координаты центра;

$\rho^2(b, b^{gt})$ - евклидово расстояние между искомыми и предсказанными координатами центра обрамляющего окна;

d - длина диагонали минимального прямоугольника который содержит как предсказанное так и искомое обрамляющее окно;

2.2 Используемые датасеты

The Pascal VOC Dataset [9]: The Pascal VOC (Visual Object Classes) содержит 20 классов объектов, включая людей. Каждое изображение из датасета содержит аннотации для сегментации, детектирования и классификации. Широко используется для теста производительности в задачах сегментации и детектирования.

COCO Dataset [10]: The MicroSoft COCO (Common Objects in Context) аналогичен предыдущему, однако содержит 80 различных классов объектов.

CrowdHuman Dataset [11]: датасет содержащий 2 класса (head и person) и аннотации к ним для задачи детектирования. Сосредоточен на задаче детектирования людей в толпе.

Таблица 1: Статистика по используемым датасетам

	Pascal VOC	COCO	CrowdHuman
images with person	8102	64115	15000
persons	17784	257252	339565
persons / image	$2,20$	$4,01$	$22,64$

Почти каждый из перечисленных датасетов имеет свой уникальный формат для аннотации данных. Поэтому аннотации из них были преобразованы к формату YOLO разметки изображений. А именно, для каждого изображения из датасета был создан файл формата .txt, содержащий для каждого объекта на изображении строку следующей структуры

<object-class> <x_center> <y_center> <width> <height>

где:

- <object-class> – целое число, идентификатор класса
- <width> <height> – относительная ширина и высота обрамляющего окна
- <x_center> <y_center> – центр обрамляющего окна

2.3 Параметры тренировки

Размер батча (*batch size*): так как нельзя пропустить через нейронную сеть весь датасет за один раз, набор данных разбивается на поднаборы (пакеты или партии) одинакового размера. После того, как все данные из батча обработались нейронной сетью будет получено текущее значение функции потерь. Размер батча влияет на скорость обучения, но также может повлиять на итоговую точность обученной нейронной сети. Если размер батча будет слишком малым, он может оказаться недостаточно репрезентативным в масштабе датасета, и наоборот, при слишком большом размере – слишком обобщенным. Для тренировки нейросети было решено использовать размер батча 64 изображения.

Число итераций: число итераций отображает число батчей, использованных в процессе обучения. В процессе исследования различные варианты нейросети обучались на наборах данных, содержащих двадцать различных классов, или только один класс – "person". В первом случае обучение проводилось на протяжении 250000 итераций, а во втором – 60000.

Скорость обучения (*learning rate*): коэффициент скорости обучения — это параметр, используемый в градиентных методах обучения нейронных сетей и определяющий насколько сильно будут скорректированы веса на каждой итерации. Выбирается в диапазоне от 0 до 1, однако при слишком малых значениях данного параметра метод будет сходиться дольше и может попасть в локальный минимум, а при слишком больших – расходиться. Для обучения использовалось начальное значение 0,00261. В процессе обучения, при достижении определенного числа итераций (200000 и 225000 или 50000 и 55000 для вариантов с двадцатью или одним классом соответственно) это значение умножалось на параметр $\gamma = 0,1$.

Оптимизатор: для уменьшения значения функции потерь в процессе обучения используют оптимизатор. Каждую итерацию он изменяет значение параметров нейронной сети (весов), чтобы достигнуть лучшего результата. В данной работе применялся метод стохастического градиентного спуска (*Stochastic gradient descent, SGD*) с параметрами $momentum = 0,949$ и $decay = 0,0005$.

Аугментация данных: для предотвращения переобучения модели нейронной сети и увеличения размера обучающей выборки часто используется такой метод как аугментация данных, который позволяет создать дополнительные данные из уже имеющихся. К изображениям из датасета в процессе обучения применяются различные преобразования. В данной работе изображение перед подачей в нейронную сеть с некоторой вероятностью модифицировалось следующими способами:

- отражение изображения по горизонтали
- изменение показателя насыщенность (*saturation*) изображения в интервале от $-1,5$ до $1,5$
- изменение показателя экспозиция (*exposure*) изображения в интервале от $-1,5$ до $1,5$
- прибавление величины из интервала от $-0,1$ до $0,1$ к величине тона (hue) всех пикселей изображения

2.4 Реализация

Для тренировки и тестирования различных архитектур нейронных сетей использовался фреймворк с открытым исходным кодом Darknet [12], а для конвертации аннотаций к различным датасетам были написаны скрипты на языке программирования Python.

