МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра компьютерных технологий и систем

НИЧИПОРУК РОМАН ОЛЕГОВИЧ

МЕТОДЫ ОБНАРУЖЕНИЯ АНОМАЛЬНОГО ПОВЕДЕНИЯ ЧЕЛОВЕКА НА ВИДЕОПОСЛЕДОВАТЕЛЬНОСТИ НА ОСНОВЕ ЕГО ДВИЖЕНИЯ

Дипломная работа

Научный руководитель:
Старший преподаватель кафедры
компьютерных технологий и сетей
С. В. Шолтанюк

Допущена к защите
«_____»_____20___г.
Заведующий кафедрой компьютерных технологий и систем
доктор педагогических наук, кандидат физико-
математических наук, профессор В.В. Казачёнок

Минск, 2025

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Дипломная работа, 54 страницы, 29 рисунков, 1 таблица, 14 источников.

Ключевые слова: КОМПЬЮТЕРНОЕ ЗРЕНИЕ, МАШИННОЕ ОБУЧЕНИЕ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, VGG16, YOLO, ОПТИЧЕСКИЙ ПОТОК, HOG, ХАРАКТЕРИСТИЧЕСКИЙ ВЕКТОР, SVM.

Объект исследования: поведение человека на видеопоследовательности.

Предмет исследования: эффективность и особенности применения каждого из полученных методов для обнаружения аномального поведения человека.

Цель исследования: постановка и решение задачи обнаружения аномального поведения человека на видеопоследовательности.

Методы исследования: сравнительный анализ полученных решений, анализ производительности полученных алгоритмов, для оценки применимости в различных условиях, проектирование архитектуры и разработка решений системы.

Полученные результаты их новизна: проведено тестирование различных подходов для обнаружения аномального поведения человека. Установлены различия в точности прогнозирования и характеристиках получаемых от методов.

Область возможного практического применения: обеспечение безопасности на улицах, в общественном транспорте с целью выявления потенциальных угроз, исследование поведения человека в различных условиях.

РЭФЕРАТ

Дыпломная праца, 54 старонкі, 29 малюнкаў, 1 табліца, 14 крыніц.

Ключавыя словы: КАМП'ЮТАРНЫ ЗРОК, МАШЫНАЕ НАВУЧАННЕ, ЗВЕРТАЧНЫЯ НЕЙРОННЫЯ СЕТКІ, VGG16, YOLO, АПТЫЧНЫ ПАТОК, HOG, ХАРАКТАРЫСТЫЧНЫ ВЕКТАР, SVM.

Аб'ект даследавання: паводзіны чалавека на відэа.

Прадмет даследавання: эфектыўнасць і асаблівасці прымянення кожнага з атрыманых метадаў для выяўлення анамальных паводзін чалавека.

Мэта даследавання: пастаноўка і рашэнне задачы выяўлення анамальных паводзін чалавека на видеопоследовательности.

Метады даследавання: параўнальны аналіз атрыманых рашэнняў, аналіз прадукцыйнасці атрыманых алгарытмаў, для ацэнкі дастасавальнасці ў розных умовах, праектаванне архітэктуры і распрацоўка рашэнняў сістэмы.

Атрыманыя вынікі іх навізна: праведзена тэставанне розных падыходаў для выяўлення анамальных паводзін чалавека. Устаноўлены адрозненні ў дакладнасці прагназавання і характарыстыках атрыманых ад розных метадаў.

Вобласць магчымага практычнага прымянення: забеспячэнне бяспекі на вуліцах, у грамадскім транспарце з мэтай выяўлення патэнцыйных пагроз, даследаванне паводзін чалавека ў розных умовах.

SUMMARY

Diploma work, 54 pages, 29 pictures, 1 table, 14 sources.

Keywords: COMPUTER VISION, MACHINE LEARNING, CONVOLUTIONAL NEURAL NETWORKS, VGG16, YOLO, OPTICAL FLOW, HOG, FEATURE VECTOR, SVM.

The object of the research: human behavior in a video sequence.

The subject of the research: the effectiveness and application features of each of the obtained methods for detecting abnormal human behavior.

The aim of the research: formulation and solution of the problem of detecting abnormal human behavior in a video sequence.

Research methods: comparative analysis of the solutions obtained, performance analysis of the algorithms obtained, to assess their applicability in various conditions, architecture design and development of system solutions.

The results of the work and their novelty: various approaches have been tested to detect abnormal human behavior. Differences in the accuracy of forecasting and the characteristics obtained from the methods have been established.

Recommendation on the usage: ensuring safety on the streets, in public transport in order to identify potential threats, the study of human behavior in various conditions.

ВВЕДЕНИЕ

Применение алгоритмов компьютероного зрения в наше время становится все более популяным решением в совершено разных сферах жизнедеятельности человека. Уже сейчас трудно представить жизнь и функционирование общества без приборов основанных или использующих некоторых из методов, например, программная стабилизация видео, сегментация и классификация обйектов на изображении.

Одной из самых популярных задач компьютерного зрения является задача обнаружения аномалий в поведении объекта на видеопоследовательности, которая была разобрана в различных исследованиях и статьях.

В контексте моей работы иследованы аномалии поведения человека на видеопоследовательности на основе их движения. Для данной задачи можно выделить несколько различных сфер приложения, различаемых своей направленностью, например:

1. задачи по определению состояния здоровья человека;
2. задачи по определению агрессивного поведения человека;
3. задачи по определению правомерности действий человека на дороге;
4. и множество других задач.

Целью работы является постановка и решение задачи обнаружения аномального поведения человека на видеопоследовательности. В рамках исследования предполагается разработать, протестировать и проанализировать эффективность различных решений и методов, включая подходы основанные на оптическом потоке и современные модели, основанные на сверточных нейронных сетях.

Задачи, решенные в рамках дипломной работы для достижения поставленной цели:

1. рассмотреть и реализовать алгоритм основанный на оптическом потоке;

2. иследовать и применить сверточные нейронные сети в рамказ поставленной задачи;

3. выявить слабые и сильные места методов по полученным данным;

4. выявить, если будет возможно, лучшее из полученных решений.

ГЛАВА 1

Обзор основных методов исследования движения объектов видеопоследовательностей

1.1 Понятие аномалий

Аномалии представляют собой закономерности в данных, которые не соответствуют четко определенному понятию нормального поведения. Обнаружение аномалий представляет собой задачу выявления закономерностей в данных, которые отклоняются от ожидаемого поведения.

Аномальные события определяются как события, частота возникновения которых мала. События могут касаться толпы, как пример, при уличной драке, или при возникновении драки, так и индивидуально человека, например, падение, движение на четвереньках, нахождение в положении лежа на тротуаре и другое. В данный момент методы обнаружения аномального поведения на видеопоследовательности можно разделить на две большие группы [6]:

1. Траекторные методы.

2. Необъектные центрированные методы.

Из всего вышеперечисленного получаем математическое определение аномального и нормального события:

Пусть у нас есть несколько различных множеств N_1, N_2, N_3, N_n, где $n \in \mathbb{N}$, которые определяют классы нормальных событий на множестве данных D. С учетом этого получаем:

Событие h называется нормальным тогда и только тогда, когда выполняется условие:

$$h \in N_i, \quad n \in \mathbb{N}. \tag{1.1}$$

Выполнение условия 1.1 говорит о том, что для того чтобы классифицировать событие как нормальное, необходимо, чтобы оно относилось к одному из классов N_i.

Событие g называется аномальным тогда и только тогда, когда выполняется каждое из следующих условий:

$$g \notin N_i, \quad n \in \mathbb{N}, \tag{1.2}$$

$$g \in D. \tag{1.3}$$

Выполнение условий 1.2 и 1.3 говорит о том, что для того чтобы классифицировать событие как аномальное, необходимо, чтобы оно не относилось ни к одному из классов N_i, и при этом относилось ко всему набору данных.

Например, в простом двумерном наборе данных можно выделить две нормальные области, обозначенные как H_1 и H_2, в которых сосредоточено большинство наблюдений.

Точки, находящиеся на значительном расстоянии от этих областей, такие как O_1 и O_2, а также точки в регионе O_3, считаются аномалиями. Это связано с тем, что их поведение не соответствует общей структуре данных и отличается от наблюдаемых закономерностей в нормальных областях.

Данный пример можно увидеть на рис. 1.2.

Данный подход позволяет наглядно идентифицировать аномалии даже в визуально представленных данных, что важно для анализа и понимания природы отклонений.

1.2 Типы движения объектов на видео

Движение объекта, которое наблюдается на видеопоследовательности, обычно представляет собой один из четырех основных классов или их комбинации:

1. Смещение объекта в сторону, рис. 1.1.

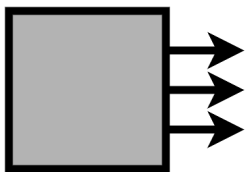


Рисунок 1.1 Смещение в сторону

2. Смещение по оптической оси видеокамеры, рис. 1.3.

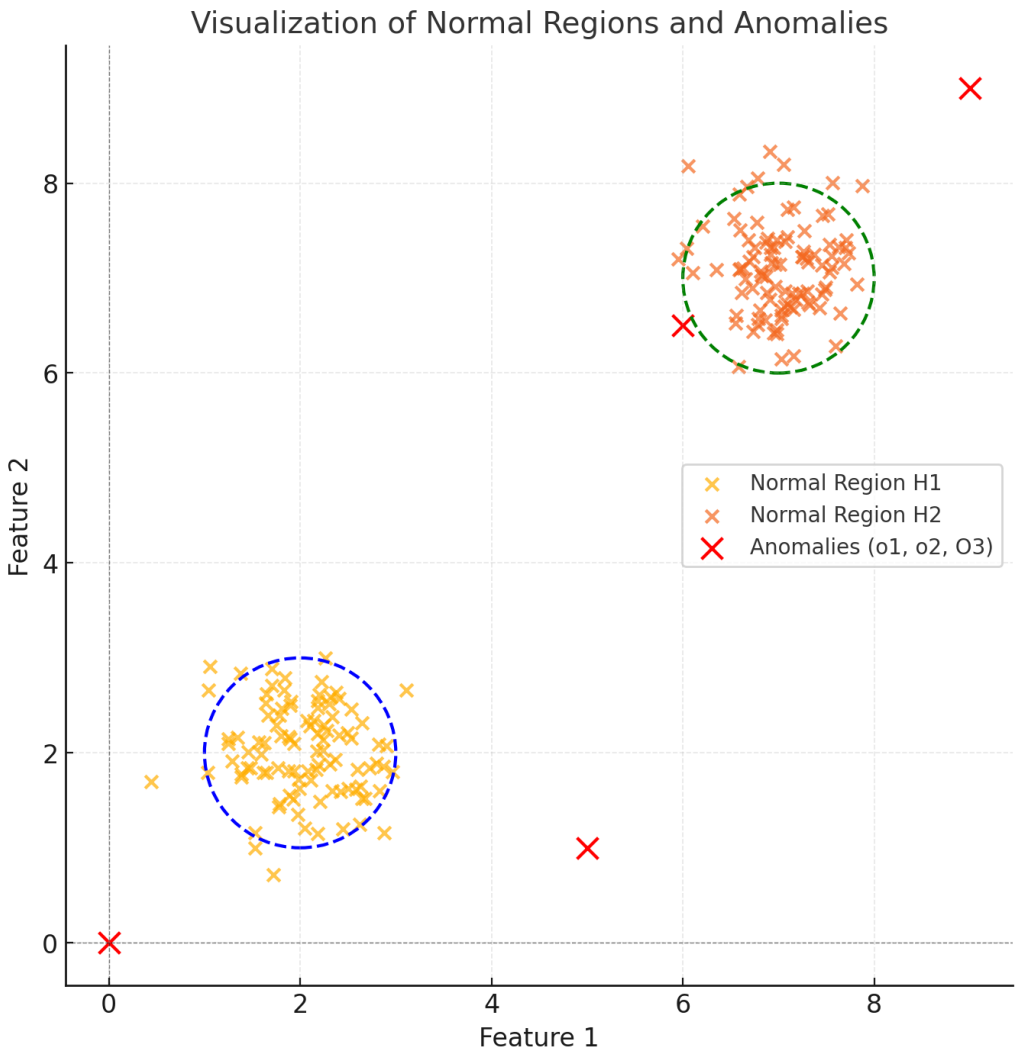


Рисунок 1.2 Визуализация нормальных областей и аномалий

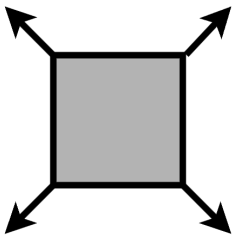


Рисунок 1.3 Смещение по оптической оси видеокамеры

3. Вращение объекта вокруг оптической оси, рис. 1.4.

