МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Реализация алгоритма локализации возгорания на изображении»

Студент А.А. Дьяченко
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м.н., С.В. Баумгертнер
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

В данной выпускной квалификационной работе (ВКР) исследуется вопрос о применении сверточных нейронных сетей для распознавания объектов на изображении.

Объектом ВКР является распознавание объектов на изображении.

Предмет ВКР: алгоритм локализации и классификации объектов на изображении с применением сверточных нейронных сетей.

Целью работы является реализация алгоритма, способного распознавать на изображении такие классы объектов, как дым и огонь.

Структура ВКР состоит из введения, анализа принципов работы искусственных нейронных сетей и их применения для распознавания объектов на изображении, выбора и реализации алгоритма распознавания объектов на изображении, обучения и оценки точности нейронной сети, входящей в состав реализованного алгоритма, заключения.

В ведении определяется актуальность данной работы, основная цель исследования и задачи, выполнение которых необходимо для её достижения.

В первом разделе описываются основные принципы работы и обучения искусственных нейронных сетей и особенности сверточных нейронных сетей. Во втором разделе производится реализация алгоритма обнаружения объектов на изображении, основанного на сверточной нейронной сети. В третьем разделе производится обучение нейронной сети, входящей в состав алгоритма и оценка точности реализованного алгоритма. В заключении делаются выводы по проделанной работе и прогнозы по улучшению и применению реализованного алгоритма.

В результате ВКР реализован алгоритм для распознавания возгораний, который сможет помочь ликвидировать пожары на раннем этапе.

Данная работа делалась по заказу в рамках работы центра IT-Students

Бакалаврская работа содержит 55 страниц, 30 рисунков, 1 приложение, 3 части и 25 источник используемой литературы.

Abstract

The topic of the graduation work is *Implementing an algorithm to localize fire in an image.*

The present graduation work is devoted to using convolution neural networks for detecting objects in the image.

The object of the investigation is object recognition.

The subject of the investigation is the fire localization algorithm and classification of objects in the image by using convolution neural networks.

The aim of the investigation is to implement the algorithm allowing to detect fires in the image.

The graduation work may be divided into several logically connected parts which are introduction, analysis of artificial neural networks operation principles, analysis of how to train staff to use the artificial neural networks, selection and development of the object recognition algorithm, testing the neural network for detecting fire, as well as conclusions.

In introduction, we explain the relevance of the research, its main purpose and the tasks to be accomplished.

The first chapter gives full coverage to the existing methods of fire localization and classification of objects in an image. This chapter also describes the main idea of neural networks operation and reveals the methods of training staff to use neural networks.

The second section deals with choosing an algorithm for detecting objects in the image and implementing this algorithm.

The third chapter focuses on creating a training data set, training staff to use neural networks the corresponding algorithm consists of, as well as assessment of the model.

In conclusion, forecasts related to applying and improving the implemented fire detection algorithm is made. Overall, the results suggest that the implemented algorithm is proved to help extinguish fire in its early stages.

Содержание

Введение

В последнее время области применения машинного зрения быстро расширяются, охватывая все больше различных сфер жизнедеятельности. На сегодняшний день для реализации систем машинного зрения в основном применяются сверточные нейронные сети, которые являются одним из видов искусственных нейронных сетей. Именно благодаря способности к обучению, искусственные нейронные сети используются для решения сложных задач классификации и регрессионного анализа. Системы компьютерного зрения, использующие такой подход, обладают хорошей устойчивостью к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. На основе сверточных нейронных сетей реализовываются системы для беспилотного управления транспортными средствами, распознавания лиц и выявления различных заболеваний на медицинских снимках.

Актуальность данной работы обуславливается тем, что применение методов распознавания объектов на изображении может помочь уменьшить роль человеческого фактора в тех случаях, где первостепенно важным является быстрое реагирование на конкретные происшествия. Одним из таких случаев является возникновение пожаров.

Целью данной выпускной квалификационной работы (ВКР) является реализация алгоритма, способного распознавать возгорания на изображении, а именно определять наличие и правильное местоположение таких объектов, как дым и огонь на изображении.

Объектом ВКР является распознавание объектов на изображении.

Предмет ВКР: алгоритм локализации и классификации объектов на изображении с применением сверточных нейронных сетей.

Для достижения поставленной цели были определены следующие задачи:

– описать принцип работы и обучения искусственных нейронных сетей;

– разобрать особенности архитектуры сверточных нейронных сетей;

– выбрать и реализовать алгоритм для распознавания объектов на изображении на основе сверточных нейронных сетей;

– обучить нейронную сеть, входящую в состав алгоритма распознавать возгорания;

– проверить точность работы реализованного алгоритма.

Бакалаврская работа состоит из введения, анализа принципов работы искусственных нейронных сетей и их применения для распознавания объектов на изображении, выбора и реализации алгоритма распознавания объектов на изображении, основанного на использовании сверточных нейронных сетей, обучения и оценки точности нейронной сети, входящей в состав реализованного алгоритма и заключения.

В первом разделе проводится анализ искусственных нейронных сетей и методов их обучения, а также рассматривается принцип работы сверточных нейронных сетей.

Во втором разделе анализируются современные методы распознавания объектов на изображении с применением сверточных нейронных сетей и производится реализация метода, наиболее подходящего для распознавания возгораний.

В третьем разделе описывается процесс создания обучающей выборки для обучения сверточной нейронной сети, входящей в состав алгоритма обнаружения объектов. Производится обучение сверточной нейронной сети и оценка точности реализованного алгоритма.

Использование реализованного алгоритма для распознавания возгораний в совокупности с беспилотными транспортными средствами позволит детектировать пожары на ранних этапах и быстро устранять их. Подобное решение поможет уменьшить количество крупных пожаров в лесах, городах и заповедниках, а также сократить затраты на их восстановление.

1 Анализ принципов работы искусственных нейронных сетей и их применения для распознавания объектов на изображении

1.1 Обзор существующих методов распознавания объектов на изображении

В данной работе необходимо реализовать алгоритм, который будет способен распознавать на изображении объекты, принадлежащие к таким классам как дым и огонь, так как возгорание может быть визуально выражено одним из этих классов объектов или сразу двумя. Процесс распознавания объекта на изображении включает в себя 2 этапа: нахождение местоположения объекта и присвоение ему конкретного класса, исходя из его свойств. Иногда для решения задачи обнаружения объекта на изображении применяются такие подходы, как наложение цветовых фильтров на изображение, выделение и анализ контуров, сопоставление с шаблонами, а также выявление и поиск особых точек. Данные методы позволяют выделять образы, которые могут помочь определить класс объекта на изображении. К сожалению, перечисленные способы обнаружения объекта не подходят для распознавания возгорания из-за плохой обобщающей способности, поскольку возгорание не имеет конкретных признаков, а дым и огонь могут иметь различную форму, яркость и положение на изображении. Именно поэтому для достижения цели данной работы было решено использовать методы машинного обучения.

Машинное обучение является подразделом искусственного интеллекта – области, занимающейся разработкой систем, которые смогли бы справляться с задачами, решение которых считается исключительно прерогативой человека. В настоящее время наиболее популярным способом классификации объектов на изображении является такой метод машинного обучения, как искусственные нейронные сети, а именно их специальная архитектура, называемая сверточной нейронной сетью. Данный подход обеспечивает частичную устойчивость к изменениям масштаба, смещениям,

поворотам, смене ракурса и прочим искажениям, но основной особенностью сверточных нейронных сетей является их обучаемость [10].

Недостатком сверточных нейронных сетей является то, что они не способны классифицировать множество объектов на одном изображении, а для локализации возгорания необходимо распознавать несколько классов объектов [17]. Для задач такого рода существуют специальные алгоритмы на основе сверточных нейронных сетей, такие как R-CNN, YOLO и SSD, которые позволяют производить локализацию и классификацию множества объектов на изображении.

Таким образом, были рассмотрены основные подходы, применяемые для распознавания объектов на изображении и сделан вывод, что для локализации возгораний на изображении необходимо использовать методы, основанные на применении сверточных нейронных сетей. Для этого необходимо понять принцип работы искусственной нейронной сети.

1.2 Анализ принципа работы искусственной нейронной сети

Искусственная нейронная сеть – это математическая модель, построенная на основе теории деятельности головного мозга, применяемая для решения различных прикладных задач. Она имитирует работу нейронов головного мозга - простых однотипных вычислительных элементов, которые, обмениваясь сигналами между собой, способны выполнять сложные задачи. Искусственные нейроны представляют из себя функцию, на вход которой подаются некоторые значения, помноженные на корректируемые параметры – веса, после чего результаты произведения суммируются и полученное значение проходит через нелинейную функцию, являющуюся показателем силы возбуждения нейрона [20]. Искусственные нейронные сети способны объединять в себе множество таких нейронов и позволяют решать ряд задач, таких как: кластеризация, классификация, регрессионный анализ, компьютерное зрение, распознавание речи и машинный перевод. Особенностями искусственных нейронных сетей является обучаемость и

нелинейность, которые позволяют находить сложные зависимости между данными и учитывать опыт для улучшения результата своей работы [20]. Структура искусственной нейронной сети представляет из себя направленный граф, в качестве узлов которого выступают нейроны, а в качестве дуг – связи с весовыми коэффициентами. Данная структура показана на рисунке 1.1.

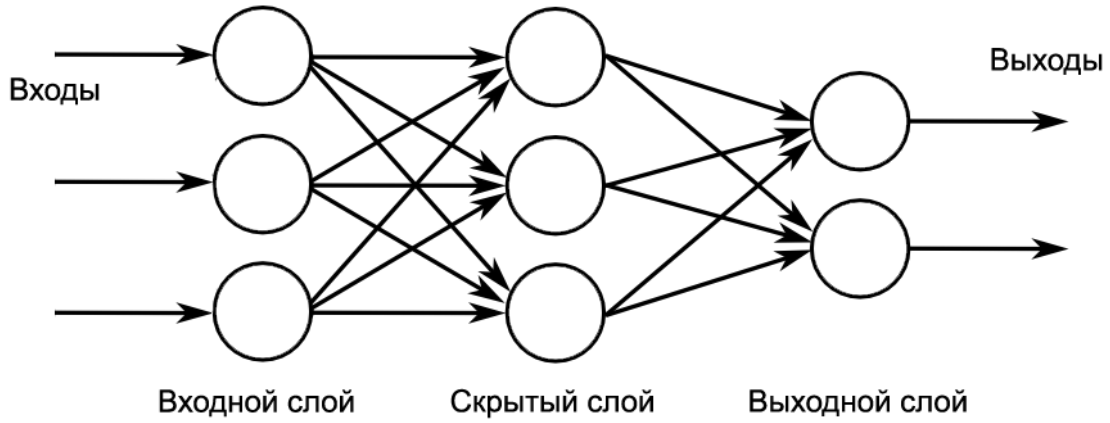

Рисунок 1.1 – Структура искуственной нейронной сети

Понять принцип работы искусственной нейронной сети можно на примере персептрона – одной из первых моделей нейронной сети. Модель персептрона представлена на рисунке 1.2.