2.5 Внесенные изменения

Одним из наиболее очевидных способов повысить точность нейронной сети является увеличение количества сверточных слоев, что увеличит глубину нейронной сети. Однако, это также может привести к потери значимой информации при ее передачи между слоями. Поэтому было решено использовать остаточные блоки (residual block) [13] для добавления новых слоев в нейронную сеть. В итоге, перед каждым выходом нейронной сети было добавлено по одному residual блоку, содержащему два сверточных

слоя (рис. 6). Дальнейшее расширение нейронной сети в глубину не сильно увеличивало показатель точности.

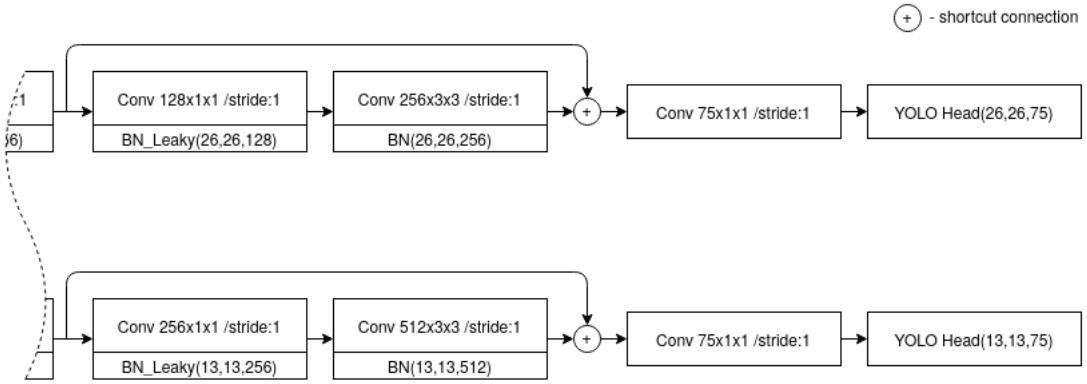


Рис. 6: Пример измененного участка сети при добавлении residual блоков.

Еще одним из очевидных вариантов повысить точность распознования конкретного класса является исключение всех остальных классов. Для этого из обучающей выборки были исключены все лишние классы и число фильтров в последнем сверточном слое изменено на значение $(C + 5) * 3$, где C это количество классов, т. е. на 18.

После было исследовано как размер сетки выходного слоя, а именно число ячеек сетки по горизонтали и вертикали, влияет на точность детектирования людей. Так как можно заметить, что чаще всего люди распределены вдоль некоторой горизонтальной линии, были рассмотрены варианты с увеличением количества ячеек по горизонтали и уменьшением по вертикали, чтобы итоговый размер сетки выходного слоя оставался примерно одинаковым. В таблице 2 приведены получившиеся результаты.

Таблица 2: Результаты на датасете Pascal VOC в зависимости от размера сетки

Grid size $(W \times H)$	13×13	14×12	15×11	16×11	19×9
AP(%) for person	$78,74$	$79,24$	$79,52$	$79,40$	$78,85$

Последним внесенным изменением было добавление третьего выход-

ного слоя, аналогично полной версии YOLOv4. Архитектура полученной нейронной сети представлена на рис. 7.