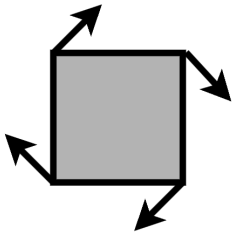


Рисунок 1.4 Вращение вокруг оптической оси

4. Вращение объекта вокруг собственной оси перпендикулярно оптической оси, рис. 1.5.

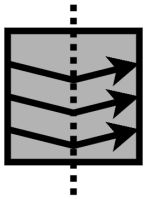


Рисунок 1.5 Вращение вокруг своей оси перпендикулярно оптической оси

С течением времени объект может переходить из одного класса движения в другой. Он переходит из состояния спокойствия в состояние движения, если выполняется хотя бы одно из следующих условий:

$$(x_{k-1}, x_{k-1}) \neq (x_k, y_k), \tag{1.4}$$

$$F_{k-1} \neq F_k, \tag{1.5}$$

$$S_{k-1} \neq S_k, \tag{1.6}$$

где (x_k, x_k) - координаты точки объекта в момент времени k, F_k - характеристика формы в момент времени k, S_k - характеристика размера объекта в момент времени k.

Выполнение условия (1.4) говорит о том, что если происходит изменение определенного числа координат с течением времени, то объект находится в движении.

Выполнение условия (1.5) говорит о том, что если происходит изменение формы объекта с течением времени, то объект находится в движении.

Выполнение условия (1.6) говорит о том, что если происходит изменение размеров объекта с течением времени, то объект находится в движении.

С течением времени объект может переходить из одного класса движения в другой. Он изменяет свой класс из состояния спокойствия в состояние движения, если выполняется хотя бы одно из следующих условий: Процесс перехода объекта из состояния движения в состояние покоя, происходит после прекращения движения. Следовательно, объект переходит в состояние спокойствия, если верны следующие условия:

$$(x_{k-1}, x_{k-1}) = (x_k, y_k), \tag{1.7}$$

$$F_{k-1} = F_k, \tag{1.8}$$

$$S_{k-1} S_k, \tag{1.9}$$

где (x_k, x_k) - координаты точки объекта в момент времени k, F_k - характеристика формы в момент времени k, S_k - характеристика размера объекта в момент времени k.

Выполнение условия (1.7) говорит о том, что достаточное число координат не изменяется с течением времени, значит, объект находится в состоянии спокойствия.

Выполнение условия (1.8) говорит о том, что если не происходит изменение формы объекта с течением времени, то объект находится в состоянии покоя.

Выполнение условия (1.9) говорит о том, что если не происходит изменение размеров объекта с течением времени, то объект находится в состоянии покоя.

1.3 Поиск объектов с помощью гистограммы направленных градиентов и метода опорных векторов

Пара гистограммы направленных градиентов(HOG) [11] и метода опорных векторов(SVM) [10] представляет собой классическую пару алгоритмических методов компьютерного зрения, которая используется для детекции объектов как старая альтернатива оптическому потоку.

1.3.1 Гистограмма направленных градиентов

Классически дескрипторы принимают цветное изображение, то есть трехмерные векторы размеров $n \times m \times l$, а в качестве выходных данных отдают

соответственно вектор длины k. Дескриптор функции является представлением изображения или его части, которое упрощает его, отбрасывая неважную в рамках поиска информацию. Проще говоря, дескриптор извлекает интересующие пользователя особенности из кадра.

Гистограмма направленных градиентов является дескриптором в классическом его определении в рамках компьютерного зрения. В качестве входных данных приходит матрица размеров $64 \times 128 \times 3$, а возвращает вектор длины 3780. Идея заключается в том, что величина градиентов велика вокруг краев и углов объектов (областей с резким изменением интенсивности) [2].

Пример применения гистограммы направленных градиентов на рис. 1.6.

Для расчёта дескриптора гистограммы направленных градиентов выполняются следующие этапы:

1. Изображение преобразуется в оттенки серого по формуле:

$$i(x, y) = 0.2126r(x, y) + 0.7152g(x, y) + 0.072b(x, y), \tag{1.10}$$

   где y представляет собой значение интенсивности в полученном сером изображении, а r, g, b соответственно значения красного, зеленого и синего цветов на исходном кадре.

   Делается это для того, чтобы анализировать только интенсивность изображения.

2. Вычисляются величины градиента по x и y, в случае *opencv* делается это при помощи оператора Собеля с размером ядра 1:

$$g_x(x, y) = i(x + 1, y) - i(x - 1, y), \tag{1.11}$$

$$g_y(x, y) = i(x, y + 1) - i(x, y - 1). \tag{1.12}$$

   Используя $g_x(x, y)$ и $g_y(x, y)$ вычисляется величина и направление градиента:

$$g(x, y) = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}, \tag{1.13}$$

$$\phi(x, y) = arctg\frac{g_y(x, y)}{g_x(x, y)}. \tag{1.14}$$

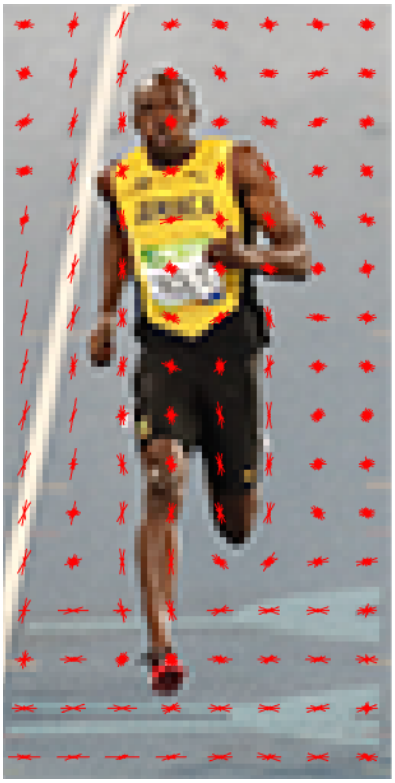


Рисунок 1.6 Визуализация гистограммы ориентированных градиентов

Ниже, на рис. 1.7, представлен пример результата с данного этапа.

3. Изображение делится на сетку ячейками, обычно размером $8x8$ пикселей [2].

4. На втором этапе для каждого пикселя в каждом блоке были вычислены модуль и направление градиента. Теперь для каждого из блоков формируется гистограмма направлений градиентов, состоящая из 9 интервалов:

$$\phi_i \in (0^\circ, 20^\circ, 40^\circ, 60^\circ, 80^\circ, 100^\circ, 120^\circ, 140^\circ, 160^\circ). \tag{1.15}$$

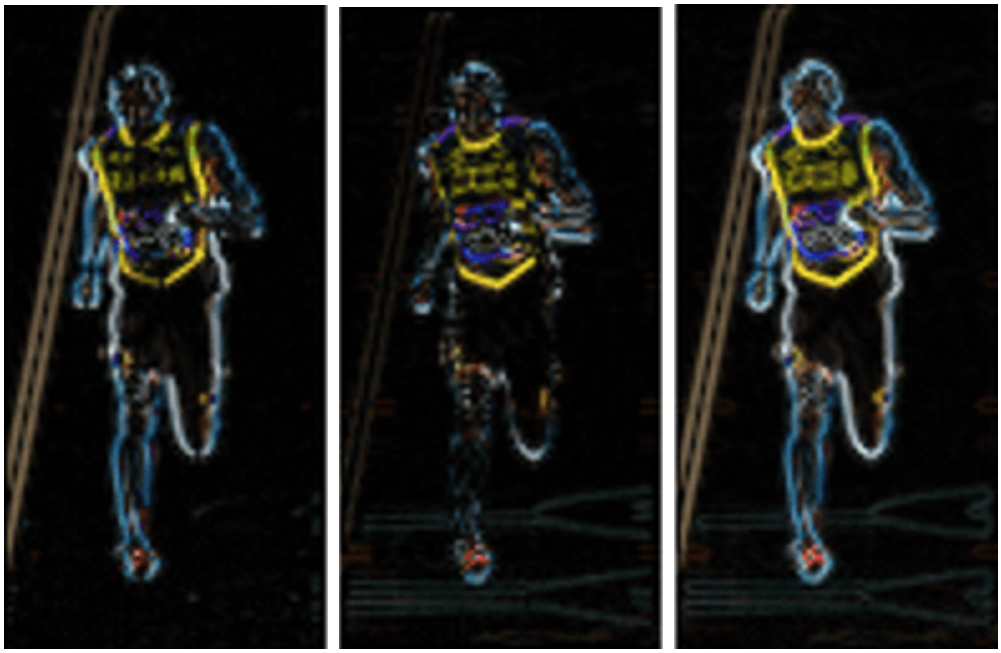
   В итоге каждый пиксель будет иметь значение в одном или двух интервалах, с весом $M(x, y)$.

5. Гистограммы нормализуются в блоках, чтобы уменьшить чувствительность к местам с повышенной контрастностью. Нормализация происхо-

дит по формуле:

$$\mathbf{v}_{\text{norm}} = \frac{\mathbf{v}}{\sqrt{\mathbf{v}_1^2 + \mathbf{v}_2^2 + \cdots + \mathbf{v}_{36}^2 + \epsilon^2}}, \tag{1.16}$$

где $\mathbf{v}$ — это вектор признаков, а ϵ — малая постоянная для численной устойчивости, например $\epsilon = 1 \times 10^{-6}$.



(a) Абсолютное значение градиента по оси x (b) Абсолютное значение градиента по оси y (c) Величина градиента

Рисунок 1.7 Результаты второй стадии

6. Чтобы вычислить характеристический вектор исходного изображения, используется окно размером 16×16, которое скользит с шагом 8 пикселей. На каждом окне строится нормализованный вектор признаков v, и все они объединяются в один длинный вектор [2]:

$$f \in \mathbb{R}. \tag{1.17}$$

1.3.2 Метод опорных векторов

Основная идея метода SVM заключается в классификации объектов путем преобразования исходных векторов, описывающих эти объекты, в пространство более высокой размерности. Две параллельные гиперплоскости строятся

параллельно уже существующей гиперплоскости, разделяющей классы, и по касалельной к ближайшим элементам двух разделяемых классов. В этом новом пространстве метод ищет разделяющую гиперплоскость с максимальным зазором. [4].

Визуализацию метода опорных векторов можно увидеть на рис. 1.8.

Алгоритм опирается на предположение, что увеличение разрыва между этими параллельными гиперплоскостями ведет к снижению средней ошибки классификации [12].

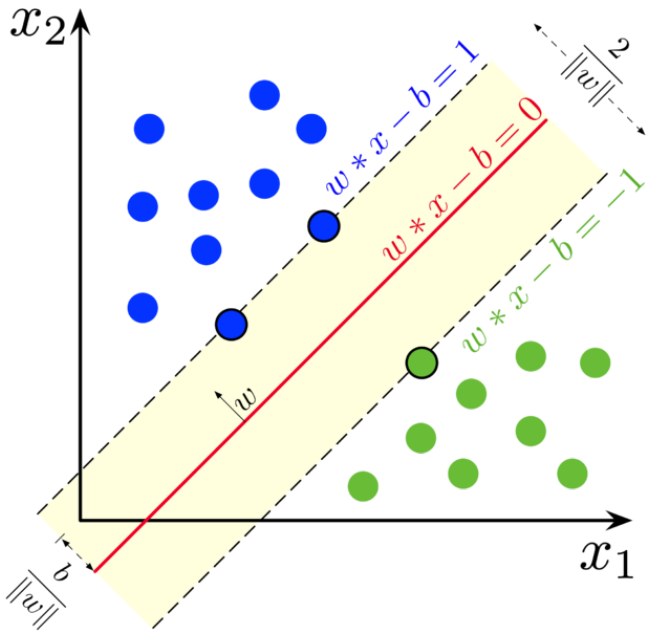


Рисунок 1.8 Визуализация метода опорных векторов

Разделяющей гиперплоскостью называется подпространство коразмерности 1 задаваемое как $\langle \vec{w}, \vec{x} \rangle$, где $\tilde{\mathbf{w}} \in \mathbb{R}^k$ — нормальный вектор к гиперплоскости, а $\tilde{\mathbf{x}} \in \mathbb{R}^k$ — некоторая точка, которая делит $\mathbb{R}^n$ на два полупространства: $\langle \vec{w}, \vec{x} \rangle - b > 0$ и $\langle \vec{w}, \vec{x} \rangle - b < 0$.