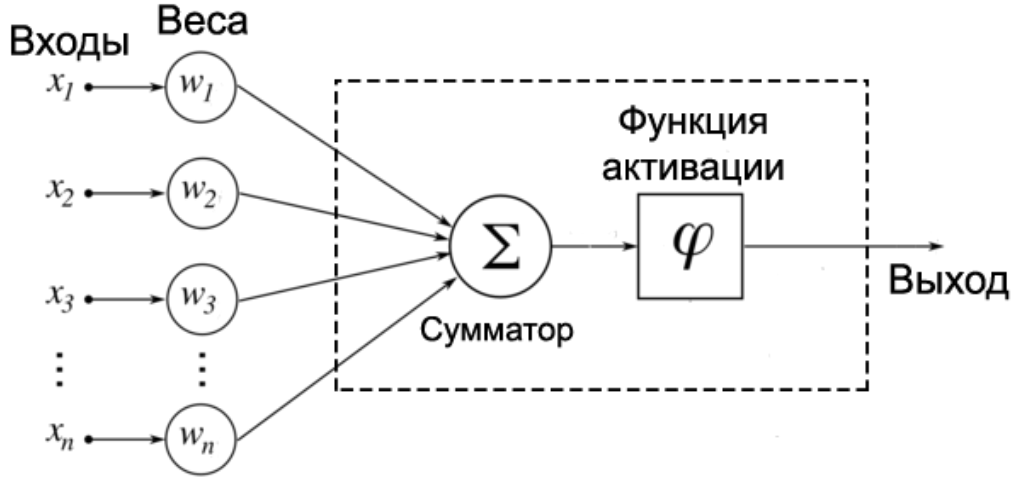

Рисунок 1.2 – Модель персептрона

Между входом или иначе входными нейронами, получающими сигнал, и сумматором находятся связи, каждая из которых имеет свой весовой

коэффициент, определяющий значимость соответствующего входа для формирования выходного значения. Сумматор вычисляет взвешенную сумму посредством сложения входных сигналов, помноженных на свои весовые коэффициенты. Данная сумма подается в функцию активации, которая определяет выходное значение нейрона [1].

Функция активации определяет должен ли нейрон быть активирован. Другими словами, она производит нормализацию проходящего через нее сигнала и представляет результат работы нейронной сети в нужном диапазоне, в зависимости от типа функции активации [20]. Без данной функции любая архитектура нейронной сети представляет из себя обычную модель линейной регрессии, поскольку именно она позволяет выполнять нелинейное преобразование для входных данных и решать более сложные задачи [23]. Разновидностей функций активации огромное количество, основными из них являются: ступенчатая, линейная, сигмоидальная, гиперболический тангенс, функция усеченного линейного преобразования (ReLU), которые представлены на рисунке 1.3.

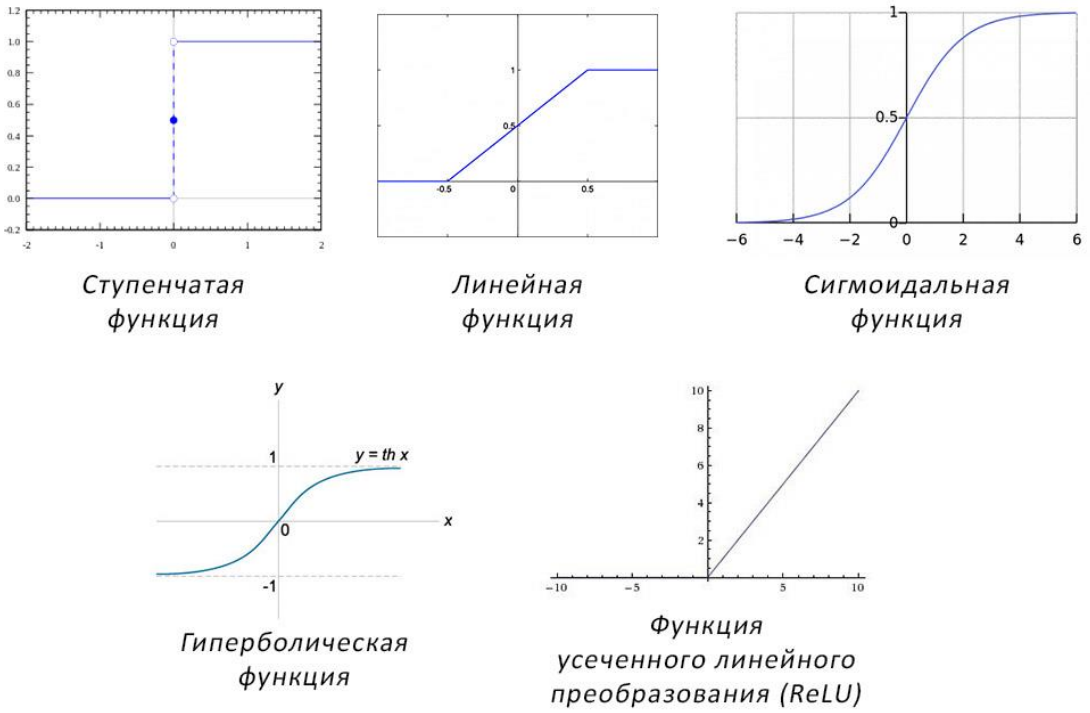

Рисунок 1.3 – Основные функции активации

Функция активации выбирается исходя из типа задачи, решаемой нейронной сетью.

Персептрон является простейшим видом нейронной сети, который позволяет решать простые задачи классификации, но не предназначен для решения более сложных задач. Именно для этого были разработаны многослойные нейронные сети прямого распространения, которые в отличии от персептрона состоят из нейронов, сгруппированных по слоям. В таких сетях нейроны каждого слоя соединены с нейронами следующего слоя, но не связаны друг с другом. Все слои данной нейронной сети, за исключением первого и последнего слоя называются скрытыми [15].

Одной из важных составляющих почти всех архитектур нейронных сетей является использование смещения во входном и скрытых слоях. Смещение – это нейрон, не принимающий входных данных, поскольку его входной сигнал всегда равен единице. Данный параметр позволяет сдвигать функцию активации влево и вправо, что помогает сделать обучаемую модель более гибкой [3]. Многослойная нейронная сеть, содержащая нейроны смещения представлена на рисунке 1.4.

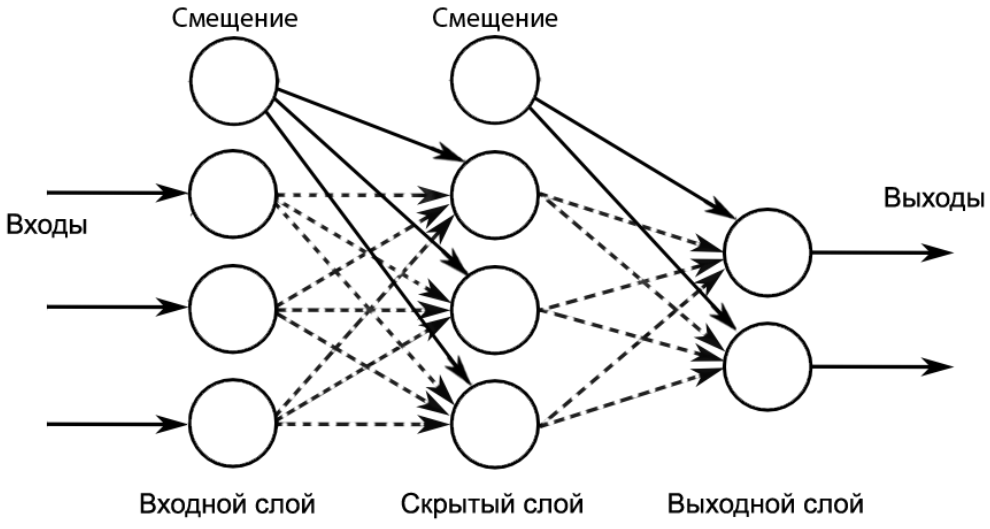

Рисунок 1.4 – Многослойная нейронная сеть с нейронами смещения

Таким образом, существует огромное число различных архитектур нейронных сетей в зависимости от количества скрытых слоев, нейронов в

данных слоях и видов связей между нейронами, каждая из которых лучше справляется со своим типом задач, однако для того, чтобы нейронная сеть научилась решать задачи разного рода, её необходимо обучать.

1.3 Анализ методов обучения искусственных нейронных сетей

Одной из главных особенностей искусственных нейронных сетей является их обучаемость. Основная цель обучения состоит в том, чтобы минимизировать ошибку между выходными значениями сети, получаемыми при прямом распространении ошибки и ожидаемыми значениями. Прямым распространением ошибки называется процесс получения выходных данных нейронной сети на основе одного экземпляра тренировочных данных [3]. Сама ошибка искусственной нейронной сети вычисляется с помощью функции потерь. Наиболее используемыми функциями потерь являются среднеквадратичное отклонение и среднее абсолютное отклонение. Данные функции потерь представлены в формуле (1).

$$\begin{gathered} MSE: (y' - y)^2, \\ MAE: |y' - y|, \end{gathered} \tag{1}$$

где y' – ожидаемое значение;

y – результат работы сети.

Существующие методы обучения нейронных сетей можно разделить на стохастические и детерминированные. Стохастические методы основаны на корректировании весовых коэффициентов сети случайным образом, которые сохраняются только в том случае, если внесенные изменения приводят к уменьшению ошибки. Детерминированные методы пытаются точно определять и корректировать только те весовые коэффициенты, которые вносят наибольший вклад в ошибку [1].

Для обучения нейронных сетей в основном используется один из детерминированных методов, называемый градиентным спуском [2]. Данный метод оптимизации позволяет минимизировать функцию потерь нейронной сети путем итеративного корректирования весовых коэффициентов. Принцип

градиентного спуска на примере одного весового коэффициента представлен на рисунке 1.5.