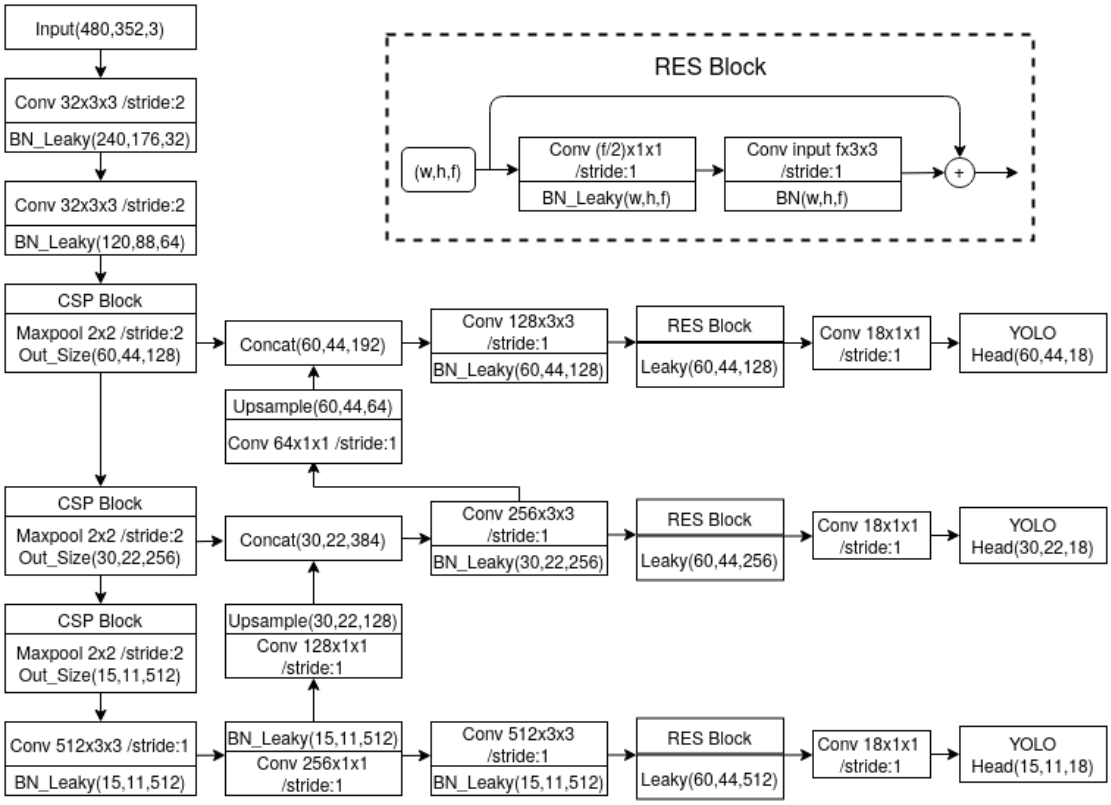


Рис. 7: Архитектура yolov4-tiny для детектирования 20 классов.

2.6 Результаты

В конечном итоге в нейронную сеть были внесены следующие изменения:

– добавление residual блоков перед выходным слоем

– исключение лишних классов

– изменение размера сетки детектирования выходного слоя

– добавление третьего выходного слоя

результаты представлены в таблице 3.

Таблица 3: Итоговые результаты

Model of YOLOv4-tiny	mAP(%) on Pascal	AP(%) for person on Pascal	AP(%) for person on COCO & CrowdHuman	FPS*
original (20 classes)	$60,54$	$72,26$	–	460
with added res-blocks (20 classes)	$62,77$	$73,80$	–	425
with added res-blocks (only person class)	–	$78,74$	–	425
with added res-blocks grid size 15×11 (only person class)	–	$79,52$	–	433
with added res-blocks grid size 15×11 (only person class) COCO & CrowdHuman	–	$86,50$	$64,02$	433
with added res-blocks grid size 15×11 (only person class) COCO & CrowdHuman 3 output layers	–	$86,77$	$67,51$	397

* значение полученное на nVidia RTX 2070 (без отрисовки результата детектирования)

Для проверки внесенных изменений тренировка в основном производилась на датасете Pascal VOC, т. к. на нем точность на валидационной выборке стабилизировалась гораздо быстрее. После тестирования внесенных изменений, нейронная сеть была обучена на данных из датасетов COCO и CrowdHuman что значительно увеличило точность. Также стоит отметить, что хотя дополнительный выходной слой не сильно повлиял на точность на датасете Pascal VOC, но значительно повысил точность на объединенных данных (с $64,02\%$ до $67,51\%$ по метрике AP). Примеры детектирования в приложении 1.

Глава 3. Детектирование на основе детектора движения и сверточной нейронной сети

В качестве одного из способов детектирования людей в видеопотоке был предложен алгоритм, использующий комбинацию из детектора движения и сверточной нейронной сети, состоящий из двух этапов:

1. Выделение из кадра с помощью детектора движения областей интереса.

2. Классификация полученных областей с помощью нейронной сети.

Позже, из-за появлявшихся краткосрочных ложных срабатываний в алгоритм был добавлен дополнительный этап с отслеживанием объектов, классифицированных как человек, для фильтрации по времени нахождении объекта в видеопотоке.

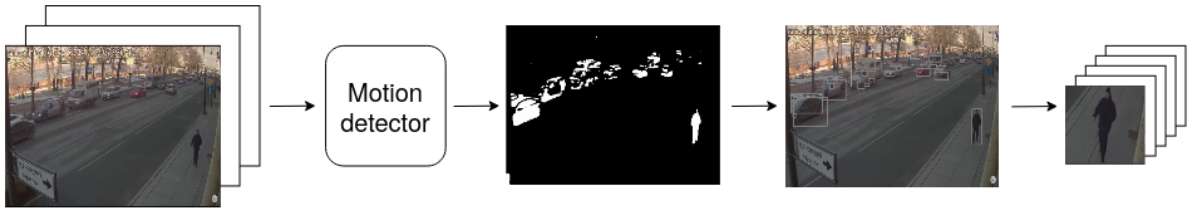


Рис. 8: Этап выделения областей интереса.