Говорят, что гиперплоскость разделяет два класса C_1 и C_2, если их объекты лежат с разных сторон гиперплоскости, то есть выполнено:

$$\begin{cases} \langle \vec{w}, \vec{x} \rangle - a > 0, & \forall x \in C_1, \\ \langle \vec{w}, \vec{x} \rangle - a < 0, & \forall x \in C_2, \end{cases} \tag{1.18}$$

либо

$$\begin{cases} \langle \vec{w}, \vec{x} \rangle - a < 0, & \forall x \in C_1, \\ \langle \vec{w}, \vec{x} \rangle - a > 0, & \forall x \in C_2, \end{cases} \tag{1.19}$$

для $\forall a \in \mathbb{R}^k$.

При этом данный алгоритм работает как в случае линейной разделимости, так и в случае линейной неразделимости. В сущности, в рамках работы алгоритма решается задача квадратичной оптимизации. Задачи определяются по-разному с учетом изначальных условий:

Для линейной разделимости:

$$\begin{cases} \|\vec{w}\|^2 \to min, \\ c_i(\langle \vec{w}, \vec{x} \rangle - a) \geq 1, \ 1 \geq i \geq n. \end{cases} \tag{1.20}$$

Для линейной неразделимости:

$$\begin{cases} \frac{1}{2}\|\vec{w}\|^2 + C \sum_{i=1}^{n} \epsilon_i \to min_{w,x,\epsilon}, \\ c_j(\langle \vec{w}, \vec{x} \rangle - a) \geq 1 - \epsilon_i, \ 1 \geq i \geq n, \\ \epsilon_i \geq 0, \ 1 \geq i \geq n. \end{cases} \tag{1.21}$$

$$\begin{cases} \frac{1}{2}\|\vec{w}\|^2 + C \sum_{i=1}^{n} \epsilon_i \to min_{w,x,\epsilon}, \\ c_j(\langle \vec{w}, \vec{x} \rangle - a) \geq 1 - \epsilon_i, \ 1 \geq i \geq n, \\ \epsilon_i \geq 0, \ 1 \geq i \geq n. \end{cases} \tag{1.22}$$

где коэффициент C представляет собой параметр настройки метода.

Далее, применяя теорему Куна–Таккера к этим двум задачам, а также дальнейшими преобразованиями, получаем:

Для линейной разделимости:

$$\begin{cases} -L(\lambda) = \sum_{i=1}^{n} \lambda_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j c_i c_j (x_i, x_j) \to min_{\lambda}, \\ \lambda_i \geq 0, \ 1 \leq i \leq n, \\ \sum_{i=1}^{n} \lambda_i c_i = 0. \end{cases} \tag{1.23}$$

Для линейной неразделимости:

$$\begin{cases} -L(\lambda) = -\sum_{i=1}^{n} \lambda_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j c_i c_j (x_i, x_j) \to min_{\lambda}, \\ 0 \leq \lambda_i \leq C, \ 1 \leq i \leq n, \\ \sum_{i=1}^{n} \lambda_i c_i = 0. \end{cases} \tag{1.24}$$

На практике при работе SVM решается задача для линейной неразделимости, а не 1.20. Связано это с тем, что невозможно гарантировать линейную разделимость точек на два класса для общих задач.

В зависимости от возможности линейно разделить точки применяются разные алгоритмы:

1. Алгоритмом с мягким зазором для случая линейной разделимости точек.

2. Алгоритмом с жестким зазором для случая линейной неразделимости точек.

Ознакомиться с этими алгоритмами можно в [4].

В итоге алгоритм классификации объектов может быть записан в виде:

$$a(x) = sin\big(\sum_{i=1}^{n} \lambda_i c_i x_i x - a\big). \tag{1.25}$$

1.4 Методы, основанные на оптическом потоке

Один из относительно новых и перспективных методов анализа движения объектов на видеопоследовательности основан на применении оптического потока. С его помощью можно определить распределение скоростей и смещений точек объекта при его перемещении между двумя последовательными кадрами. Оптический поток активно применяется для анализа различных типов движения объектов, котовые были описаны в разделе 1.2, относительно неподвижного или подвижного фона, но в случае динамических объектов, которые в процессе движенния не только меняют положение пикселей в пространстве, но и меняют свою форму и очертание, может приводить к проблемам при определении их движения [9].

Динамическая сцена состоит из различных движущихся объектов, которые могут быть разделены по свойствам на несколько категорий:

1. объекты фона с беспорядочным движением,

2. объекты с вращательным или поступательным движением,

3. объекты с изменением внешней формы,

4. объекты с изменением формы и внутренней структуры;

Отсюда следует, что в динамической системе объекты классифицируются по скорости, величине перемещения и глубине их расположения на сцене. Для более точного описания движения объекты были разделены на слои в зависимости от их скорости смещения. Такой подход позволил проводить идентификацию объектов путем создания масштабной пирамиды и анализа движения с использованием оптического потока.

Оптический поток дает возможность для оценки изменения положения (сдвига) каждой точки тела на двух последовательных кадрах видеопоследовательности. Делается это с серыми версиями кадров и основывается на изменении интенсивности в окрестностях каждого пикселя в диапазоне времени от t до $t + \Delta t$. Получаем, что смещение пикселя (dx, dy, dt) позволяет записать уравнение для интенсивности [9]:

$$I(x, y, t) = I(x + dx, y + dy, t + dt). \tag{1.26}$$

Если значение смещения двух последовательных кадров мало, то эту формула переписывается в виде [9]:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt. \tag{1.27}$$

Из формул 1.26, 1.27 и значения смещения пикселя (dx, dy, dt) можем получить уравнение оптического потока:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0, \tag{1.28}$$

где V_x и V_y — горизонтальное и вертикальное значение компоненты скорости, а $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial I}{\partial t}$ — частные производные по интенсивности [9].

Сложности при обработке таких изображений возникают, когда фон содержит подвижные объекты, что может привести к ошибочному включению этих объектов в классификацию.

Для улучшения работы алгоритма было введено определение интегрального оптического потока, вычисляемого для каждого пикселя, как интеграл

по значениям оптического потока по всем кадрам видеопоследовательности [9]:

$$I(x, y) = \sum_{t} V(x, y), \partial t = 0, \tag{1.29}$$

где I — интегральный оптический поток, а V — оптический поток для двух последовательных кадров.

Использование интегрального оптического потока позволяет исключить из исследования принадлежности объекты, принадлежащие фону, так как они преимущественно не совершают движения, и значение их оптического потока очень мало. Таким образом, значения оптического потока, составляющие векторное поле, накапливаются для каждого последующего слоя. Это позволяет удалить фоновое движение и усилить основное движение. Операция накопления интегрального оптического потока может включать в себя операцию сравнения векторов между слоями, что помогает снизить ошибки классификации динамических объектов. Проще говоря интегральный оптический поток позволяет убрать из иследования объекты которые постояннноприсутствуют на видеопоследовательности, как например, игнорировать движение кроны деревьев от ветра [9].

Ниже, на рис. 1.9, будет приведен алгоритм из статьи [9], использующий интегральный оптический поток для анализа движения динамических объектов, предоставляющий возможность изучать не только движение объекта в целом, но и перемещение и взаимодействие его отдельных элементов.

Ниже приведено краткое описание стадий алгоритма из [9]:

1. На первом этапе происходит сбор последовательности кадров из видеопотока для дальнейшей обработки. Из накопленных кадров создаётся последовательность изображений, формирующая исходное видео, которое будет анализироваться.

2. На втором этапе происходит предварительная обработка объектов, а именно сегментация для выделения объектов, их трекинг, применение фильтрации после чего кадр преобразуется в бинарную матрицу.

3. На третьем этапе происходит вычисление определение характеристик (скорость, траектория, форма и т. д.) и их запись в таблицу данных.

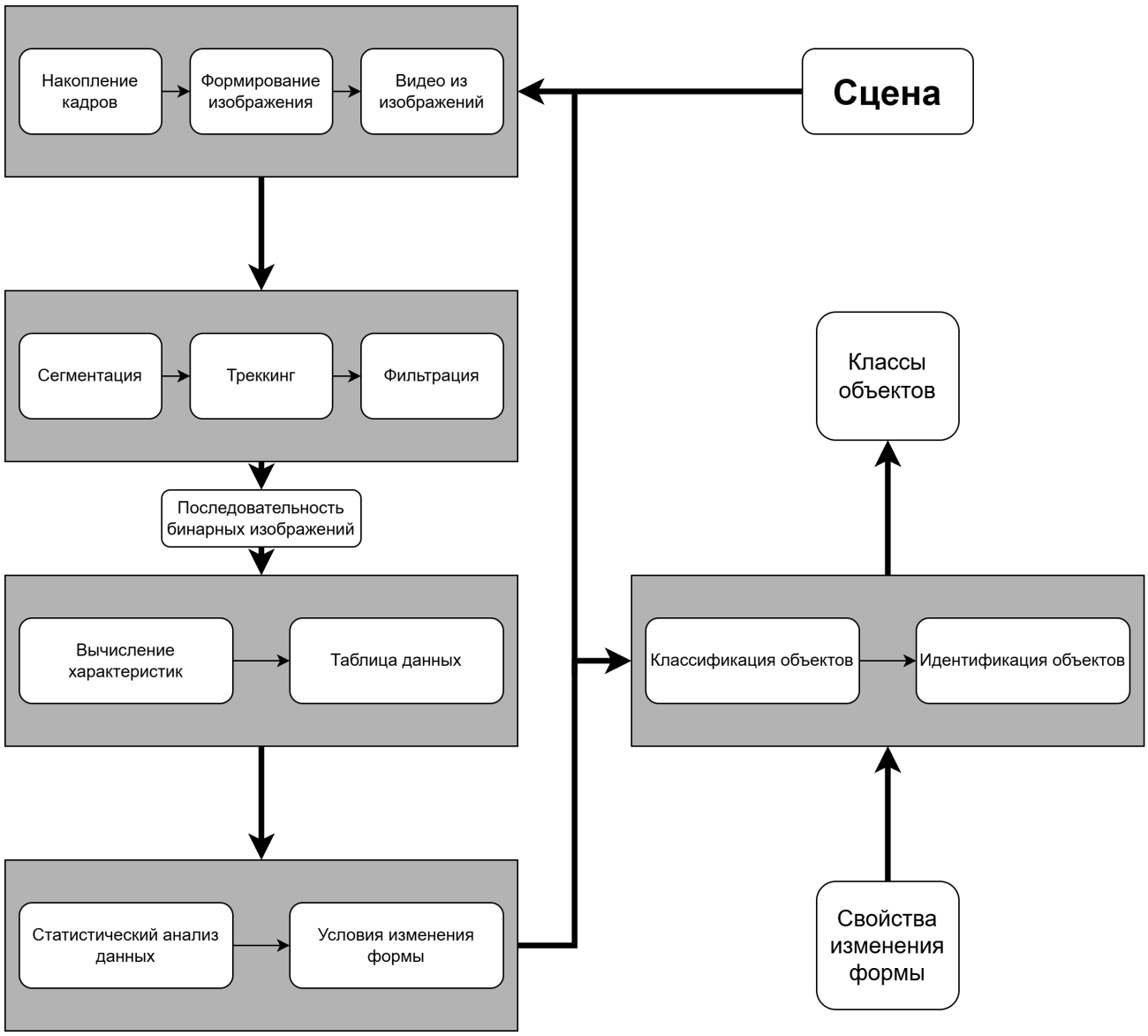


Рисунок 1.9 Общая схема обработки динамических объектов на видео

4. На четвертом этапе происходит статистический анализ данных в который входит изучение характеристик для выявления закономерностей и условий изменения формы объектов.

5. На пятом этапе происходит классификация и идентификация объектов. Тоесть сначало определятся к какому классу относится иследуемый объект, затем происходит определение их уникальности.

1.5 Методы, основанные на нейронных сетях

Использование нейронных сетей, пожалуй, является самым популярным подходом в исследовании изображений и видео человека.

Архитектуру стандартной нейронной сеть можно увидеть на рис. 1.10.