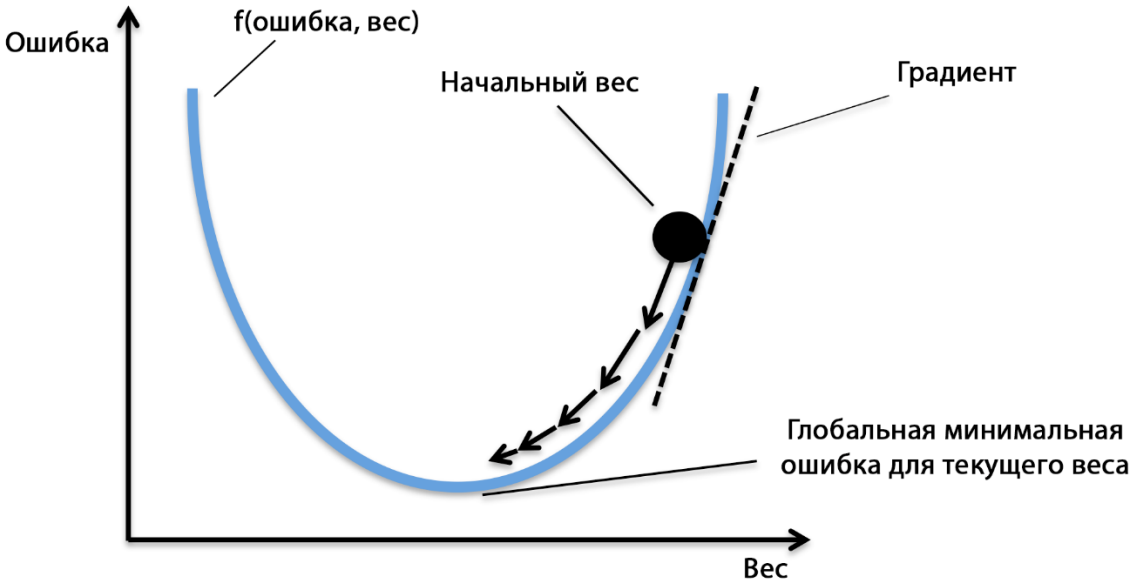

Рисунок 1.5 – Принцип градиентного спуска

Обучение по алгоритму градиентного спуска представляет из себя поиск градиента функции f, изображенной на рисунке 1.5 в конкретной точке, являющейся текущей ошибкой нейронной сети . Данный градиент будет указывать направление возрастания функции. Таким образом, чтобы минимизировать ошибку, необходимо сдвигать значение весового коэффициента в сторону, противоположную направлению градиента [18]. Корректирование весового коэффициента по методу градиентного спуска представлено в формуле (2).

$$w^{+} = w - \eta \cdot \nabla E, \qquad (2)$$

где w^{+}– новое значение весового коэффициента;

w – текущее значение весового коэффициента;

∇E – градиент ошибки;

η – константа, характеризующая скорость обучения, то есть значение на величину которого изменяются новые весовые коэффициенты.

При выборе маленькой скорости обучения, минимизация функции потерь будет происходить слишком долго, а при выборе большого значения можно так и не достигнуть минимума, перешагнув его [18].

Для наглядности на рисунке 1.5 представлена зависимость ошибки от одного весового коэффициента, однако в многослойных нейронных сетях количество таких коэффициентов может достигать десятки и сотни тысяч. Для того чтобы минимизировать функцию потерь такой сети, необходимо найти градиенты для всех весовых коэффициентов, то есть рассчитывать их частные производные. Метод, позволяющий это сделать, называется обратным распространением ошибки. Суть данного метода заключается в нахождении частных дифференциалов по каждому весовому коэффициенту, что позволяет определить вклад каждого из этих весов в общую ошибку [20]. Вначале вычисляется ошибка выходного слоя и ее значение передается на скрытый слой, стоящий перед выходным. Дальше значение ошибки этого скрытого слоя передается на другой слой, стоящий перед ним. Данная операция повторяется для всех слоев, начиная от последнего и заканчивая первым, позволяя найти производную потерь по весам этих слоев, которая позволяет определить в каком направлении необходимо регулировать веса для уменьшения общих потерь. Ошибка весовых коэффициентов последнего и всех остальных слоев вычисляется по-разному [2]. Нахождение вклада весовых коэффициентов последнего слоя в общую ошибку считается по формуле (3).

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial i_k}{\partial w_{j,k}} \cdot \frac{\partial o_k}{\partial i_k} \cdot \frac{\partial E}{\partial o_k} = o_j \cdot \frac{\partial f_k(S_k)}{\partial S_k} \cdot \delta_j, \qquad (3)$$

где $w_{j,k}$ – весовой коэффициент, соединяющий нейроны j и k;

$\frac{\partial E}{\partial w_{j,k}}$ – вклад веса $w_{j,k}$ в общую ошибку;

$\frac{\partial i_k}{\partial w_{j,k}}$ – вклад веса $w_{j,k}$ во взвешенную сумму нейрона k;

$\frac{\partial o_k}{\partial i_k}$ – вклад взвешенной суммы в выходной сигнал нейрона k;

$\frac{\partial E}{\partial o_k}$ – вклад выходного сигнала нейрона k в общую ошибку;

o_j – выходной сигнал нейрона j;

$\frac{\partial f_k(S_k)}{\partial S_k}$ – производная функции активации нейрона j;

δ_j – ошибка выходного сигнала нейрона j, находящаяся с помощью функции потерь.

Нахождение вклада весовых коэффициентов всех слоев, кроме последнего в общую ошибку считается по формуле (4).

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial i_k}{\partial w_{j,k}} \cdot \frac{\partial o_k}{\partial i_k} \cdot \left(\sum_l \frac{\delta_l^{+1}}{\partial o_k}\right), \quad (4)$$

где $\frac{\delta_l^{+1}}{\partial o_k}$ – вклад выходного сигнала нейрона k в ошибку выходного сигнала нейрона l на следующем слое, стоящим за слоем нейрона k.

Данный дифференциал вычисляется как умножение весового коэффициента $w_{k,l}$ на произведение $\frac{\partial o_k}{\partial i_k} \cdot \frac{\partial E}{\partial o_k}$, вычисляемое в формуле (3).

Таким образом, обучение нейронной сети с помощью градиентного спуска происходит посредством итеративной операции корректирования весовых коэффициентов [17]. Одна итерация настройки весов выглядит следующим образом:

– производится прямое распространение ошибки для каждого примера тренировочных данных и находится ошибка с помощью функции потерь;

– вычисляется сумма квадратов всех ошибок;

– с помощью обратного распространения ошибки корректируются значения весовых коэффициентов нейронной сети.

Недостатком градиентного спуска является то, что настройка весовых коэффициентов, а следственно и обучение модели происходит очень долго, так как необходимо вычислить ошибки для всего тренировочного набора данных, который может быть очень большим [19]. Данную проблему решает

использование модифицированного алгоритма градиентного спуска, называемого стохастическим градиентным спуском. Модифицированный алгоритм отличается тем, что производит корректировку весов на основе каждого образца или небольшой партии тренировочных данных. Из-за того, что корректирование весовых коэффициентов на основе небольшой партии данных может быть достаточно неточным, функция потерь минимизируется не плавно, а рывками, но сходится к локальному минимуму намного быстрее [18]. Сравнение процесса обучения с использованием разного градиентного спуска приведено на рисунке 1.6.

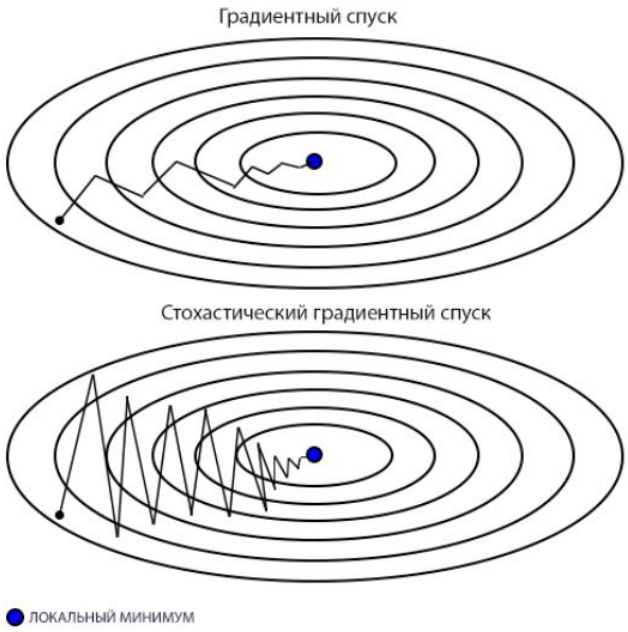

Рисунок 1.6 – Процесс обучения при использовании различных вариантов градиентного спуска

Основной проблемой алгоритма градиентного спуска является то, что при большом количестве весовых коэффициентов найденный минимум функции может оказаться локальным, а не глобальным, поэтому обычно в начале обучения весовые коэффициенты задаются случайным образом, чтобы попытаться найти новый минимум функции и еще больше минимизировать функцию потерь. Если же функцию потерь не удается минимизировать, то следует внести изменения в структуру искусственной нейронной сети [18].

Таким образом, был рассмотрен основной детерминированный метод обучения нейронных сетей, называемый градиентным спуском. Однако для того, чтобы научить нейронную сеть распознавать образы на изображении, необходимо использовать специальную архитектуру искусственной нейронной сети, называемую сверточной.

1.4 Описание особенностей сверточной нейронной сети

Сверточная нейронная сеть является одной из наиболее популярных архитектур нейронных сетей, применяемая для анализа визуальных образов и классификации изображений. Данная архитектура спроектирована на основе принципа зрительной коры головного мозга, в которой отдельные участки клеток реагируют на возбуждения в своих конкретных областях поля зрения, называемых рецептивными полями. При восприятии различных образов, в зрительной коре активируются разные группы нейронов. В целом, архитектура сверточной нейронной сети позволяет выделять в данных различные признаки, начиная от очень простых и заканчивая более сложными [22].

Свёрточная нейронная сеть состоит из чередующихся слоев свертки и подвыборки, после которых следуют полносвязные слои. Архитектура свёрточной нейронной сети представлена на рисунке 1.7.