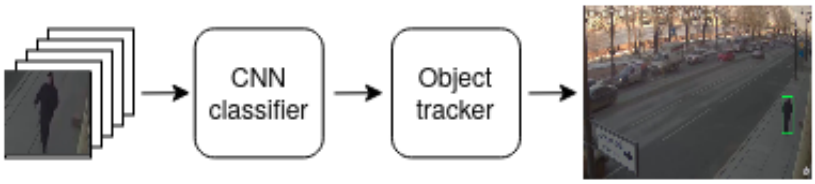


Рис. 9: Этап классификации и фильтрации полученных областей интереса.

3.1 Детектор движения

Для детектирования движения был выбран алгоритм адаптивного вычитания фона (Adaptive Background Subtraction) [14], так как он не требует большого числа вычислений и хорошо справляется с изменяющимся со временем фоном.

Идея алгоритма заключается в получении фонового изображения путем вычисления среднего значения N последних кадров для последующего вычитания из него текущего кадра и сравнения с некоторым пороговым значением. Но, чтобы не хранить в памяти предыдущие кадры и ускорить вычисления, фон считают рекурсивно по формуле

$$B_{i+1}(x, y) = (1 - \alpha)B_i(x, y) + \alpha I_i(x, y),$$

где α — параметр, определяющий, насколько быстро новая информация заменит старую, а $B_i(x, y)$ и $I_i(x, y)$ — значения пикселей (x, y) i-го фона и кадра соответственно.

Дополнительно, чтобы избавиться от различного шума в видеопотоке, к каждому кадру было применено размытие по Гауссу.

3.2 Архитектура нейронной сети

Для классификации объектов используется сверточная нейронная сеть, состоящая из трех сверточных и трех полносвязных слоев. В сверточных и полносвязных слоях использовалась функция активации ReLU, а к выходному тензору была применена сигмоидная функция активации. На вход сеть принимает изображение в формате RGB и выдает число в диапазоне от 0 до 1 — вероятность того, что на изображении человек. Такую архитектуру было решено использовать, чтобы минимизировать количество параметров нейронной сети. Более подробно архитектура представлена на рис. 10.

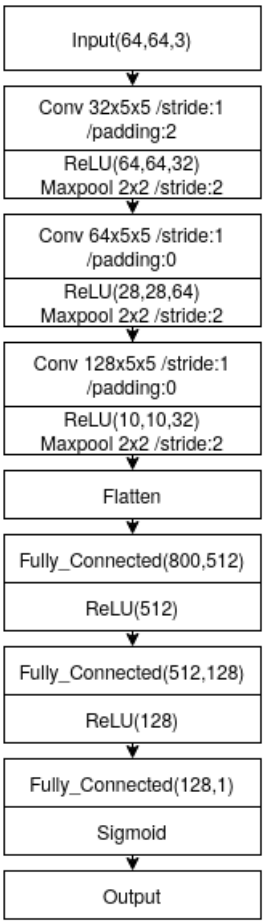


Рис. 10: Архитектура полученной нейронной сети.

3.3 Отслеживание объектов

Для слежения за объектами, появившимися в видеопотоке и классифицированными как человек, используется алгоритм сравнения евклидова расстояния между координатами центра полученных обрамляющих окон с координатами центра обрамляющих окон из предыдущих кадров. Алгоритм работает по следующей схеме:

Algorithm Object tracker with Euclid

$CenterPointsWithIds \leftarrow \{\}$
$DetectedCount \leftarrow 0$
while *Get DetectedBoxes* **do**
    for $(bx,\ by)$ *in DetectedBoxes* **do**
        $SameObjectDetected \leftarrow False$
        for $(cx,\ cy,\ id)$ *in Center Points* **do**

Algorithm Object tracker with Euclid (continuation)

```
            if EuclidDistance((cx, cy)), (bx, by)) ≤ treshold then
                CenterPointsWithIds[id] ← (bx, by, id)
                SameObjectDetected ← True
                break
            end if
            if SameObjectDetected = False then
                id = DetectedCount
                DetectedCount ← DetectedCount + 1
                CenterPointsWithIds[id] ← (bx, by, id)
            end if
        end for
    end for
end while
```

3.4 Используемые датасеты

Для обучения и проверки нейронной сети был собран датасет, состоящий из нескольких публичных.