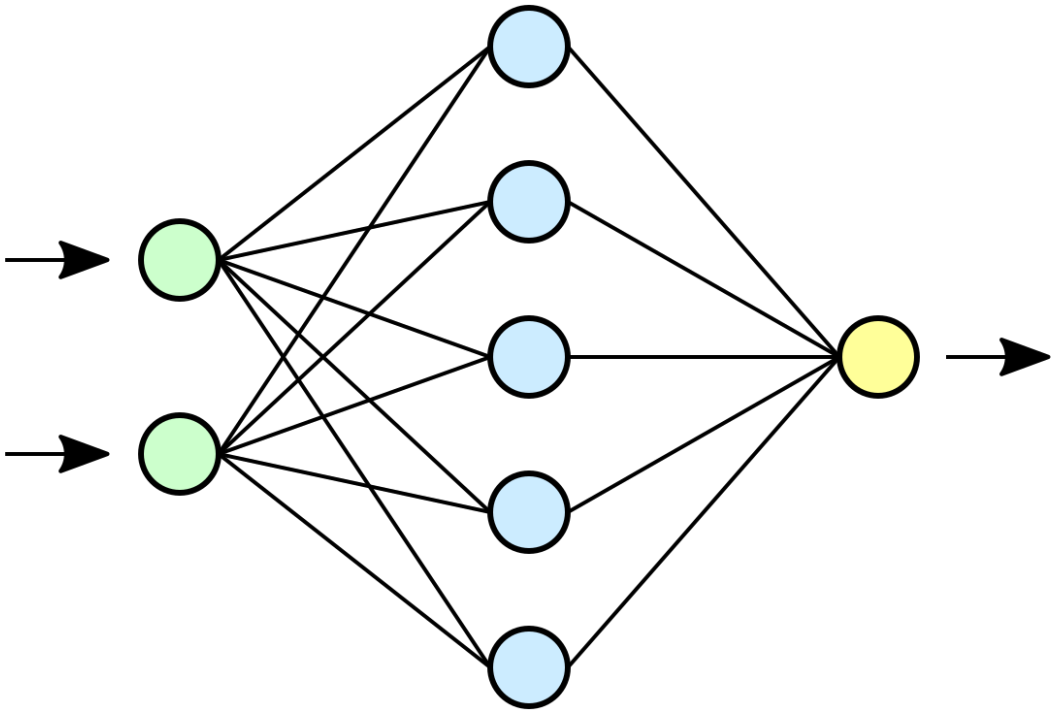


Рисунок 1.10 Архитектура нейронной сети

Для данной задачи можно использавать 2 класса нейронных сетей: сверточные и сети семейства FlowNet.

1.5.1 Сверточные нейронные сети

Сверточные нейронные сети представляют собой отдельную ветвь нейронных сетей, специализирующихся на обработку изображений. Данные нейрости способны эффективно улавливать локальный контекст с изображения, где как известно данные располагаются близко друг к другу. Например, пиксели на изображении находятся рядом и имеют определенные визуальные характеристики [7].

Наиболее частые направления, где применяются сверточные нейронные сети, включают:

1. медицину,
2. приборостроение,
3. системы распознавания лиц,
4. распознавание документов.

В рамках данных сфер происходит распознавания и классификации образов на изображении [8].

Например, в случае медицины сверточные нейронные сети используются для обнаружения патологий и постановки точного диагноза. Уже сейчас при поиска зон пораженных раковыми клетками применяют обученные на это модели, что дает возможность определять зоны и степень поражения организма в 2 раза точнее при определенных случаях.

Чтобы конволюционная нейронная сеть смогла распознать деревья на изображении, ей необходимо выполнить ряд типичных действий для каждого слоя изображения.

Ниже, на рис. 1.11, можно увидеть, как работает конволюционная нейронная сеть:

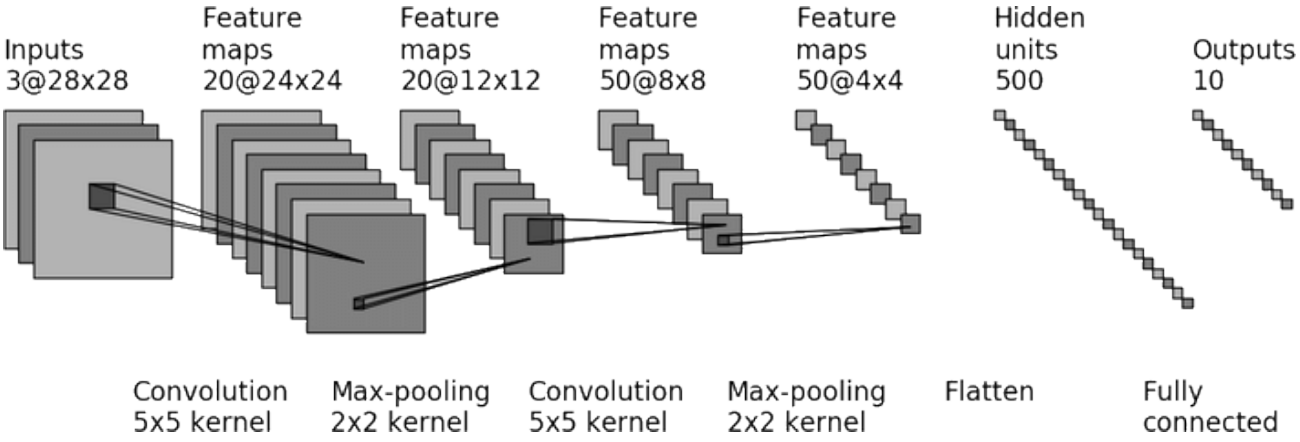


Рисунок 1.11 Архитектура сверточной нейронной сети

Структуру сверточной нейронной сети, очень грубо но можно назвать воронкообразной. Это выражается в том, что сначала происходит получение

информации общего плана после чего уже происходит фокусирование на деталях. Уже здесь можно заметить сходство с тем как работает человеческое зрение. Например, пусть нам необходимо встретится со знакомым в торговом центре. По началу мы рассматриваем общий план, включающий в себя: людей, стены, вывески. После того как приходит понимание, что это место встречи, мы начинаем просматривать лица всех людей с целью найти нашего знакомого. Данный подход называют обучением представления. Реализация данной архитектуры выполняется посредством нескольких слоев, причем большее число слоев, дает более мощную сверточную сеть [7].

Слои применяемые при разработке нейронной сети будут описаны в параграфе 1.5.2.

1.5.2 Операции применяемые в сверточных нейронных сетях

При проектировании конволюционных нейронных сетей существует набор базовых операций и слоев применяемых для создания модели. Ниже приведен их перечень:

1. Конволюция;
2. Сверточный слой;
3. Пулинговый слой;
4. Начальный модуль (Inception module);
5. Остаточный блок (Residual block).

Конволюция представляет собой операцию между двумя матрицами, которая имеет следующий вид:

$$conv(A, B) = C, \tag{1.30}$$

где матрица A имеет размеры nxm, матрица B имеет размеры $k \times k$, а результирующая матрица C имеет размеры $(m - k) \times (n - k)$. Где каждый элемент

матрицы C представляет собой скалярное произведение матрицы B на некоторую матрицу $A_{i,j}$, представляющую собой окно в матрице A размерами как матрица B. А значение $C_{i,j}$ вычисляется следующим образом:

$$C_{i,j} = \sum_{u=0}^{k} \sum_{v=0}^{k} A_{i+u,j+v} B_{u,v}. \tag{1.31}$$

Ниже, на рис. 1.12, представлен пример работы свертки с ядром 3 на 3:

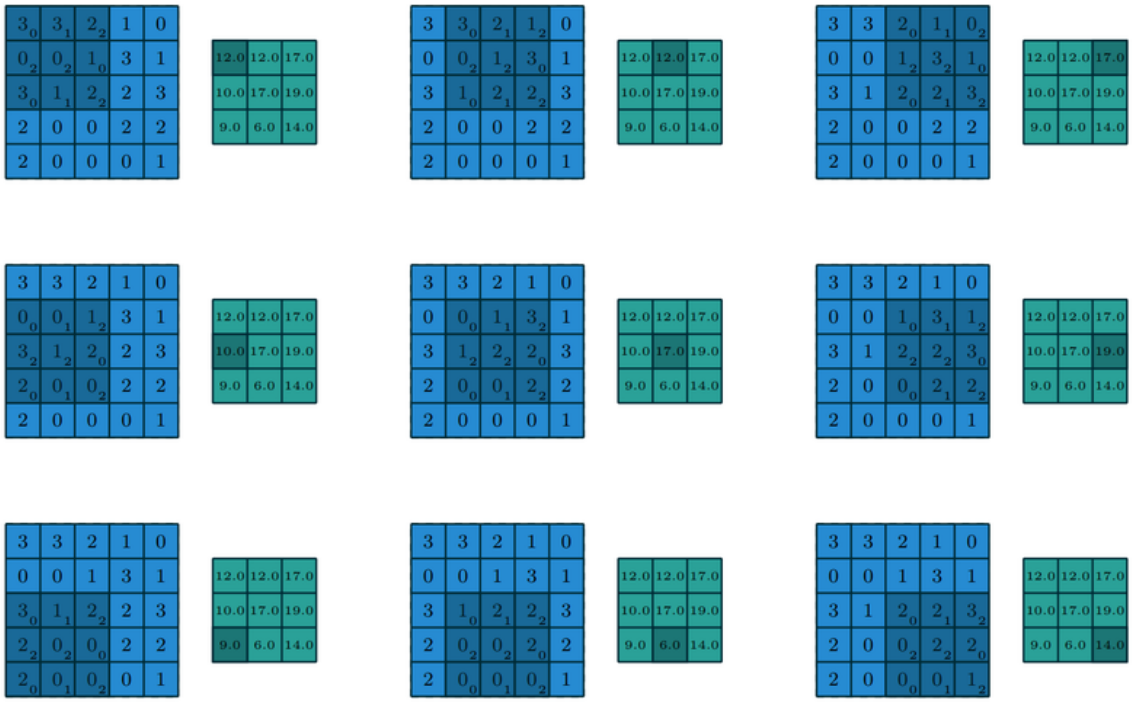


Рисунок 1.12 Пример применения конволюции

Сверточный слой нейронной сети осуществляет операцию свертки над выходными данными предыдущего слоя, при этом веса ядра свертки выступают в роли параметров обучения. При этом существует еще один постоянный вес, называемый сдвигом.

При этом количество операций свертки, применяемых на одном слое, не является ограниченным. В этом случае результат одной свертки будет являться входными данными для следующей операции и так далее.

Пулинговый слой применяется для уменьшения размера изображения, делает он это путем деления изображения на окна размером nxn. Внутри каждого из блоков используется некоторая функция для получения некоторого

значения, которое заменяет собой весь блок. Благодаря этому число пикселей уменьшается на n^2.

Наиболее часто применяется функция максимума, и тогда этот слой носит название max-pooling. При этом операцию max-pooling можно задать следующим образом:

$$C_{i,j} = maxA_{u,v}, \quad u \in [i*n; i*n+n),\ v \in [j*n; j*n+n), \tag{1.32}$$

где матрица A имеет размеры $l \times m$, а результирующая матрица C имеет размеры $\frac{l}{n} \times \frac{m}{n}$.

Ниже, на рис. 1.13, представлен пример операции пулинга с функцией максимума с ядром 2 на 2.

Начальный модуль представляет собой специальный слой нейронной сети, который был предложен в работе [14]. Суть данного слоя заключается в том, что он позволяет извлекать признаки с разными масштабами и уровнями абстракции одновременно, используя параллельные сверточные слои разного размера.