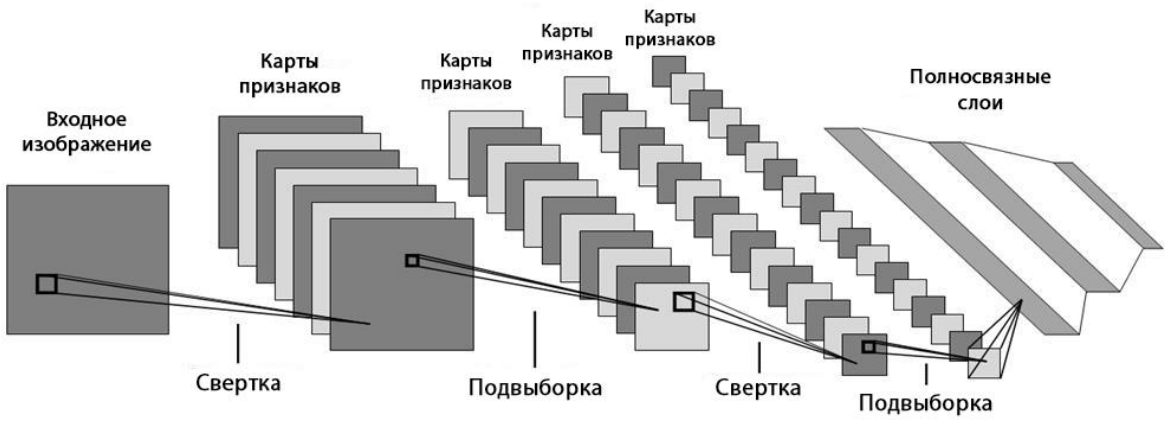

Рисунок 1.7 – Архитектура свёрточной нейронной сети

Свёрточный слой всегда является первым в сверточной нейронной сети и реализует операцию свёртки, которая отражает реакцию отдельного нейрона на конкретную область поля зрения [14]. На вход свёрточного слоя подается изображение в виде матрицы пикселей, размерность которой определяется размером изображения. Если изображение черно-белое, то на вход подается одна матрица, каждый пиксель которой описывается значением яркости в диапазоне от 0 до 255. Цветные изображения представляются тремя матрицами, каждая из которых отвечает за один из каналов цветовой модели RGB, значения матриц описывают интенсивность цвета пикселей. В качестве нейрона выступает некое ядро свертки – матрица, элементы которой являются весовыми коэффициентами [14]. Глубина ядра свертки должна быть такой же, как и глубина изображения, к примеру для цветного изображения применяется ядро свертки, состоящее из трех матриц, каждая из которых применяется для свертки по одному из каналов цветного изображения. Размерность матрицы ядра свертки обычно варьируется от 3 до 7 и определяет количество признаков, которые необходимо объединить в один новый признак более высокого уровня. Операция свертки начинается с определения рецептивного поля, размер которого равен ядру свертки. Данное поле скользит с некоторым смещением по исходному изображению и пиксели изображения, попавшие в него, поэлементно перемножаются с ядром свертки. Элементы новой матрицы, полученной в результате перемножения, суммируются и образуют новый выходной пиксель. Операция свертки представлена на рисунке 1.8.

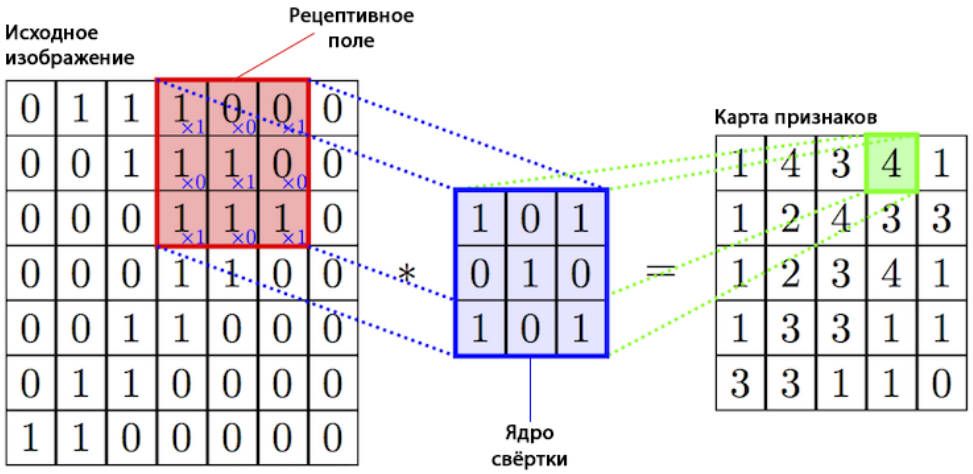

Рисунок 1.8 – Операция свертки

Рисунок 1.8 описывает операцию свертки для черно-белого изображения. Свертка цветного изображения отличается лишь тем, что к каждому цветовому каналу применяется одна из матриц, составляющих трехмерное ядро свертки, после чего полученные матрицы суммируются в одну результирующую матрицу. Операция свертки для цветного изображения представлена на рисунке 1.9.

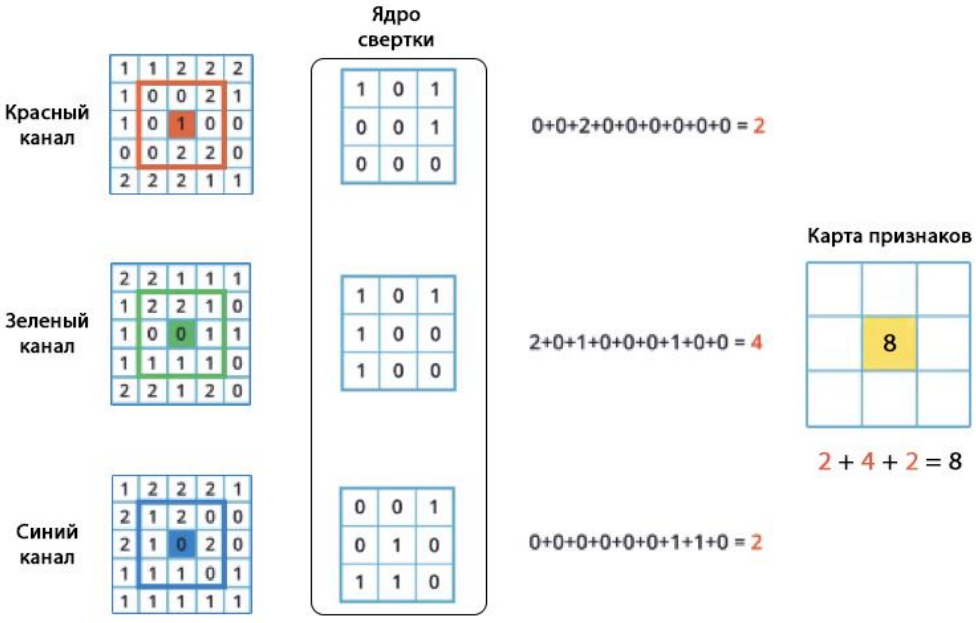

Рисунок 1.9 – Свертка цветного изображения

Из-за того, что граничные пиксели исходного изображения не оказываются в центре рецептивного поля, матрица, полученная в результате свертки, имеет меньшую размерность, чем исходное изображение. Чтобы избежать постоянного сжатия изображения при выполнении свертки и сохранить информацию о его границах, вокруг изображения добавляется рамка из нулей, ширина которой зависит от размера ядра свертки [14]. Данная операция называется заполнением (zero-padding). Свертка цветного изображения с заполнением представлена на рисунке 1.10.

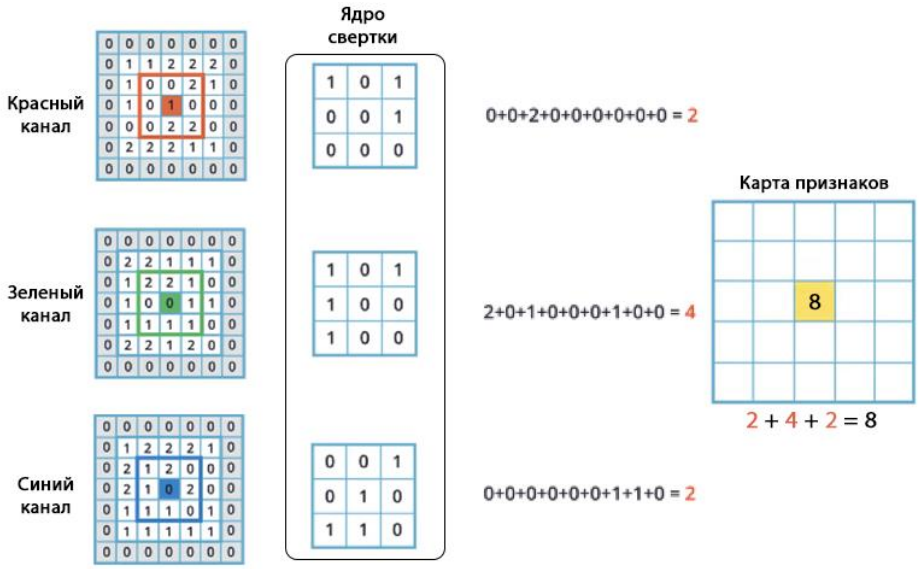

Рисунок 1.10 – Свертка цветного изображения с применением заполнения

Также, как и в многослойных нейронных сетях, в данной архитектуре присутствует смещение – значение, изменяемое в процессе обучения, которое складывается с каждым элементом выходной матрицы. Перед поступлением на следующий слой, полученная матрица проходит через функцию активации. Обычно в сверточных слоях в качестве функции активации используют функцию усеченного линейного преобразования, которая преобразовывает все отрицательные значения в 0, а остальные не изменяет [21]. Данная функция была представлена на рисунке 1.3.

Итак, в результате свертки получается матрица, называемая картой признаков, на которой выделен конкретный признаки, такой как: различные прямые и кривые линии, полуокружности или более сложные признаки, в

зависимости местонахождения сверточного слоя в модели нейронной сети. Если первый сверточный слой выявляет базовые признаки, то последующие сверточные слои выявляют признаки все более и более высокого уровня, обобщая информацию на изображении [14]. Сверточные слои чаще всего используют более одного ядра свертки, каждый из которых идентифицирует конкретный признак, поэтому в результате свертки получается набор различных числовых матриц, число которых характеризует глубину карты признаков. Чем больше количество используемых ядер свертки, тем более подробную информацию можно получить об исходном изображении.