INRIA Person Dataset [15]: датасет, содержащий изображения людей в различном окружении и аннотации к ним для задачи детектирования. Среди всех людей, присутствующих на изображении, выделены только те, чья высота в пикселях > 100. Это может вызвать проблемы при использовании в задаче детектирования, однако, никак не отразится на задаче классификации.

Vehicle Image Database [16]: датасет содержит изображения транспортных средств, снятых с разных ракурсов, и изображения, извлеченные с фотографий дорог, но не содержащие транспортные средства.

CIFAR-100 Dataset [17]: датасет для задачи классификации изображений, содержит 100 различных классов, сгруппированных в 20 суперклассов.

Из датасета INRIA Person Dataset были получены изображения, содержащие людей, полностью или частично, и изображения фона. А из датасетов Vehicle Image Database и CIFAR-100 — изображения, содержащие транспортные средства или различных животных. Итоговый размер датасета составляет 8700 изображений, половина из которых содержит людей.

На рис. 11 и рис. 12 приведены примеры из получившегося набора.

Рис. 11: Положительные примеры

Рис. 12: Отрицательные примеры

3.5 Параметры тренировки

Размер батча: 32

Число итераций: 10000

Скорость обучения: было выбрано начальное значение равное 0,001 и каждые 3000 итераций это значение умножалось на параметр $\gamma = 0,1$

Оптимизатор: Adam

Аугментация данных: перед подачей в нейронную сеть изображение с некоторой вероятностью модифицировалось следующими способами:

- отражение изображения по горизонтали
- обрезание части изображения и масштабирование до исходного размера
- изменение показателя насыщенность (*saturation*) изображения в интервале от -1,5 до 1,5
- прибавление величины из интервала от –0,1 до 0,1 к величине тона (hue) всех пикселей изображения

3.6 Реализация

Алгоритм реализован на языке программирования Python. Для детектора движения использовалась библиотека алгоритмов компьютерного зрения OpenCV и библиотека для работы с многомерными матрицами и массивами NumPy, а для получения областей интереса — метод findContours из OpenCV. Для нейронной сети использовалась библиотека PyTorch.

3.7 Результаты

Нейронная сеть обучалась методом обратного распространения ошибки с использованием функции потерь Binary Cross Entropy Loss. После обучения на протяжении 10000 итераций точность составила 97,27% по метрике Accuracy (отношение правильного количества предсказаний к их общему количеству) на тестовой выборке.

Реализованный алгоритм работает со средней частотой 44,5 кадра в секунду на машине со следующими характеристиками: Ryzen 2700, 8 ГБ оперативной памяти, nVidia RTX 2070 с 8 ГБ видеопамяти. Алгоритм справляется с детектированием людей, в частности, если они частично перекрыты или малы относительно всего изображения. Пример работы алгоритма представлен на рис. 13.

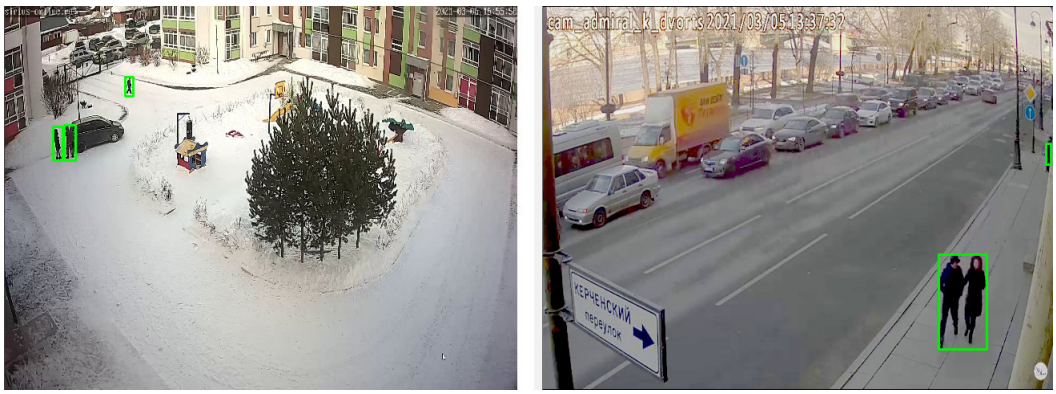


Рис. 13: Примеры работы алгоритма.