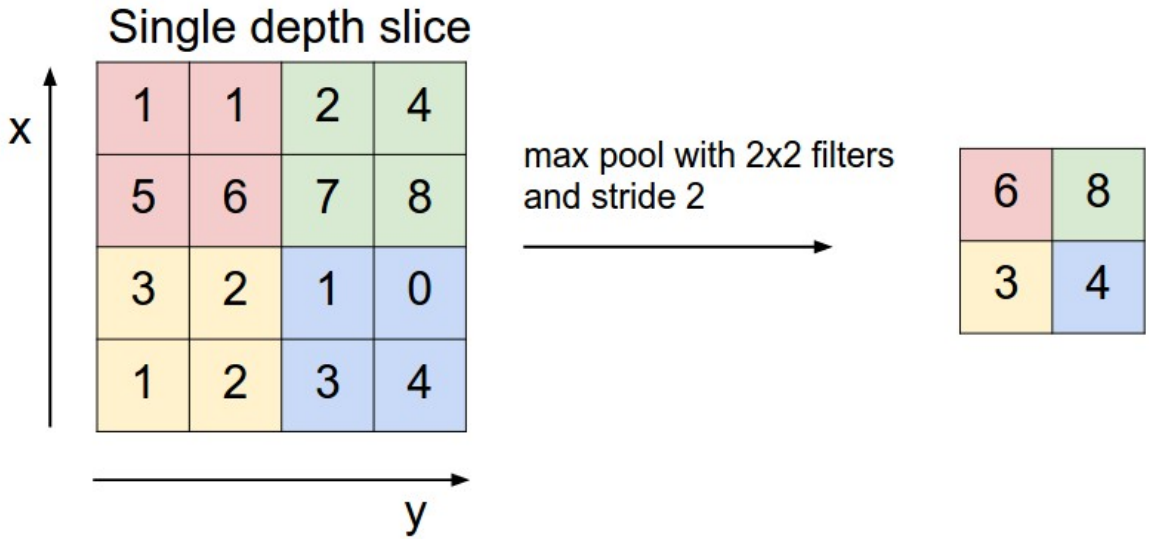
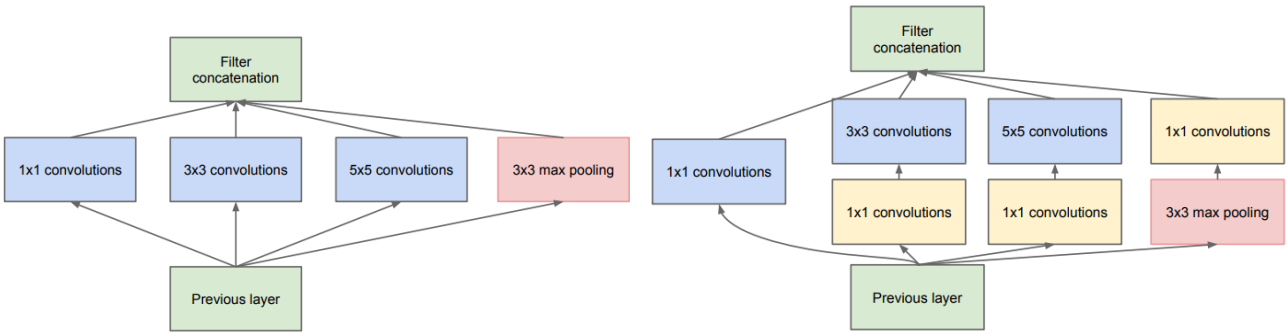


Рисунок 1.13 Пример операции пулинга с функцией максимума

Принцип работы заключается в том, что вместо последовательного применения слоев, начальный модуль обрабатывает один и тот же вход разными сверточными фильтрами параллельно, а затем объединяет их результаты.

Ниже, на рис. 1.14, представлен пример начального модуля из работы [14]:



(a) Наивная версия (b) Версия с уменьшением размеров

Рисунок 1.14 Начальный модуль

Остаточный блок представляет собой слой главным смыслом которого является борьба с двумя серьезными проблемами в обучении глубоких нейронных сетей, а именно исчезающий градиент и взрывающийся градиент.

Принцип работы заключается в том, чтобы взять пару слоёв, и добавить дополнительные пути, которые проходят с пропуском некоторого числа слоев.

Пусть $z^{(k)}$ — выход k-ого слоя до применения функции активации, а $a^{(k)}$ — выход после. Тогда остаточный блок будет выполнять следующее преобразование:

$$a^{(k+2)} = activ(z^{(k+2)} + a^{(k)}), \tag{1.33}$$

где $activ$ — функция активации.

Ниже, на рис. 1.15, представлен пример остаточного блока.

1.5.3 Сверточная нейронная сеть YOLOv5

Данная сверточная нейронная сеть была представленна Joseph Redmon, Santosh Divvala, Ross Girshick и Ali Farhadi в статье [13] и представляет собой целое семейство YOLO(You Only Look Once), все сети которого используют одну архитектуру нейронных конволюционных сетей для класификации и прогнозирования рамок объекта. Они разбивают изображение на ячейки, каждая из которых старается определить рамки и просчитать какие вероятности принадлежности к классу характерны для объекта. Из-за особенности архитектуры YOLO выполняет прогнозы с низкими затратами по времени,

что делает ее привлекательным решением для систем с требованием обнаружения в реальном времени [3].

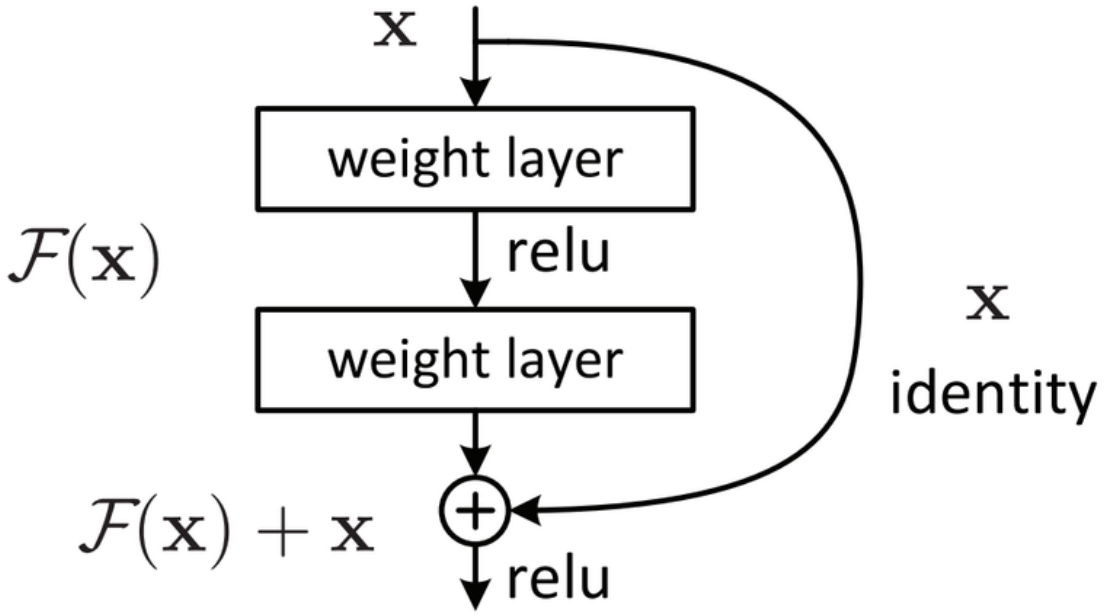


Рисунок 1.15 Пример остаточного блока

YOLOv5, как не трудно догадатся одна из сетей семейства YOLO, зарекомендовала себя современным и популярным решением для обнаружения объектов. Она использует одноступенчатую архитектуру детектора, в которой модель сразу предсказывает координаты нескольких ограничивающих рамок, их классификацию и вероятность присутствия объекта, а затем корректирует их позицию.

Архитектура YOLOv5 делится на три основные компонента [3]:

1. Backbone: Основной блок сети, построенный на базе модифицированной структуры New CSP-Darknet53, которая является доработкой архитектуры Darknet, применяемой в предыдущих версиях.

2. Neck: Этот модуль служит связующим звеном между магистралью и головой модели. В YOLOv5 используются структуры SPPF и New CSP-PAN.

3. Head: Отвечает за формирование итоговых результатов обнаружения. Для этой цели в YOLOv5 применяется YOLOv3 Head.

Пример работы YOLOv5 можно увидеть на рис. 1.16.

Рисунок 1.16 Пример работы YOLOv5

Подробное представление структуры модели можно увидеть ниже, на рис. 1.17.

1.5.4 Сверточная нейронная сеть VGG16

VGG16 представляет собой архитектуру сверточной нейронной сети, предложенную K. Simonyan и A. Zisserman из Оксфордского университета в исследовании 2015 года. Архитектура VGG16 представлена на рисунке 1.18. Изображения проходят через ряд сверточных слоев, где используются фильтры с малым рецептивным полем размером 3×3. Этот размер является минимальным для формирования представления о пространственном расположении объектов, таких как право/лево, верх/низ и центр. Существует модифицированная конфигурация где используется свертка размера 1×1, который может быть представлен как линейная трансформация входных каналов со сверточным шагом при значение 1 пиксель [5].

Пространственное дополнение на входе сверточного слоя получает такие значения, чтобы размерность изображения оставалось не изменным, то есть дополнение равно 1 для 3×3 сверточных слоев [5].

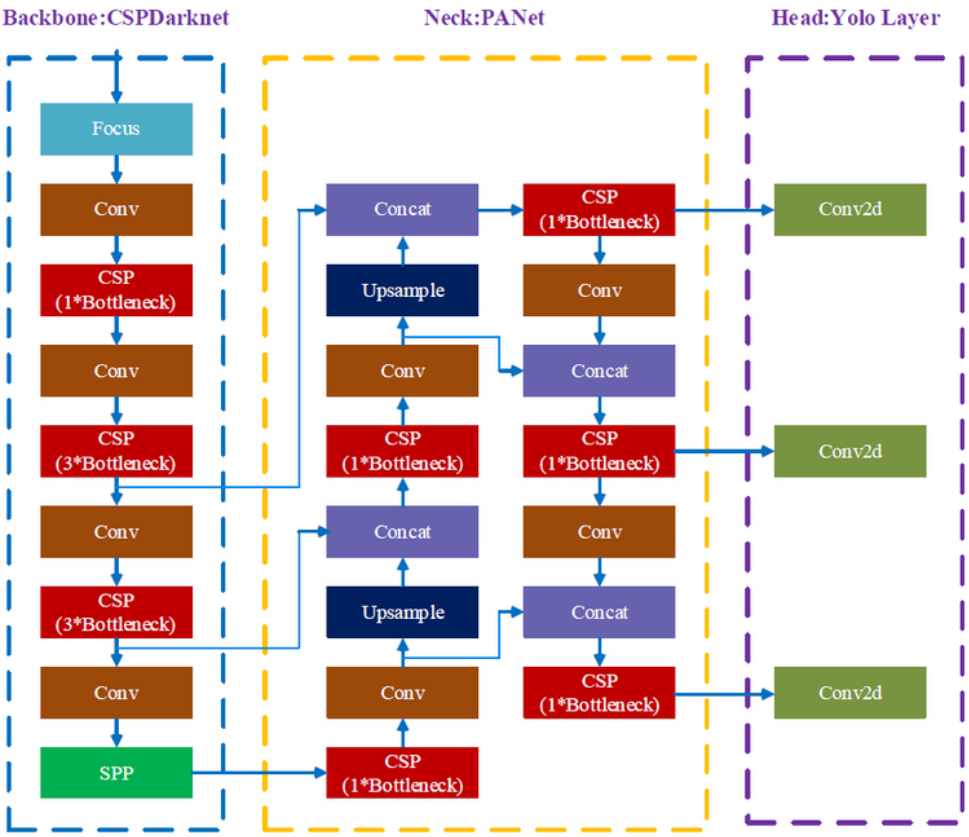


Рисунок 1.17 Архитектура нейронной сети YOLOv5

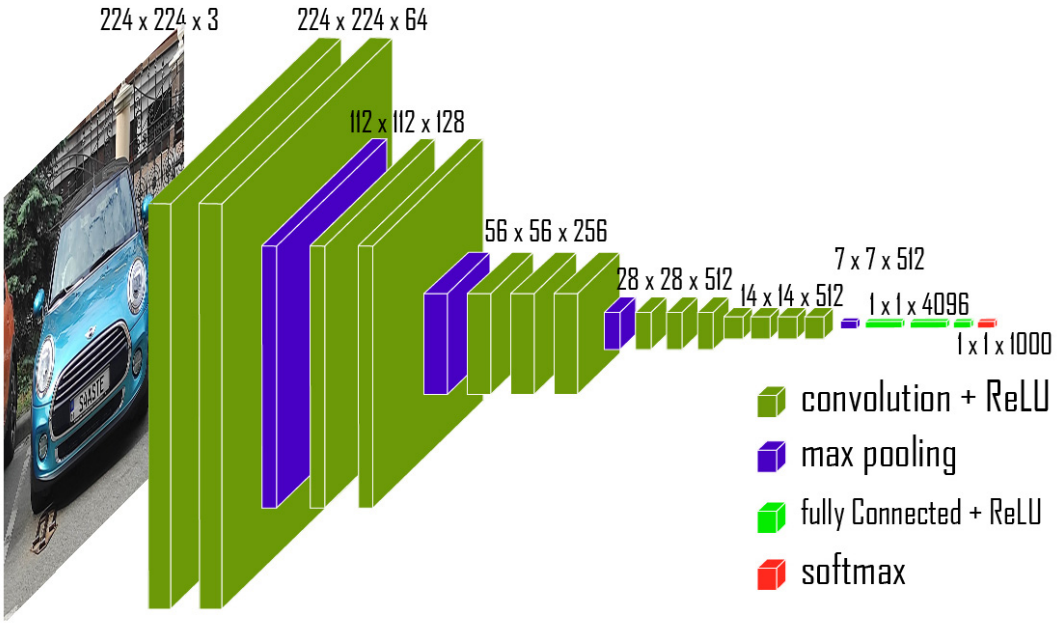


Рисунок 1.18 Архитектура нейронной сети VGG16

ГЛАВА 2

Постановка задачи определения движения человека на заданной видеопоследовательности

В рамках работы будет рассмотрена и решена задача, связанная с определением действия человека на видеопоследовательности. Будет разработано три решения:

- Одно будет основано на оптическом потоке,
- Второе решение будет основано на комбинации конволюционной нейронной сети YOLO5 и VGG16.
- Третье решение будет основано на комбинации пары гистограммы направленных градиентов и метода опорных векторов и конволюционной нейронной сети VGG16.