После слоя свертки обычно следует слой подвыборки. Данный слой уменьшает пространственную размерность карты признаков, это достигается путем замены некоторой области пикселей одним пикселем. Необходимость этой операции объясняется тем, что важна именно информация о наличии конкретных признаков на изображении, а не их местоположение, поэтому оно сжимается до менее подробного, состоящего из доминирующих признаков [14]. Кроме того, операция подвыборки уменьшает количество параметров сверточной сети, тем самым снижая вычислительную нагрузку. На практике используется всего два типа подвыборки: по среднему значению, в котором некоторая области пикселей объединяется в среднее значение интенсивности, и по максимальному значению, в котором та же область объединяется в пиксель с максимальной интенсивностью. Чаще всего используют подвыборку по максимальному значению, так как она более четко выделяет ключевые признаки, подавляя лишние шумы на изображении [14]. При выполнении операции подвыборки учитываются два параметра: размер ядра подвыборки, скользящего по изображению, которое сворачивает попавшие в него пиксели до 1 пикселя и размер шага в пикселях, на которое сдвигается ядро подвыборки. Чаще всего используют пара значений параметров: 2x2, которая уменьшает размерность карты признаков в два раза. Пример выполнения операции подвыборки приведен на рисунке 1.11.

Рисунок 1.11 – Пример выполнения подвыборки по максимальному значению

Для того, чтобы выявить сложные признаки на изображении, сверточные сети содержат множество чередующихся слоев свертки и подвыборки. Таким образом, с каждой парой слоев свертки глубина карты признаков изменяется в зависимости от числа ядер свертки, а их размерность уменьшается.

Конечными слоями сверточной нейронной сети являются полносвязные слои, с помощью которых происходит классификация объекта на изображении. Весовые коэффициенты данных слоев настраиваются в процессе обучения для определения метки класса на основе выявленных признаков [20]. Перед подачей полученной карты признаков на полносвязный слой, происходит объединение слоев карты признаков в вектор, где каждый элемент показывает вероятность присутствия конкретного признака на изображении. К примеру, чтобы изображение было классифицировано как лицо, вероятность присутствия таких признаков как глаза, рот и нос должна быть высокой [14]. Последний полносвязный слой содержит количество нейронов, равное количество определяемых классов и с помощью выбранной функции активации выводит вероятности присутствия каждого класса на изображении.

Таким образом, были рассмотрены особенности архитектуры сверточной нейронной сети и разобран процесс извлечения признаков из изображения для классификации находящегося на нем объекта.

Итак, в данном разделе были рассмотрены основные принципы работы искусственных нейронных сетей, особенности их построения и методы, с помощью которых происходит их обучение. Кроме того, был подробно разобран процесс применения сверточной нейронной сети для классификации объекта на изображении. Таким образом, для реализации алгоритма локализации возгорания на изображении, необходимо проанализировать существующие методы распознавания объектов на основе сверточных нейронных сетей и выбрать наиболее подходящий.

2 Выбор и реализация алгоритма для локализации возгораний на основе сверточных нейронных сетей

2.1 Описание принципа работы алгоритма Single Shot MultiBox Detector

Задача классификации объекта на изображении решается в основном с помощью сверточной нейронной сети, однако в данной работе необходимо реализовать алгоритм, который определяет не только наличие возгорания на изображении, но и его местоположение. Для локализации объектов на изображении существуют такие методы, как SSD, YOLO или семейство алгоритмов R-CNN [6]. Первые два алгоритма работают быстрее, так как для поиска объектов изображение проходит через нейронную сеть только один раз, что является важной особенностью, позволяющей обрабатывать большее количество кадров за меньшее время. Для реализации алгоритма локализации возгорания на изображении был выбран метод распознавания объектов SSD, так как он достигает лучшего баланса между точностью и быстродействием.

Алгоритм SSD (Single Shot MultiBox Detector) позволяет классифицировать множество различных объектов на изображении, выделяя их ограничивающими прямоугольниками [8]. Существует две версии алгоритма SSD: для изображений размерностью 300×300 и 512×512 пикселей. Хотя SSD512 имеет более высокий показатель точности определения объектов, в данной работе будет реализован алгоритм SSD300, так как уменьшение размерности входного изображения способствует увеличению скорости работы алгоритма. Логически можно разделить алгоритм на 2 части: извлечение карты признаков для входного изображения и распознавание объектов. Для извлечения признаков SSD использует многослойную сверточную нейронную сеть VGG16, а распознавание объектов выполняет специальный модуль, состоящий из двух сверточных слоев: для регрессии и классификации.

Сверточная сеть VGG16, используемая для получения карт признаков очень проста в реализации, так как имеет достаточно однородную структуру,

состоящую из свертки с размером ядра 3×3 и подвыборки с размером ядра 2×2. Данная нейронная сеть содержит 16 слоев, не считая слои подвыборки, из которых 13 являются сверточными, а оставшиеся 3 полносвязными. Структура сверточной нейронной сети VGG16 представлена на рисунке 2.1.

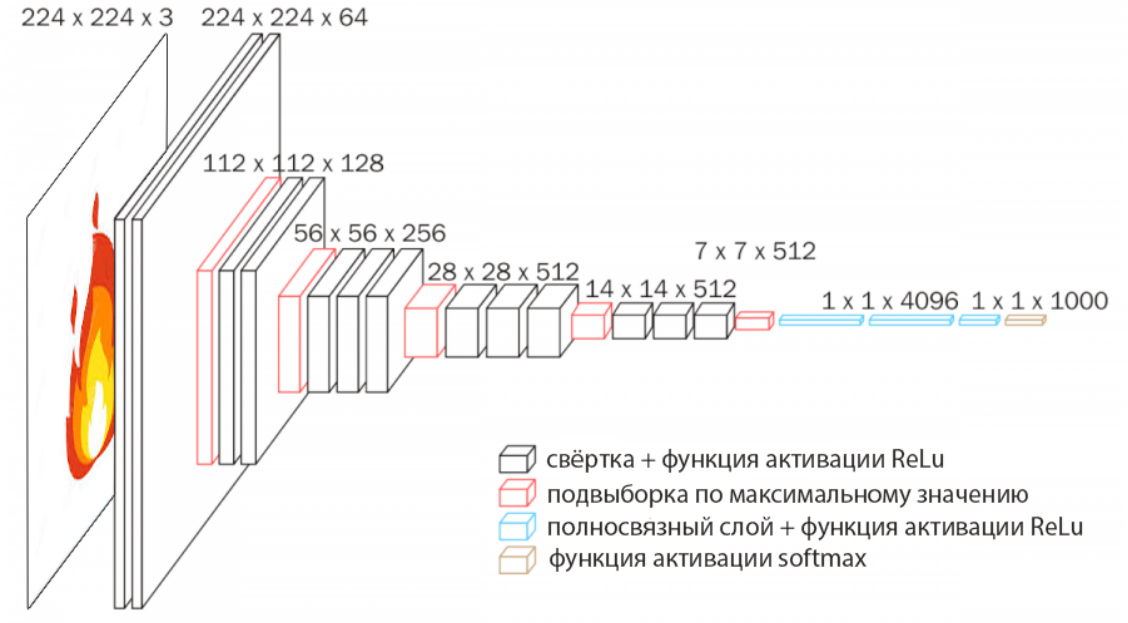

Рисунок 2.1 – Структура сверточной нейронной сети VGG16

Так как данная сверточная сеть используется лишь для извлечения карт признаков и нет необходимости проводить классификацию изображения на данном этапе, необходимо внести изменения в последние слои. В первую очередь изменяется последний слой подвыборки, чтобы не уменьшать размерность карты признаков. Для этого ядру слоя подвыборки устанавливается размер 3×3 с шагом 1 и с нулевым заполнением по краям, необходимым для того, чтобы карта признаков не сжималась. После этого два первых полносвязных слоя преобразовываются в сверточные слои, а последний слой удаляется. Учитывая то, сверточный слой отличается от полносвязного лишь тем, что связан не со всеми, а лишь с локальными нейронами в предыдущем слое и использует общий набор весовых коэффициентов и смещение для одного слоя, можно производить преобразование между сверточными и полносвязными слоями [14]. Для этого необходимо задать для него размер ядра свертки, равный размерности карты

признаков, поступающей на вход. В данном случае, карта признаков, после измененного слоя подвыборки имеет размер 19×19, что является слишком большим ядром свертки, вычисление которой будет трудозатратным. Для этого используется операция свертки с расширением, это означает, что выбирается ядро свертки меньшего размера, однако элементы ядра свертки умножаются на элементы входных данных не последовательно, а с некоторым промежутком. В данном случае берется 1024 сверточных ядер размера 3×3 с расширением 6, это означает, что сверточное ядро свертки 3×3 смотрит на рецептивное поле 13×13 и умножение весовых коэффициентов ядра свертки на карту признаков происходит через каждые 5 элементов. Данный подход позволяет не вычислять большое число параметров по сравнению с ядром свертки 19×19, однако качество выделения признаков снижается [14]. Размерность получаемой карты признаков в результате данной свертки можно узнать с помощью формулы (5).

$$FMS = (W - F + 2P) : S + 1, \quad (5)$$

где FMS – размер новой карты признаков;

W – размер входного представления;

F – размер рецептивного поля ядра свертки;

P – ширина рамки граничного заполнения;

S – шаг свертки.

Таким образом, размер полученной карты признаков составляет 19×19 с глубиной 1024. Следующий полносвязный слой переделывается в сверточный слой с количеством сверточных ядер 1024 и размером ядра свертки 1×1. Подобные слои с единичным ядром свертки и количеством сверток равным глубине входной карты признаков не изменяют размерность и глубину входной карты признаков, а лишь уточняют признаки, извлеченные на предыдущих этапах [14]. Таким образом, измененная структура сверточной нейронной сети готова для извлечения карт признаков объектов и может быть связана с остальными слоями архитектуры SSD, представленной на рисунке 2.2.