Глава 4. Система детектирования людей в интересующей зоне

4.1 Выбор метода для реализации системы

В процессе работы над ВКР были рассмотрены два способа детектирования людей на изображении:

– детектирование на основе YOLOv4-tiny

– детектирование на основе детектора движения и сверточной нейронной сети

Перечисленные методы при сравнении имеют как преимущества так и недостатки. Так, алгоритм с применением детектора движения лучше справляется с детектированием малых или частично перекрытых объектов, но имеет недостаток в виде появляющихся ложных срабатываний, от которых не получилось полностью избавиться. С другой стороны метод на основе YOLO работает более стабильно и имеет более высокую скорость обработки изображений.

В итоге для дальнейшего применения был выбран первый из перечисленных методов, так как ложные срабатывания, появляющиеся при использовании второго метода, в системе видеоконтроля крайне не желательны. Более того, высокая скорость обработки изображений YOLOv4-tiny дает возможность одновременно обрабатывать видео, полученные с нескольких камер, на одном устройстве.

4.2 Реализация системы

После реализации алгоритма детектирования людей, задача по отслеживанию их появления в интересующей зоне становится тривиальной. Т. к. детектирование подразумевает под собой определение местоположения объекта, все что остается сделать — это задать зону и использовать получаемые координаты объектов для проверки.

Для реализации данной системы был создан проект на языке программирования C++. С помощью фреймворка Darknet была создана биб-

лиотека, которая использовалась для применения полученной нейронной сети на основе YOLOv4-tiny в данном проекте. Также использовалась библиотека алгоритмов компьютерного зрения OpenCV для реализации взаимодействия с пользователем и отображения результата работы алгоритмов.

Работа системы состоит из следующих этапов:

1. Этап инициалицазии
    1.1. Выделение зоны, в которой отслеживается появление людей
    1.2. Выделение зоны, в которой производится детектирование объектов
2. Отслеживание появления людей в интересующей зоне

Таким образом, при запуске пользователю предлагается выделить зоны для слежения и детектирования. Такое разделение позволяет выбрать в качестве зоны слежения область произвольной формы, а также избавиться при необходимости от детектирования объектов в ненужных областях. Пример этапа инициализации в приложениях 2 и 3.

После начинается процесс детектирования людей и слежения за интересующей областью. Найденные в области детектирования люди будут помечены синим обрамляющим окном, а найденные в области слежения — красным. Примеры работы в приложении 4.

Заключение

В результате работы были рассмотрены существующие алгоритмы для детектирования объектов на изображении. Был предложен алгоритм для детектирования людей в видеопотоке с использованием детектора движения и сверточной нейронной сети, а также была произведена модификация существующего метода YOLOv4-tiny для улучшения точности в рассматриваемой задаче. После, полученные результаты были проанализированы и выбран наиболее подходящий под поставленную задачу метод, который в дальнейшем использовался для реализации системы отслеживания присутствия людей в интересующей области.

Список литературы

[1] Said Y., Atri M., Tourki R. Human detection based on integral histograms of oriented gradients and SVM // International Conference on Communications, Computing and Control Applications (CCCA). IEEE Trans. Circuits Syst. 2011. P. 1–5.

[2] Redmon J., Divvala S., Girshick R., et al. You only look once: unified, real-time object detection // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Trans. Circuits Syst. 2016. P. 779–788.

[3] Girshick R., Donahue J., Darrell T., et al. Rich feature hierarchies for accurate object detection and semantic segmentation // IEEE Conference on Computer Vision and Pattern Recognition. IEEE Trans. Circuits Syst. 2014. P. 580–587.

[4] Girshick R. Fast R-CNN // IEEE International Conference on Computer Vision (ICCV). IEEE Trans. Circuits Syst. 2015. P. 1440–1448.

[5] Ren S., He K., Girshick R., et al. Faster R-CNN: towards real-time object detection with region proposal networks // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 39. No 6. IEEE Trans. Circuits Syst. 2017. P. 1137–1149.

[6] Bodla N., Singh B., Chellappa R., et al. Soft-NMS — improving object detection with one line of code // IEEE International Conference on Computer Vision (ICCV). IEEE Trans. Circuits Syst. 2017. P. 5562–5570.

[7] Bochkovskiy A., Wang C., Liao H. YOLOv4: optimal speed and accuracy of object detection. // ArXiv, abs/2004.10934. 2020.

[8] Lin T., Dollár P., Girshick R., et al. Feature pyramid networks for object detection // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Trans. Circuits Syst. 2017. P. 936–944.

[9] The PASCAL VOC project [Электронный ресурс]: URL:http://host.robots.ox.ac.uk/pascal/VOC/ (дата обращения: 24.05.21).