Цель работы: разработать решения, способные анализировать видеопоток для извлечения наибольшего количества информации о действиях, поведении и общем контексте, с целью последующего определения, насколько поведение человека соответствует норме. Если быть точнее, то определение аномальности поведения ложится на плечи человека, а не алгоритма. Само решение должно выдавать наибольшее количество анализируемых данных, на основании которых пользователь будет делать вывод о нормальности поведения объекта исследований.

2.1 Описание подхода к решению

Для обнаружения аномалий в видеопоследовательности нужно разработать критерий, который будет основывать свое решение на сравнении текущих характеристик движения с эталонными (нормальными) значениями.

В данном параграфе подробно будут описаны архитектуры решения, предлагаемые в работе, где у каждого будет свой критерий аномальности.

2.1.1 Решение основанное на оптическом потоке

Основной идеей данного решения является то, что аномальные события (например, экстренные ситуации, необычные действия) вызваны резким изменением скорости движения объекта.

Для этого будет использован метод Farneback библиотеки opencv, который используется для расчёта плотного оптического потока, позволяющего вычислить вектор смещения каждого пикселя. Анализ векторов смещения позволяет понять, какое движение происходило между двумя кадрами.

Для упрощения расчетов видеопотока изображения преобразуются в серые оттенки. Это позволяет ускорить обработку и фокусироваться на информации о движении, не учитывая цветовые компоненты. Для снижения влияния шума и мелких деталей будет применено размытие Гаусса, что позволит оставить только движение больших объектов, таких как:

- Люди,
- Машины,
- Животные,
- И другие.

Теперь краткое описание алгоритма по блокам:

1. Алгоритм начинает с чтения видеофайла. Видео обрабатывается с помощью библиотеки OpenCV.
2. Далее производится несколько операций:
    - Сжатие кадров для уменьшения вычислительной нагрузки.
    - Преобразование в оттенки серого.
    - Фильтрация для удаления шума и сглаживания изображения.
3. Настройка порога для аномального поведения:
    - Порог, при котором движение будет считаться значительным.

- Коэффициент, с которым сравнивается текущая скорость потока с максимальной для предыдущего кадра, чтобы классифицировать поведение как аномальное.

4. Вычисление оптического потока.

5. Происходит обнаружение аномального поведения:

   - Если максимальная скорость потока в текущем кадре значительно превышает скорость потока предыдущего кадра с учетом порога, алгоритм считает поведение аномальным.

   - Когда поведение становится аномальным, в консоль выводится сообщение которое говорит о ненормальности движения.

6. Визуализация результата.

Ниже на рис. 2.1 представлена блок-схема решения.

2.1.2 Данные, примененные для обучения нейронных сетей

Для обучения нейронных сетей использовался датасет Human Action Recognition:

www.kaggle.com/datasets/meetnagadia/human-action-recognition-har-dataset

В данном датасете находится более 12 тысяч фотографий, где каждой в соответствие ставится один из 15 семантических классов. Также каждое изображение соответствует лишь одному действию.

При этом датасет заранее размечен на тренировочные и тестовые данные.

В таблице 2.1 приведены классы, которые использовались при обучении данной модели.

Изначально данный датасет создавался с целью решить задачу распознования поведения человека, применив различные подходы к существующей проблеммы.

Рисунок 2.1 Блок-схема первого решения

Ниже на рис. 2.2 приведены примеры данных из датасета.

Рисунок 2.2 Примеры данных из датасета

Таблица 2.1 Семантические классы данного набора данных

Номер	Название	Описание
1	calling	Проведение телефонного разговора
2	clapping	Хлопанье в ладоши
3	cycling	Велосипедная поездка
4	dancing	Танцевальное движение
5	drinking	Питье напитка
6	eating	Прием пищи
7	fighting	Схватка, драка
8	hugging	Объятие, обнимание
9	laughing	Смех, хихиканье
10	listening_to_music	Прослушивание музыки
11	running	Бег
12	sitting	Сидение
13	sleeping	Сон
14	texting	Переписка по сообщениям
15	using_laptop	Использование ноутбука

2.1.3 Решение основанное на комбинации нейронных сетей YOLOv5 и VGG16

Основной идеей данного решения является нахождение человека на видеопоследовательности и определение, к какому классу относится выполняемое им действие.

Решение делится на две части. На первом этапе происходит извлечение отдельных кадров. Далее применяется обученная модель YOLOv5, специально настроенная для детекции одного ключевого класса - человека. Делается это для того, чтобы избежать переброса фокуса с человека на другой объект, понятно, что алгоритм вернется через пару кадров назад к человеку, но упущенный момент может оказаться важным при анализе. На втором этапе из полученных изображений мы определяем, какое действие выполняет найденный объект.

Ниже на рис. 2.3 указана архитектура данной сети:

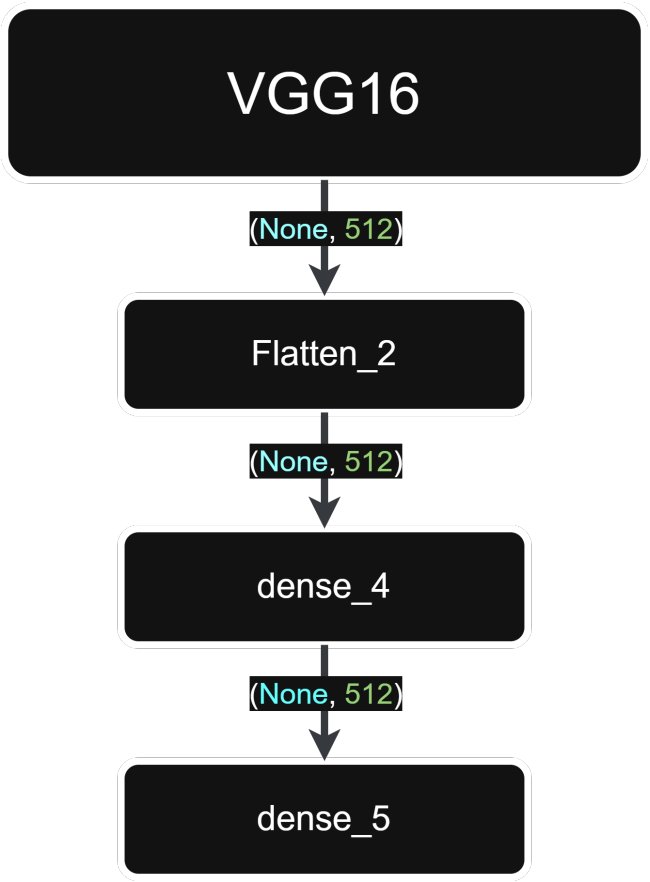


Рисунок 2.3 Архитектура применяемой конволюционной нейронной сети VGG16

Ниже привел описание работы алгоритма:

1. Алгоритм начинает с чтения видеофайла. Видео обрабатывается с помощью библиотеки OpenCV.

2. Далее производится сжатие кадров для дальнейших шагов.

3. Происходит применение YOLOv5 для поиска первого объекта у которого вероятность оказаться человеком более 50%, после чего определяется в границах каких пикселей находится данный объект.

4. Далее происходит предобработка полученных данных:

    - Определяется где находится центр полученной границы, делается это с использований следующих математических формул:

$$x_c = \frac{(x_1 + x_2)}{2}, \tag{2.1}$$

$$y_c = max(x_2 - x_1, y_2 - y_1). \tag{2.2}$$

- Определяется сторона квадрата, с центром в (x_c, y_c):

$$side = \frac{(x_1 + x_2)}{2}. \tag{2.3}$$

5. После того как было вычислено все необходимое, происходит извлечение полученного квадрата и применение к нему маштабирования, чтобы привести его к формату трехмерной матрицы (3, 160, 160).

6. К матрице полученной в пункте 5 применяется обученую модель VGG16 и определяем вероятности, которые определяют к какому классу относится действие выполняемое человеком.

7. Визуализация результата.

Ниже на рис. 2.4 представлена блок-схема решения.
В конце происходит сохранение итогового видео.

2.1.4 Решение основанное на комбинации HOG+SVM и нейронной сети VGG16

Основной идеей данного решения как и в случае предыдущего является нахождение человека на видеопоследовательности и определение, к какому классу относится выполняемое им действие. Однако вместо применения нейронной сети для детекции человека, будет использована пара методов HOG и SVM. Данное изменение должно ускорить выполнение и дать пользователю больше управления над ходом выполнения.

Решение делится на три части. На первом этапе происходит извлечение отдельных кадров. Далее применяется HOG-детектор к кадру для поиска людей. На втором этапе происходит извлечение наибольшей из областей, полученных на предыдущем этапе, и масштабирование для передачи на следующий этап. На третьем этапе из полученных изображений определяется, какое действие выполняет найденный объект. Делается это с использованием конволюционной нейронной сети VGG16.

Рисунок 2.4 Блок-схема второго решения

Нейронная сеть VGG16, применяемая в данном решении, та же самая, что и в предыдущем.

Ниже привел описание работы алгоритма:

1. Алгоритм начинает с чтения видеофайла. Видео обрабатывается с помощью библиотеки OpenCV.

2. Кадр масштабируется до фиксированного размера в 400x300.

3. Происходит применение детектора HOG к кадру для поиска людей. Затем происходит извлечение объекта с наибольшей площадью.

4. После того как было вычислено все необходимое, происходит извлечение полученного квадрата и применение к нему маштабирования, чтобы привести его к формату трехмерной матрицы (3, 160, 160).

5. К матрице полученной в пункте 4 применяется обученую модель VGG16 и определяем вероятности, которые определяют к какому классу относится действие выполняемое человеком.

6. Визуализация результата.

Ниже на рис. 2.5 представлена блок-схема решения.

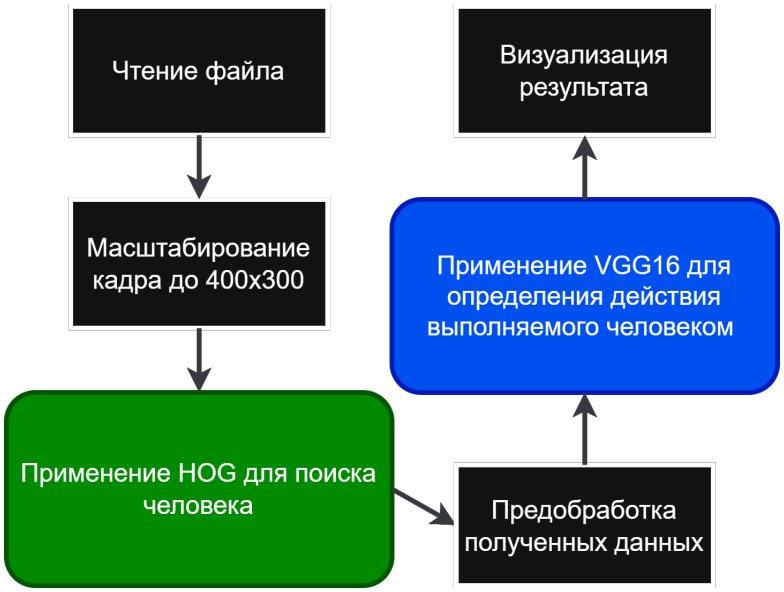


Рисунок 2.5 Блок-схема третьего решения

В конце происходит сохранение итогового видео.