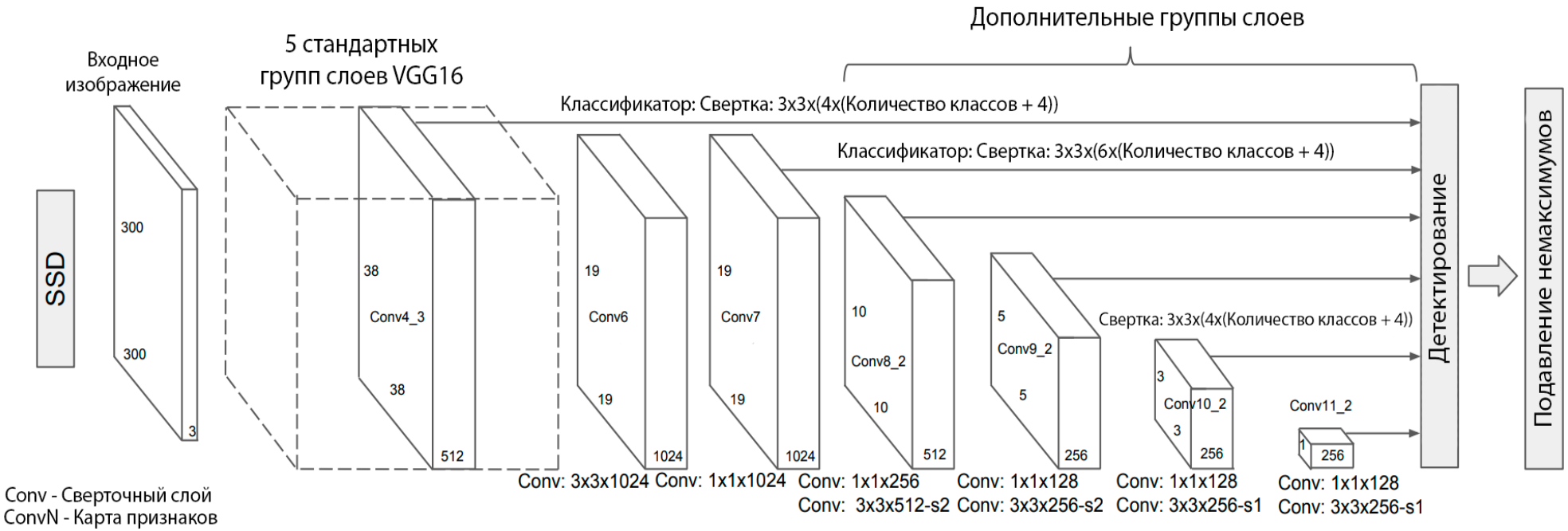

Рисунок 2.2. – Архитектура нейронной сети алгоритма SSD

На рисунке 2.2 видно, что помимо 5 стандартных групп слоев и 2 модифицированных слоев стандартной сети VGG16 добавлены 4 дополнительные группы сверточных слоев, содержащих по 2 сверточных слоя. Первый уменьшает глубину карты признаков за счет свертки с размером ядра 1×1, а второй сверточный слой изменяет размерность карты признаков, используя ядро свертки 3×3 и параметры, такие как шаг и нулевое заполнение, отличающиеся для разных слоев. В 8 и 9 сверточных блоках используется шаг свертки 2 и нулевое заполнение шириной 1. Для последующих слоев используется шаг свертки 1 без нулевого заполнения. Такая структура сверток в дополнительных сверточных слоях алгоритма SSD помогает с минимальной вычислительной нагрузкой производить масштабирование карты признаков [8]. Данная операция необходима, поскольку для распознавания объектов разного размера алгоритм SSD использует карты признаков разного масштаба, получаемые с нескольких слоев. Как видно на изображении 2.2, всего используется 6 размерностей карт признаков: 38×38 и 19×19, получаемые с четвертой группы слоев оригинальной сети VGG16 и седьмого полносвязного слоя, преобразованного в сверточный слой, а также 10×10, 5×5, 3×3 и 1×1, получаемые с дополнительных групп сверточных слоев.

Детектирование объектов в алгоритме SSD производится двумя отдельными сверточными слоями, на которые поступают описанные выше карты признаков. Первый сверточный слой предсказывает местоположения объекта, а второй присваивает ему конкретный класс [8]. Таким образом, для распознавания объектов на изображении используются методы регрессии ограниченного прямоугольника по модифицированному алгоритму MultiBox и классификации. Схема модуля детектирования объектов на изображении представлен на рисунке 2.3.

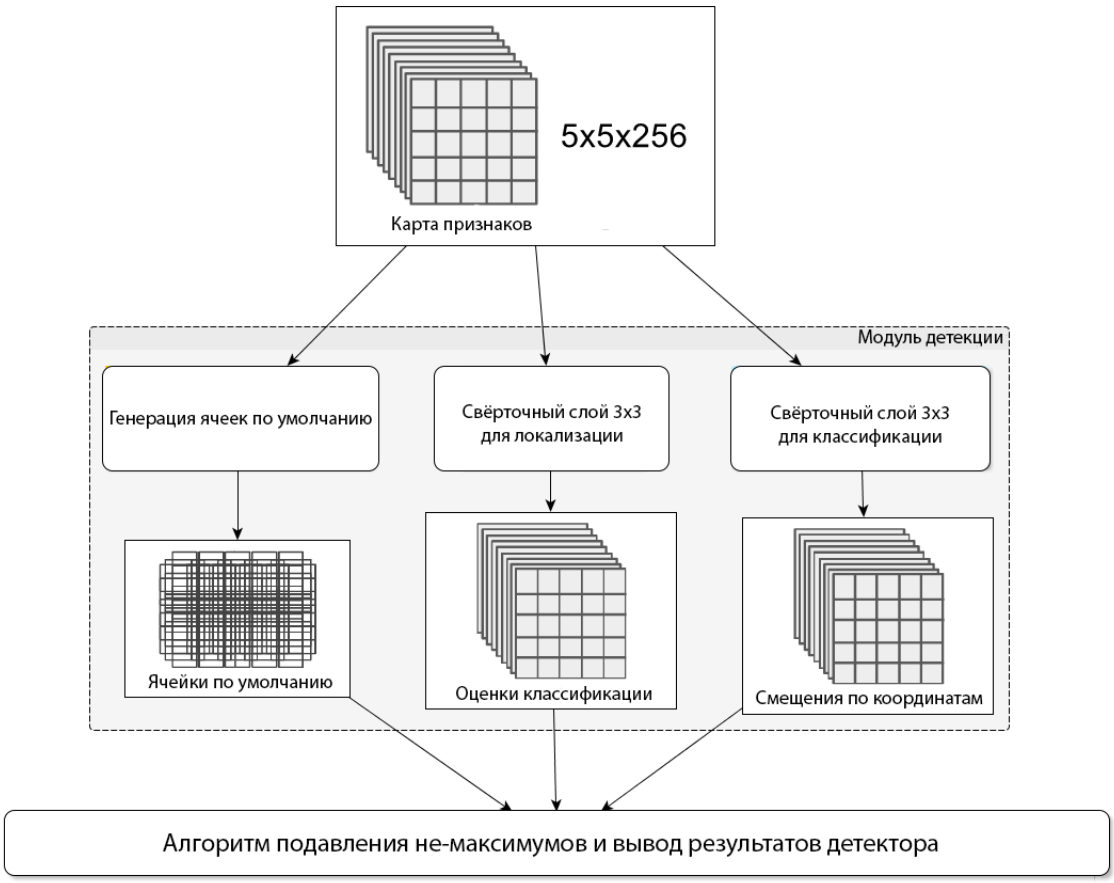

Рисунок 2.3 – Схема модуля детектирования алгоритма SSD

Первым делом, для каждой полученной с разных слоев карты признаков генерируются наборы ограничивающих прямоугольников фиксированного размера, называемые рамками по умолчанию. Данные ограничивающие рамки распределяются по всей карте признаков и привязываются к конкретным её точкам [6]. К примеру, для первой карты признаков размера 38×38 для каждой точки генерируются 4 вида ограничивающих прямоугольников разной формы и определенного масштаба, зависящего от размера карты признаков. В результате для данной карты признаков всего генерируется 5776 рамок по умолчанию. Данные рамки играют важную роль в алгоритме SSD, так как те из них, которые лучше всего совпадают с реальным расположением объекта на изображении, используются для обучения регрессионной модели, предсказывающей положение ограничивающего прямоугольника для объекта [8]. Пример

расположения рамок по умолчанию на карте признаков размером 8×8 представлен на рисунке 2.4.

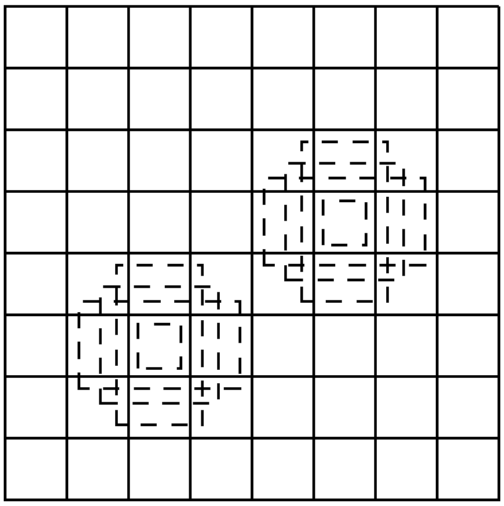

Рисунок 2.4 – Пример расположения рамок по умолчанию на карте признаков

При уменьшении размерности карт признаков, масштабы рамок по умолчанию по отношению к ним увеличивается. В целом, масштаб рамок по умолчанию увеличивается от 0,1 до 0,9 в процентном отношении к размерам карт признаков [6]. Каждый набор рамок по умолчанию центрирован относительно окна 3×3 и не выходит за его пределы, что видно из рисунка 2.4. Это необходимо для того, чтобы рамки по умолчанию совпадали с размером ядра сверточных слоев, которые производят классификацию и регрессию. Данный размер ядра свертки имеют все дополнительные группы сверточных слоев. Кроме того, для различных карт признаков генерируется свое количество рамок по умолчанию, располагающихся на одной точке карты признаков. К примеру, для карт признаков с шириной и высотой 19×19, 10×10 и 5×5 в одной точке центрированы 6 различных рамок по умолчанию вместо четырех, что повышает вероятность лучше охватить объект на изображении. Всего для всех карт признаков генерируется 8732 ограничивающие рамки.

Рамки по умолчанию для одной точки карты признаков имеют разную форму и соотношения высоты и ширины. К примеру, для генерации самого маленького равностороннего прямоугольника высота и ширина задаются равными масштабу рамок по умолчанию на данной карте признаков. Для генерации большого равностороннего прямоугольника в качестве высоты и ширины выступает произведение масштаба рамок текущей карты признаков и масштаба рамок следующей карты признаков. Для получения прямоугольников в качестве значения ширины используется произведение масштаба рамок по умолчанию на текущей карте признаков и квадратного корня из 2. Для расчета высоты происходит уже не произведение, а деление этих значений. Чтобы получить прямоугольник с такими же пропорциями, но вытянутым в другую сторону, необходимо поменять местами параметры высоты и ширины [6]. Таким образом генерируются 4 рамки по умолчанию разной формы и размера. Для генерации 6 рамок, можно повторить две предыдущие операции, используя квадратный корень из 3.