[10] COCO 2020 Challenges [Электронный ресурс]: URL:https://cocodataset.org/#home (дата обращения: 24.05.21).

[11] CrowdHuman: A Benchmark for Detecting Human in a Crowd [Электронный ресурс]: URL:https://www.crowdhuman.org (дата обращения: 24.05.21).

[12] Darknet: Open Source Neural Networks in C [Электронный ресурс]: URL:https://pjreddie.com/darknet/ (дата обращения: 24.05.21).

[13] He K., Zhang X., Ren S., et al. Deep residual learning for image recognition // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Trans. Circuits Syst. 2016. P. 770–778.

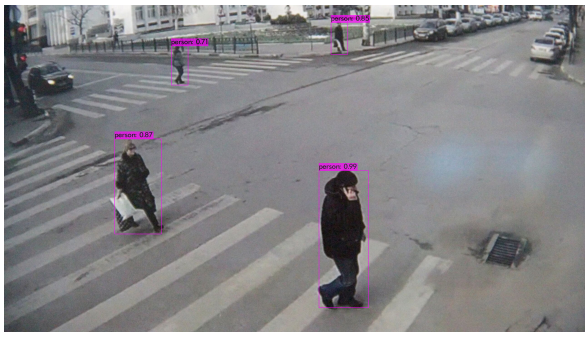
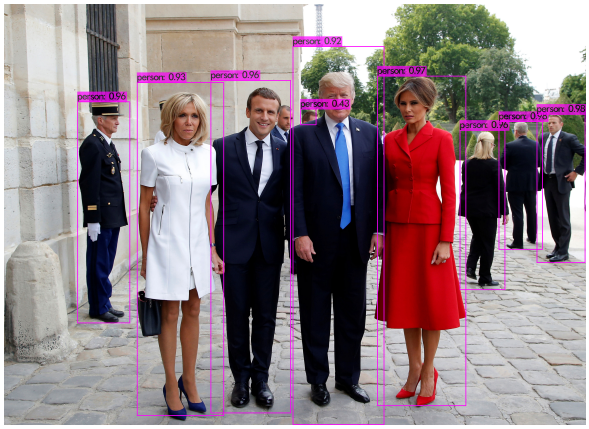
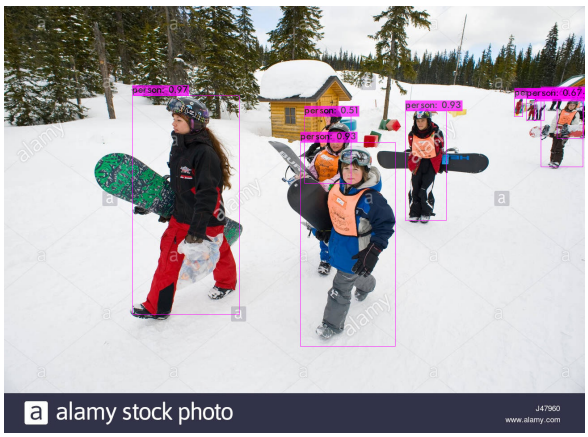
[14] Xiao-jun Tan, Jun Li, Chunlu Liu. A video-based real-time vehicle detection method by classified background learning // World Trans. Eng. Technol. Edu. 2007. Vol 6. No 1. P. 189–192.

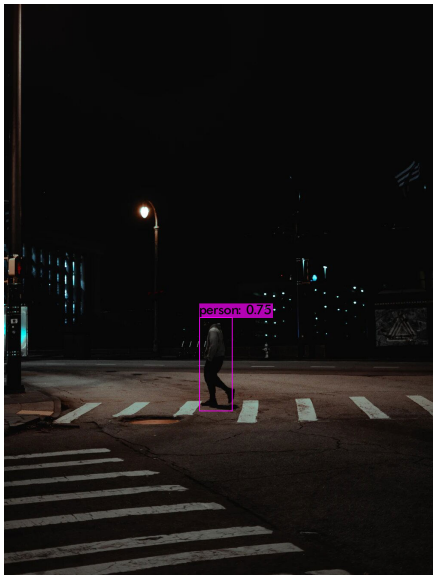
[15] INRIA Person Dataset [Электронный ресурс]: URL:http://pascal.inrialpes.fr/data/human/ (дата обращения: 24.05.21).

[16] Vehicle Image Database [Электронный ресурс]: URL:https://www.gti.ssr.upm.es/data/Vehicle_database.html (дата обращения: 24.05.21).

[17] The CIFAR-100 Dataset [Электронный ресурс]: URL:https://git-disl.github.io/GTDLBench/datasets/cifar-100_datasets/ (дата обращения: 24.05.21).