ГЛАВА 3

Полученные результаты

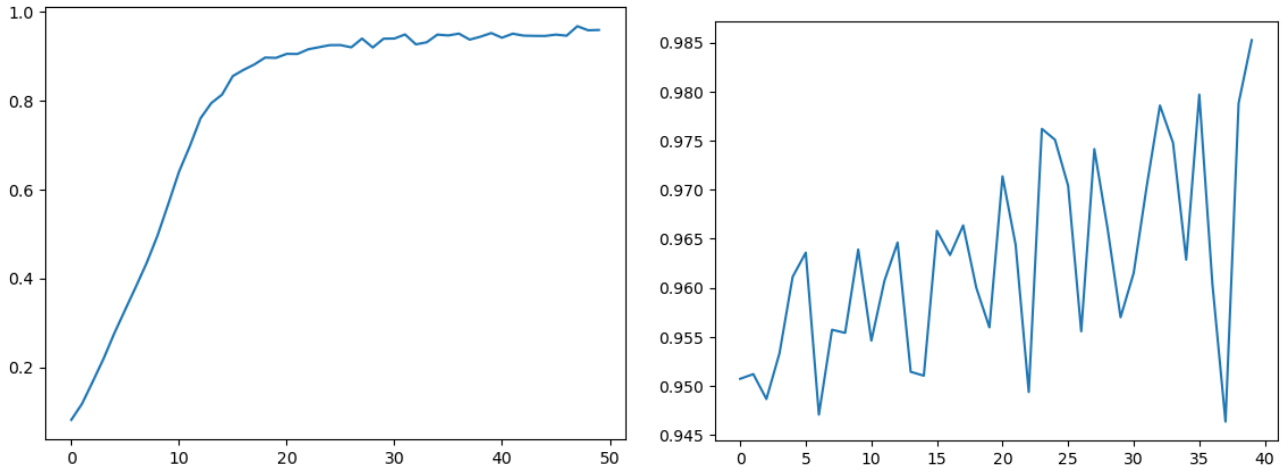
3.1 Результаты обучения нейронной сети VGG16

В данном разделе будет представлен результат обучения модифицированной сверточной нейронной сети VGG16 для определения поведения человека, описанной в параграфе 2.1.3.

Количество эпох обучения модели: 50+40.

После обучения получаем нейронную сеть которая на вход принимает трехканальные изображения 160 на 160 пикселей и определяет вероятность принадлежности кадра к одному из 15 классов поведения человека.

Ниже на рис. 3.1 представлены графики точности в процессе обучения:



(a) В момент обучения на 50 эпохах

(b) В момент дообучения на 40 эпохах

Рисунок 3.1 Значение точности

В результе обучения данной сверточной нейронной сети была получена модель, которую можно в дальнейшем применять в алгоритмах, представленных в данной работе, для классификации действий выполняемых человеком.

Ниже на рис 3.2 представлены результаты тестирования обученной нейронной сети:

(a) В момент обучения на 50 эпохах

(b) В момент дообучения на 40 эпохах

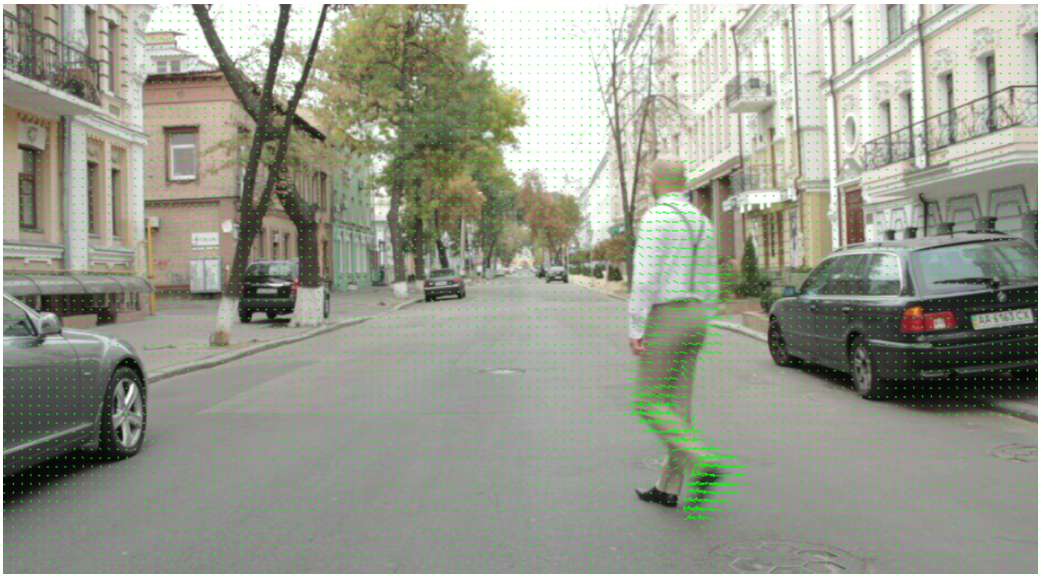
Рисунок 3.2 Значение точности

3.2 Результаты решения, основанного на оптическом потоке

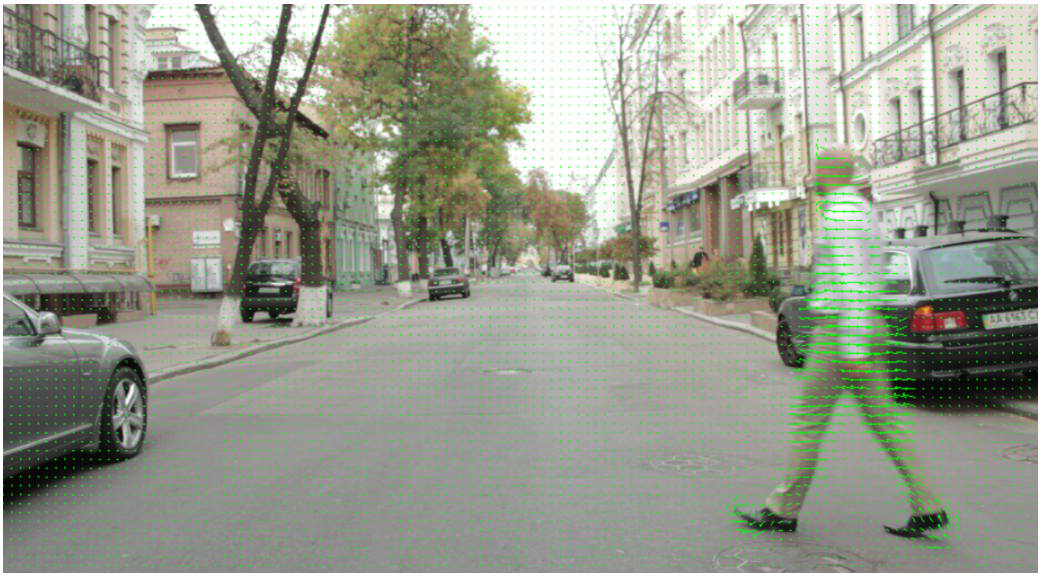
В результате было получено решение, основанное на оптическом потоке по алгоритму, приведенному в параграфе 2.1.1. Решение было разработано на языке *Python* с использованием библиотеки *OpenCV*.

Код данного алгоритма можно найти в приложении В.

На рис 3.3 приведены примеры использования данного алгоритма.



(a) Кадр 1



(b) Кадр 2

Рисунок 3.3 Пример применения алгоритма

При использовании алгоритма, основанного на оптическом потоке, получены результаты, которые позволяют определить моменты, когда объект резко изменяет свою скорость. Однако разработанный алгоритм в нынешнем виде не идеален, поскольку, например, проезжающая рядом машина может значительно исказить полученные значения. К тому же, в условиях плохой освещенности или наличия других движущихся объектов в кадре, данное решение может столкнуться с трудностями в точном определении изменений скорости человека. Это может привести к неточным или искаженным данным, которые

затруднят последующий анализ и интерпретацию результатов.

Данное решение не определяет, какой объект выполняет движение. То есть оно является наполовину автономным, так как после получения результата человек должен просмотреть момент, где алгоритм указал на нашейденную аномалию, и понять, что объект движения относится к классу людей или иных интересующих нас объектов, и действительно выполняет ненормальное действие.

Ниже на рис. 3.4 пример вывода в консоль:

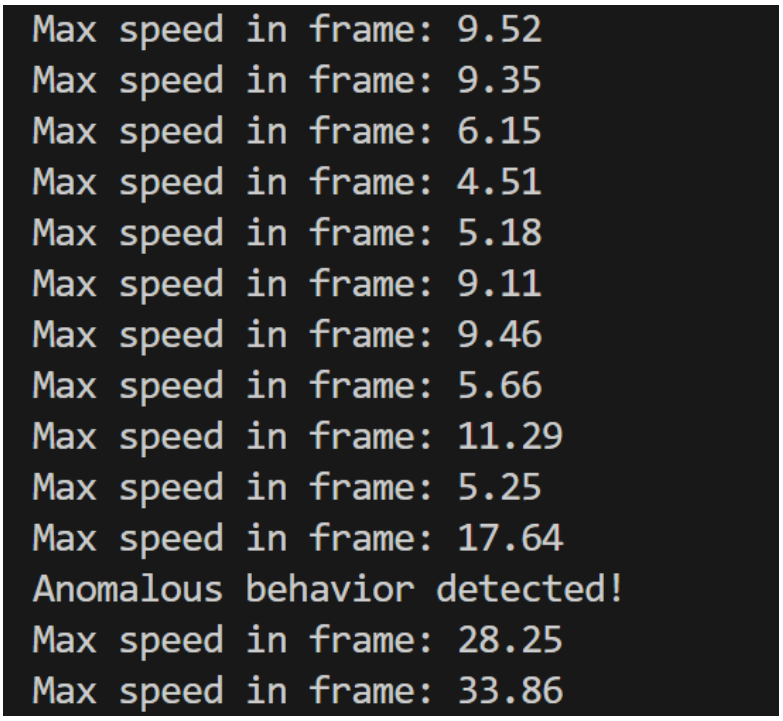


Рисунок 3.4 Блок-схема третьего решения

Конечно, определение принадлежности к некоторому классу (в нашем случае к классу людей) может быть выполнено с использованием некоторых алгоритмов, таких как:

- Конволюционные нейронные сети,
- Метод выделения признаков на основе гистограммы направленных градиентов и метода опорных векторов,
- И другие.

Далее алгоритм можно было бы перестроить ровно так, как это происходит во втором или третьем решении, то есть: настроить на работу в небольшой зоне вокруг полученного объекта, благодаря чему проблема, связанная с нежелательными объектами, была бы решена.

Однако сделать это было бы значительно сложнее, чем в вышеописанных решениях, так как форму, внутри которой располагается объект, нужно было бы как-то центрировать около одной точки, и, скорее всего, просчитывать оптический поток сразу с центром в нескольких точках. Эти ограничения связаны с тем, что два других решения не работают с видеопоследовательностью как единым целым, вместо этого они работают с кадрами так, как будто бы они независимы.

Тем не менее, с учетом возможных ограничений и артефактов, использование алгоритма предоставляет ценную информацию для исследований и практических приложений в области компьютерного зрения и машинного обучения.

3.3 Результаты решения основанное на комбинации нейронных сетей YOLOv5 и VGG16

В результате было получено решение, основанное на комбинации двух сверточных нейронных сетей YOLOv5 и модифицированной VGG16 разработанное по алгоритму, приведенному в параграфе 2.1.3. Решение было реализовано с использованием языка *Python* при помощи библиотек *OpenCV*, $tensorflow$.

Во время своей работы алгоритм находит человека и определяет действие, которое он выполняет, однако, так как датасет для данной нейросети не рассматривать все возможные случаи при которых действия выполняются, или действие происходит при другом освещении, то возникает проблема связанная с неверной классификацией поведения человека. Вместо этого предполагается рассчитывать процент того, сколько данное действие повторялось на последних 60 кадрах видеопоследовательности.