После того как рамки по умолчанию для всех карт признаков сгенерированы, необходимо понять какие из них лучше всего соответствуют правильным рамкам объекта на тренировочном изображении, чтобы обучить на них нейронную сеть. Для этого находится индекс пересечения между всеми рамками по умолчанию и действительными рамками объектов. Данный индекс называется коэффициентом Жаккара и рассчитывается по формуле (6).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \quad (6)$$

где $|A \cap B|$ – площадь пересечения;

$|A \cup B|$ – площадь объединения множеств A и B.

Значение индекса пересечения может изменяться от 0 до 1, где 0 означает, что рамки не пересекаются, а 1 означает, что рамки полностью совпадают. Те рамки по умолчанию, которые имеют индекс пересечения

больше 0,5 считаются положительным, а остальные помечаются как отрицательные [8].

После нахождения положительных и отрицательных рамок по умолчанию, карты признаков проходят через сверточные слои классификации и регрессии, которые предсказывают классы объектов на изображении и ограничивающие прямоугольники. Количества ядер сверток для слоев классификации и регрессии вычисляются по формуле (7).

$$\begin{aligned} size_{class} &= c \cdot b, \\ size_{reg} &= 4 \cdot b, \end{aligned} \tag{7}$$

где c – количество классов;

b – количество различных рамок по умолчанию, применяемых к конкретному слою.

Таким образом для каждой рамки по умолчанию предсказывается 4 смещения и вероятности по классам. Полученные с разных карт признаков предсказания позволяют обнаруживать на исходном изображении объекты, расположенные в разных местах, с разным размером [8].

Целью обучения архитектуры нейронной сети SSD является минимизация функции потерь, состоящей из ошибок локализации и классификации. Расчёт данной функции потерь представлен в формуле (8).

$$L(x, c, l, g) = \frac{1}{N} \cdot \Big(L_{conf}(x, c) + \alpha L_{loc}(x, l, g) \Big), \tag{8}$$

где N – количество рамок по умолчанию;

α – значение, необходимое для кросс-валидации и равное 1;

$L_{conf}(x, c)$ – ошибка классификации;

$L_{loc}(x, l, g)$ – ошибка локализации.

Кросс-валидацией называется техника оценивания работы модели на наборе данных, не входящим в обучающую выборку [25]. Для того, чтобы рассчитать общую ошибку по формуле (8), необходимо найти составляющие её ошибки классификации и локализации.

Для расчета ошибки локализации используется функция потерь, называемая «Smooth L1», являющаяся комбинацией среднего квадратичного отклонения и среднего абсолютного отклонения, представленных в формуле (1). Данная функция потерь представлена на формуле (9).

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 \text{ если } |x| \leq 1 \\ |x| - 0.5 \text{ иначе} \end{cases} \quad (9)$$

Расчет ошибки локализации с использованием данной функции потерь представлен в формуле (10).

$$\begin{gathered} L_{loc}(x, l, g) = \sum_i^N \sum_{m \in \{cx,cy,w,h\}} x_{ij}^k \cdot smooth_{L1}\left(l_i^m - \hat{g}_j^m\right), \\ \hat{g}_j^{cx} = \left(g_j^{cx} - d_i^{cx}\right) : d_i^w, \\ \hat{g}_j^{cy} = \left(g_j^{cy} - d_i^{cy}\right) : d_i^h, \\ \hat{g}_j^w = \lg\left(g_j^w : d_i^w\right), \\ \hat{g}_j^h = \lg\left(g_j^h : d_i^h\right), \end{gathered} \quad (10)$$

где x_{ij}^k – значение, определяющее соответствие между i-той положительной рамкой по умолчанию и j-той правильной ограничивающей рамкой объекта типа k;

m – переменная, определяющая для какого значения необходимо производить смещение (координаты центра якоря по x и по y, по ширине и по высоте соответственно);

l_i^m – значение m, предсказанное нейронной сетью для i-той положительной рамки по умолчанию;

$\hat{g}_j^m$ – смещение j-той правильной рамки объекта по значению m, относительно i-той рамки по умолчанию;

g_j^m – параметр m правильного ограниченного прямоугольника объекта;

d_i^m – параметр m i-той положительной рамки по умолчанию.

Как видно из формулы (10), ошибка локализации вычисляется только для положительных рамок по умолчанию. В свою очередь, для расчета ошибки классификации используются как положительные, так и отрицательные рамки по умолчанию, однако так как отрицательных намного

больше, берется только некоторое их количество, с наибольшими потерями уверенности в соотношении 3:1 с положительными рамками по умолчанию. Данный подход называется «hard negative mining» [25].

Ошибка классификации рассчитывается с использованием функции потерь «softmax loss», которая рассчитывает перекрестную энтропию между предсказаниями классификатора и реальным классом объекта. Перекрестная энтропия позволяет узнать, насколько точен выполненный прогноз модели. Чем меньше её значение, тем точнее предсказание к реальному классу. Расчет ошибки классификации представлен в формуле (11).

$$L_{conf}(x,c) = -\sum_{i\in pos}^{N} x_{ij}^{p} \cdot \log\left(\hat{c}_i^p\right) - \sum_{i\in neg} log\left(\hat{c}_i^0\right),$$
$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p log(c_i^p)}, \quad (11)$$

где *pos* – индекс положительных примеров;

neg – индекс отрицательных примеров;

x_{ij}^{p} – значение, определяющее соответствие между i-той положительной рамкой по умолчанию и j-той правильной ограничивающей рамкой объекта категории p (количество классов);

$\hat{c}_i^p$ – вероятность того, что i-тая положительная рамка по умолчанию содержит объект класса категории p (функция softmax);

$\hat{c}_i^0$ – вероятность того, что i-тая отрицательная рамка по умолчанию не содержит объектов (содержит объект класса «фон»);

c_i^p – предсказанная моделью уверенность в том, что объект внутри i-той положительной рамки по умолчанию принадлежит категории p.

После вычисления ошибок классификации и локализации вычисляется общая ошибка, представленная в формуле (8). Данная ошибка используется для оптимизации функции потерь по алгоритму стохастического градиентного спуска, который был описан в первом разделе. В процессе обучения нейронная сеть учится находить правильные смещения для рамок

по умолчанию, чтобы они ограничивали нужные объекты на изображении, а также правильно классифицировать данные объекты [8].

При использования обученной модели нейронной сети на изображении, может быть предсказано множество ограничивающий рамок для одних и тех же объектов, которые будут накладываться друг на друга. Для того чтобы удалить дублирующие ограничивающие рамки и оставить только лучшие, используется алгоритм под названием «non-maximum suppression» или подавление не-максимумов.

Полная схема распознавания объектов на изображении с помощью алгоритма SSD представлена на рисунке 2.5.

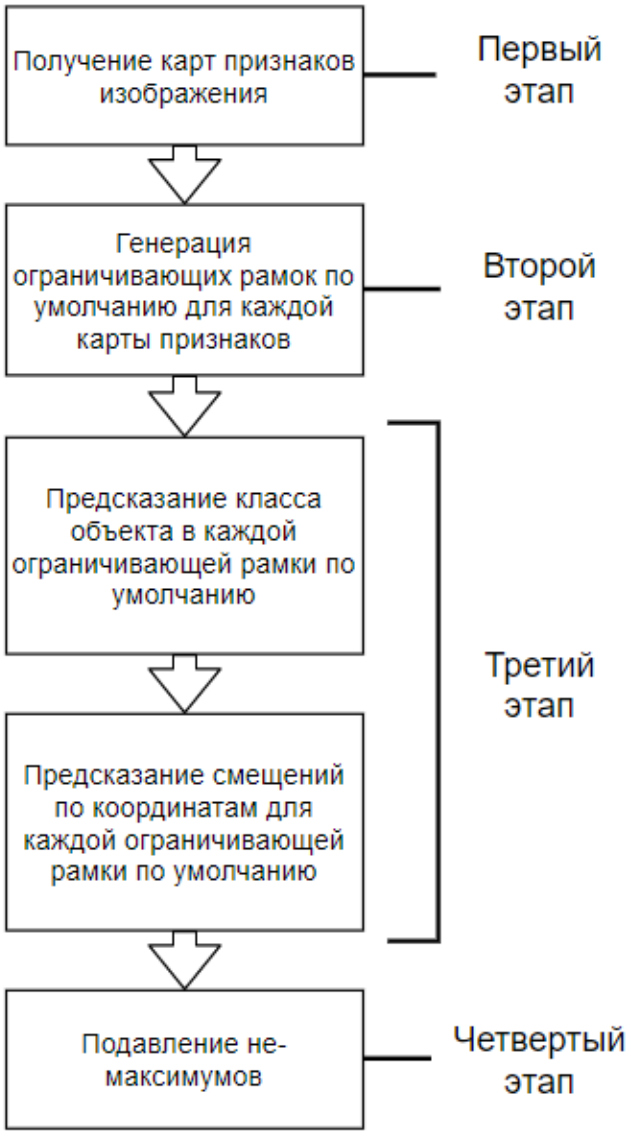

Рисунок 2.5 – Схема работы алгоритма SSD

Подход данного алгоритма состоит в следующем:

– из множества всех ограничивающих рамок выбирается только одна рамка T, с наибольшей уверенностью в том, что данный объект принадлежит к конкретному классу;

– данная ограничивающая рамка рисуется на изображении и удаляется из множества всех рамок;

– находится индекс пересечения рамки T со всеми остальными рамками, рассчитываемый по формуле (6);

– те рамки, у которыми индекс пересечения с T пересекает пороговое значение, удаляются из множества всех рамок.

Данные действия повторяются до тех пор, пока множество рамок не останется пустым. В результате этого, на выходном изображении будут присутствовать только самые лучшие ограничивающие прямоугольники с наибольшей уверенностью в том, что объект внутри ограничивающей рамки принадлежит к конкретному классу [8].

Таким образом, были основные принципы работы алгоритма распознавания объектов Single Shot MultiBox Detector и проанализированы особенности построения и обучения архитектуры сверточных нейронных сетей, входящих в его состав.

2.2 Реализация алгоритма SSD для локализации возгораний

Для реализации алгоритма SSD был выбран высокоуровневый кроссплатформенный объектно-ориентированный язык программирования Python. Данный язык программирования является мощным инструментом, так как содержит множество библиотек, реализующих программные решения для разработки различных систем, в частности для решения задач машинного обучения [5].

Для реализации и обучения нейронной сети, используемой в алгоритме SSD, была выбрана библиотека машинного обучения TensorFlow версии 2.0. Она включает в себя множество готовых решений и алгоритмов,

используемых при машинном обучении, таких как функции оптимизации, функции потерь, различные метрики, а также инструменты для проектирования различных нейронных сетей [12]. Для работы с данной библиотекой, все обрабатываемые ей данные представляются в виде специальных контейнеров, называемых тензорами – многомерными массивами. На данный момент TensorFlow является наиболее популярной библиотекой для машинного обучения.