Приложение 1. Примеры работы модифицированной YOLOv4-tiny.



Изображения обработанные YOLOv4-tiny с измененной архитектурой.

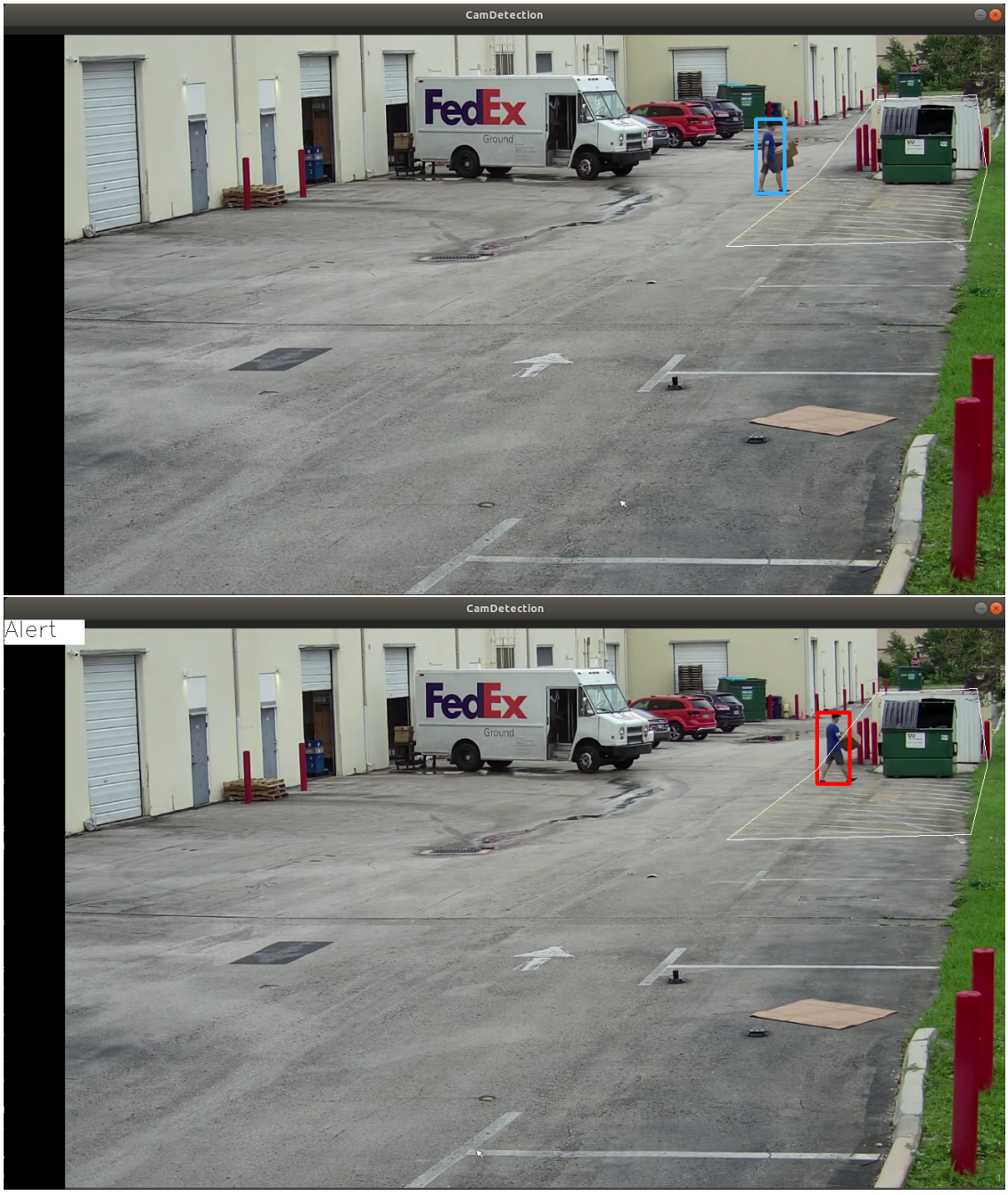
Приложение 2. Этап инициализации. Выделение области слежения.

Процесс выделения зоны интереса в которой будет отслеживаться появление людей.

Приложение 3. Этап инициализации. Выделение области детектирования.

Процесс выделения области детектирования. По умолчанию выделяется минимально возможная область.

Приложение 4. Этап отслеживания появления людей в интересующей зоне.



Детектированный снаружи (сверху) и внутри (снизу) зоны интереса человек.