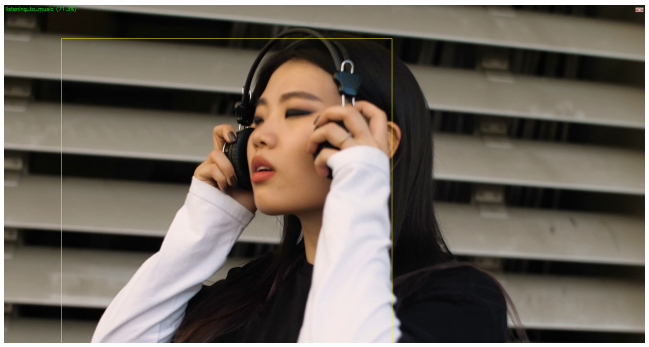
Также решение можно назвать модульным, така как оно позволяет заменить модель для определения поведения на другую.

Следовательно, для того чтобы улучшить результат, можно заранее собирать необходимые данные на месте, на котором планируется установка камеры с данным алгоритмом. Затем необходимо обучить нейронную сеть с использованием нового датасета. В результате будет получена модель менее подверженная шумам и особенностям данного места, что увеличит точность алгоритма.

Данный алгоритм нельзя отнести к классу алгоритмов на видеопоследовательности для работы в реальном времени. Для его работы необходимо заранее определить момент, на котором стоит искать возможное аномальное поведение человека. Так как время обработки одного кадра на видеопотоке занимают, около 200 миллисекунд на моём компьютере.

На рис 3.5 приведены примеры использования данного алгоритма.

(a) Пример применения на видео номер 1



(b) Пример применения на видео номер 2

(c) Пример применения на видео номер 3

Рисунок 3.5 Пример применения алгоритма

Если допускается более низкая точность, то алгоритм можно ускорить. Так как в отличии от решения, основанного на оптическом потоке, данное не работает с видеопоследовательностью, а с каждым кадром индивидуально, то пропуская n число кадров алгоритм будет работать быстрее.

Для того чтобы определить является ли действие выполняемое человеком в месте применения алгоритма аномальным, необходимо сначала определить набор поведений, которые являются нормальными. И затем сравнивать результат работы алгоритма с этим набором. По сути предлагается использовать подход из 1.1.

3.4 Результаты решения основанное на комбинации HOG + SVM и нейронной сети VGG16

В результате было получено решение, основанное на комбинации пары гистограммы направленных градиентов и метода опорных векторов с модифицированной сверточной нейронной сетью VGG16 разработанное по алгоритму, приведенному в параграфе 2.1.4. Решение было реализовано с использованием языка $Python$ при помощи библиотек $OpenCV$, $tensorflow$.

Для этого решения актуально все то что было описано для предыдущего, за исключением определенных бонусов которые дает то, что вместо YOLO используются алгоритмические методы.

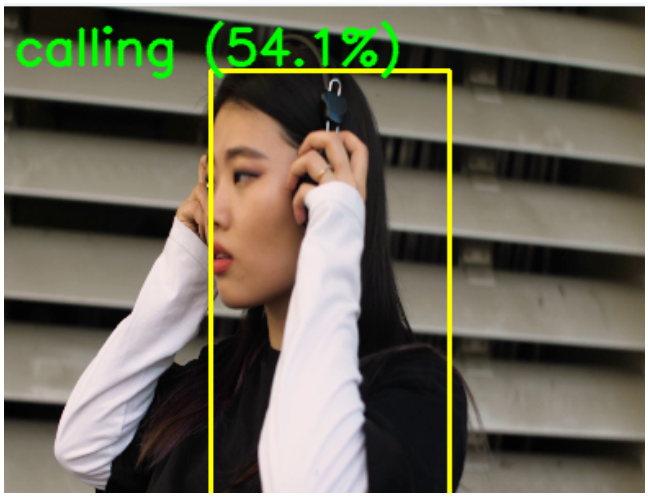
Например данное решение позволяет настраивать размер входного изображения, главное чтобы разрешение было $4:3$. Благодаря чему можно изменять время работы и точность поиска человека, чем ниже размеры изображения, тем выше скорость работы и ниже точность детекции, и наоборот, чем выше размеры изображения, тем выше точность детекции и ниже скорость работы.

В результате тестирования с использованием бинарной классификации у меня в среднем получалось, что данное решение работало быстрее, точнее, и чаще находило человека на кадре.

На рис 3.6 приведены примеры использования данного алгоритма.

(a) Пример применения на видео номер 1

(b) Пример применения на видео номер 2



(c) Пример применения на видео номер 3

Рисунок 3.6 Пример применения алгоритма

ЗАКЛЮЧЕНИЕ

В рамках дипломной работы исследованы следующие технологии в области компьютерного зрения:

1. оптический поток для определения изменения скорости движущегося объекта;

2. гистограммы направленных градиентов и метода опорных векторов для поиска человека на кадре;

3. сверточные нейронные сети для классификации поведения человека;

Данные технологии были опробованны для извлечения ценных характеристик которые могли бы использоваться для определения характера поведения человека на видеопоследовательности.

В рамках работы удалось разработать и обучить модифицированную версию сверточной нейронной сети VGG16 для получения прогноза по 15 классам поведений человека.

В результате работы удалось спроектировать и разработать 3 решения для определения нормальности поведения человека на видеопоследовательности. При этом разработанные алгоритмы были протестированы на разных видеопоследовательностях, что позволило говорить о том, что поставленные в рамках данной работы задачи были выполнены. Каждое решение обладает рядом преимуществ и недостатков, так, например, решение, основанное на оптическом потоке, является быстро настраиваемым и требующим минимального числа примеров данных для проведения корректировки, но не дающее полнценного ответа. Решения, основанные на комбинации YOLOv5, или пары гистограммы направленных градиентов и метода опорных векторов и модифицированной VGG16, является более точным и автономным решением, однако требующих заранее размеченных данных для обучения.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Богуш. Р. П. Вычисление и анализ признаков движущихся объектов для сопровождения на видеопоследовательности / Р. П. Богуш, С. В. Абламейко, И. Ю. Захарова // Вестник Полоцкого государственного университета. – 2021. С. 2-10.

2. Гильманов Р. Ф. Определение объектов на изображениях с помощью гистограммы направленных градиентов и метода опорных векторов / Р. Ф. Гильманов, Куляс О. Л. // Экономика и социум №5(60). – 2019. С. 1422-1424.

3. Зейнетдинов, Б. Г. Взгляд на yolov5 для распознавания объектов / Б. Г. Зейнетдинов // Карагандинский технический университет имени Абылкаса Сагинова. – 2024. – №6. – С. 1-7.

4. Михайлов, И. С. Разработка модификации метода опорных векторов для решения задачи классификации с ограничениями на предметную область / И.С. Михайлов, Зеар Аунг, Йе Тху Аунг // Программные продукты и системы. – 2020. – С. 439-448.

5. Муаль М. Н. Б. Использование предобученной нейросети (VGG16) для решения задачи переноса стиля изображения / М. Н. Б. Муаль // Современные информационные технологии и ИТ-образование. – 2022. С. 243-248.

6. Пятаев. А. С. Классификация методов обнаружения аномального поведения на видеопоследовательностях / А. С. Пятаев // Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. – 2017. С. 365-366.

7. Сайфутдинов, А. В. Сверточные нейронные сети: примеры реализаций: учебно-методическое пособие / А. В. Сайфутдинов // Suol Innovations Ltd. - 2023. – С. 42-44.

8. Соробин, А. Б. Сверточные нейронные сети для решения задач компьютерного зрения / А. Б. Соробин. // Москва: РТУ МИРЭА. - 2020. – С. 159.

9. Chen, Ch. Интегральный оптический поток и его применение для мониторинга динамических объектов по видеопоследовательности / Ch. Chen, Sh. Ye, H. Chen, О. В. Недзьведь, С. В. Абламейко // Журнал прикладной спектроскопии. - 2017. – С. 138-146.

10. Cortes, C. Support-Vector Networks / C. Cortes, V. Vapnik // Machine Learning. - 2005. Vol. 20, Issue 3. - P. 273–297. - DOI https://doi.org/10.1007/BF00994018.

11. Dala, N. Histograms of Oriented Gradients for Human Detection / N. Dalal, B. Triggs // International Conference on Computer Vision & Pattern Recognition (CVPR '05). - 2005. - P. 1–8. - DOI: 10.1109/CVPR.2005.177

12. Pisner D.A., Schnyer D.M. Machine Learning. Methods and Applications to Brain Disorders. 2020. - P. 101–121. DOI: 10.1016/B978-0-12-815739-8.00006-7

13. Redmon, J. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon, S. Divvala, R. Girshick, A. Farhadi. // University of Washington IEEE, 2016. - P. 779-788 DOI: 10.1109/CVPR.2016.91

14. Szegedy, C. Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. // Google Inc. , 2014. - P. 1-12 DOI: 10.1109/CVPR.2015.7298594

ПРИЛОЖЕНИЕ А

Код программы. Пример аномального поведения

```
import matplotlib.pyplot as plt
import numpy as np

# Data for normal regions
normal_region_1 = np.random.normal(loc=[2, 2], scale=0.5, size=(100, 2))
normal_region_2 = np.random.normal(loc=[7, 7], scale=0.5, size=(100, 2))

# Anomalies
anomalies = np.array([[0, 0], [9, 9], [5, 1], [6, 6.5]])

# Plotting
plt.figure(figsize=(8, 8))
plt.scatter(normal_region_1[:, 0], normal_region_1[:, 1],
            label="Normal Region H1", alpha=0.7)
plt.scatter(normal_region_2[:, 0], normal_region_2[:, 1],
            label="Normal Region H2", alpha=0.7)
plt.scatter(anomalies[:, 0], anomalies[:, 1], color='red',
            label="Anomalies (o1, o2, O3)", marker='x', s=100)

# Highlight regions H1 and H2
circle1 = plt.Circle((2, 2), 1, color='blue',
                     fill=False, linestyle='--', linewidth=1.5)
circle2 = plt.Circle((7, 7), 1, color='green',
                     fill=False, linestyle='--', linewidth=1.5)
plt.gca().add_artist(circle1)
plt.gca().add_artist(circle2)

# Labels and legend
plt.title("Visualization of Normal Regions and Anomalies", fontsize=14)
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.axhline(0, color='black', linewidth=0.5, linestyle='--', alpha=0.5)
plt.axvline(0, color='black', linewidth=0.5, linestyle='--', alpha=0.5)
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

ПРИЛОЖЕНИЕ В

Код программы. Алгоритм на оптическом потоке

```
import cv2
import numpy as np

video_path = "video.mp4"
cap = cv2.VideoCapture(video_path)

scale_factor = 0.5

ret, old_frame = cap.read()
if not ret:
    cap.release()
    cv2.destroyAllWindows()
    exit()

old_frame_resized = cv2.resize(old_frame, (0, 0), fx=scale_factor,
                               fy=scale_factor)
old_gray = cv2.cvtColor(old_frame_resized, cv2.COLOR_BGR2GRAY)
old_gray_blurred = cv2.GaussianBlur(old_gray, (5, 5), 0)

motion_threshold = 10
speed_factor = 2

is_anomalous = False
previous_max_magnitude = np.max(np.zeros_like(old_gray))

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame_resized = cv2.resize(frame, (0, 0), fx=scale_factor,
                               fy=scale_factor)
    frame_gray = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2GRAY)
    frame_gray_blurred = cv2.GaussianBlur(frame_gray, (5, 5), 0)
    flow = cv2.calcOpticalFlowFarneback(old_gray_blurred,
                                        frame_gray_blurred,
                                        None,
                                        0.5, 3, 15, 3, 5, 1.2, 0)
    magnitude, angle = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    max_magnitude = np.max(magnitude)
    print(f"Max speed in frame: {max_magnitude:.2f}")

    if max_magnitude > previous_max_magnitude * speed_factor:
        is_anomalous = True
        print("Anomalous behavior detected!")
        cv2.putText(frame, 'Anomalous Behavior Detected',
                    (50, 50), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 2)
    else:
```

```
            is_anomalous = False

        step = 8
        height, width = frame_resized.shape[:2]
        for y in range(0, height, step):
            for x in range(0, width, step):
                fx, fy = flow[y, x]
                cv2.arrowedLine(frame_resized, (x, y), (int(x + fx),
                                int(y + fy)), (0, 255, 0), 1)

        frame_resized = cv2.resize(frame_resized,
                                   (old_frame.shape[1], old_frame.shape[0]))
        cv2.imshow('Dense Optical Flow with Anomaly Detection',
                   frame_resized)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        previous_max_magnitude = max_magnitude
        old_gray_blurred = frame_gray_blurred.copy()

cap.release()
cv2.destroyAllWindows()
```