В первую очередь при реализации алгоритма для распознавания возгораний на основе архитектуры SSD были реализованы сверточные слои, используемые в модели нейронной сети. Построение слоёв модели выполняется с помощью встроенного в Tensorflow высокоуровневого API для построения моделей под названием Keras.

Создание сверточных слоев происходит с помощью метода «Conv2D», который в качестве параметров принимает количество ядер свертки, их размерность, размер граничного заполнения, расширение и функцию активации, используемую после свертки. Для создания слоев подвыборки по максимальному значению используется метод «MaxPool2D», принимаемый аналогичные параметры, за исключением расширения и функции активации [24]. Для задачи входных данных нейронной сети используется метод Input, который создает входной тензор определенного размера. Данный тензор подается на вход метода «Model», который создает модель из нескольких слоев [4]. Помимо входного тензора, данной функции необходимо передать выходные данные последнего слоя.

Программный код реализации слоев сверточной сети VGG16 представлен на рисунке 2.6.

```
def create_vgg16_layers():
    vgg16_conv4 = [
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.MaxPool2D(2, 2, padding='same'),

        layers.Conv2D(128, 3, padding='same', activation='relu'),
        layers.Conv2D(128, 3, padding='same', activation='relu'),
        layers.MaxPool2D(2, 2, padding='same'),

        layers.Conv2D(256, 3, padding='same', activation='relu'),
        layers.Conv2D(256, 3, padding='same', activation='relu'),
        layers.Conv2D(256, 3, padding='same', activation='relu'),
        layers.MaxPool2D(2, 2, padding='same'),

        layers.Conv2D(512, 3, padding='same', activation='relu'),
        layers.Conv2D(512, 3, padding='same', activation='relu'),
        layers.Conv2D(512, 3, padding='same', activation='relu'),
        layers.MaxPool2D(2, 2, padding='same'),

        layers.Conv2D(512, 3, padding='same', activation='relu'),
        layers.Conv2D(512, 3, padding='same', activation='relu'),
        layers.Conv2D(512, 3, padding='same', activation='relu'),
    ]

    vgg16_conv7 = [
        layers.MaxPool2D(3, 1, padding='same'),
        layers.Conv2D(1024, 3, padding='same', dilation_rate=6, activation='relu'),
        layers.Conv2D(1024, 1, padding='same', activation='relu'),
    ]
```

Рисунок 2.6 - Программный код реализации слоев сверточной сети VGG16

Дополнительные слои, а также слои для классификации и регрессии реализованы таким же образом и будут приведены в приложении. Для того, чтобы использовать эти слои при прямом проходе, был создан класс для модели SSD. Данный класс содержит метод инициализации модели, в котором загружаются веса, настроенные в процессе долгого обучения на наборе данных ImageNet, содержащим порядка 15 миллионов размеченных изображений, разделенных на десятки тысяч классов. Данный подход позволяет ускорить процесс обучения рассматриваемой архитектуры для

распознавания новых классов объектов. Программный код функции, осуществляющей прямой проход в модели SSD представлен на рисунке 2.7.

```
def call(self, x):
    """ Прямой проход
    """
    confs = []
    locs = []
    head_idx = 0
    for i in range(len(self.vgg16_conv4.layers)):
        x = self.vgg16_conv4.get_layer(index=i)(x)
        if i == len(self.vgg16_conv4.layers) - 5:
            conf, loc = self.compute_heads(self.batch_norm(x), head_idx)
            confs.append(conf)
            locs.append(loc)
            head_idx += 1

    x = self.vgg16_conv7(x)

    conf, loc = self.compute_heads(x, head_idx)

    confs.append(conf)
    locs.append(loc)
    head_idx += 1

    for layer in self.extra_layers:
        x = layer(x)
        conf, loc = self.compute_heads(x, head_idx)
        confs.append(conf)
        locs.append(loc)
        head_idx += 1

    confs = tf.concat(confs, axis=1)
    locs = tf.concat(locs, axis=1)

    return confs, locs
```

Рисунок 2.7 - Программный код функции, осуществляющей прямой проход в модели SSD

Данная функция принимает на вход изображение и совершает прямой проход для предсказания ограничивающих рамок объектов на изображении и их классов.

Генерация наборов обучающий данных происходит с помощью функции «create_batch_generator», предоставляющей объекты класса генератора тренировочных и проверочных данных. Эти объекты-генераторы позволяют в процессе обучения получать наборы данных, состоящие из

изображений, меток классов для всех рамок по умолчанию, а также смещений для положительных рамок по умолчанию относительно реальных рамок объекта. Программный код данной функции представлен на рисунке 2.8.

```
def create_batch_generator(root_dir, year, default_boxes,
                           new_size, batch_size,
                           augmentation=None):
    voc = VOCDataset(root_dir, year, default_boxes,
                     new_size, augmentation)

    info = {
        'idx_to_name': voc.idx_to_name,
        'name_to_idx': voc.name_to_idx,
        'length': len(voc),
        'image_dir': voc.image_dir,
        'anno_dir': voc.anno_dir
    }

    train_gen = partial(voc.generate, subset='train')
    train_dataset = tf.data.Dataset.from_generator(
        train_gen, (tf.string, tf.float32, tf.int64, tf.float32))
    val_gen = partial(voc.generate, subset='val')
    val_dataset = tf.data.Dataset.from_generator(
        val_gen, (tf.string, tf.float32, tf.int64, tf.float32))

    train_dataset = train_dataset.shuffle(40).batch(batch_size)
    val_dataset = val_dataset.batch(batch_size)

    return train_dataset.take(-1), val_dataset.take(-1), info
```

Рисунок 2.8 – Программный код функции, создающий генераторы обучающих и проверочных данных

Метод генерации использует специальный класс «VOCDataset», который подготавливает данные для обучения: считывает из обучающего набора данных тренировочные и проверочные изображения, а также метки классов с ограниченными прямоугольниками, изменяет размеры изображения и находит смещения и метки классов для положительных рамок по умолчанию [7]. Программный код данного класса будет приведен в приложении. Создание генератора обучающих наборов данных происходит с помощью метода «Dataset.from_generator» библиотеки TensorFlow. Получаемые генераторы позволяют в случайном порядке перемешивать

обучающие данные и подавать их небольшими порциями, называемыми батчами, что необходимо при стохастическом градиентном спуске.

Расчет всех ошибок для алгоритма SSD происходит в главной функции класса «SSDLoss», программный код которой представлен на рисунке 2.9.

```
def __call__(self, confs, locs, gt_confs, gt_locs):

    cross_entropy = tf.keras.losses.SparseCategoricalCrossentropy(
        from_logits=True, reduction='none')

    # подсчет ошибок классификации без разделения
    temp_loss = cross_entropy(gt_confs, confs)
    pos_idx, neg_idx = hard_negative_mining(temp_loss, gt_confs, self.neg_ratio)

    # ошибки классификации для положительных и отрицательных примеров

    cross_entropy = tf.keras.losses.SparseCategoricalCrossentropy(
        from_logits=True, reduction='sum')
    smooth_l1_loss = tf.keras.losses.Huber(reduction='sum')

    conf_loss = cross_entropy(
        gt_confs[tf.math.logical_or(pos_idx, neg_idx)],
        confs[tf.math.logical_or(pos_idx, neg_idx)])

    # ошибка регресси для положительных примеров
    loc_loss = smooth_l1_loss(
        gt_locs[pos_idx],
        locs[pos_idx])

    num_pos = tf.reduce_sum(tf.dtypes.cast(pos_idx, tf.float32))

    conf_loss = conf_loss / num_pos
    loc_loss = loc_loss / num_pos

    return conf_loss, loc_loss
```

Рисунок 2.9 – Программный код функции расчета ошибок для алгоритма SSD

Для того, чтобы рассчитать ошибку классификации используется функция потери «SparseCategoricalCrossentropy» библиотеки TensorFlow, которая рассчитывает перекрестную энтропию между предсказаниями классификатора и реальным классом объекта [13]. В начале рассчитывается ошибка классификации для каждой рамки по умолчанию, после чего с использованием алгоритма «hard negative mining» все рамки по умолчанию делятся на положительные и отрицательные, с учетом индекса перекрытия и результата перекрестной энтропии. Положительные и отрицательные рамки

по умолчанию используются для повторного расчета ошибки классификации с той же функцией потери. Для расчета ошибки локализации на основе положительных рамок используется функции потерь «Smooth L1», реализуемая методом «Huber» библиотеки TensorFlow [24].

Для одного шага обучения модели SSD была реализована функция «train_step», программный код которой представлен на рисунке 2.10.

```
def train_step(imgs, gt_confs, gt_locs, ssd, criterion, optimizer):
    with tf.GradientTape() as tape:
        confs, locs = ssd(imgs)

        conf_loss, loc_loss = criterion(
            confs, locs, gt_confs, gt_locs)

        loss = conf_loss + loc_loss
        l2_loss = [tf.nn.l2_loss(t) for t in ssd.trainable_variables]
        l2_loss = args.weight_decay * tf.math.reduce_sum(l2_loss)
        loss += l2_loss

    gradients = tape.gradient(loss, ssd.trainable_variables)
    optimizer.apply_gradients(zip(gradients, ssd.trainable_variables))

    return loss, conf_loss, loc_loss, l2_loss
```

Рисунок 2.10 – Программный код функции, реализующей шаг обучения

Для автоматического дифференцирования обучаемых переменных при обратном распространении ошибки, необходимо использовать специальный диспетчер контекста «tf.GradientTape()». Внутри данного контекста используется функция для расчёта ошибок модели SSD. Получаемые с помощью неё ошибки классификации и локализации суммируются, образуя общую ошибку детектора для одного обучающего примера [4]. После этого находится сумма квадратов ошибок для всех примеров обучающего пакета и делится на их количество. Общая ошибка используется для нахождения градиента, который используется методом оптимизации для уменьшения ошибки обучаемой модели. В качестве оптимизатора используется стохастический градиентный спуск, который задается с помощью функции

библиотеки TensorFlow «SDG». Создание оптимизатора представлено на рисунке 2.11.

```
optimizer = tf.keras.optimizers.SGD(
    learning_rate=lr_fn,
    momentum=args.momentum)
```

Рисунок 2.11 – Создание оптимизатора стохастический градиентный спуск

В качестве параметров метода оптимизации передается значение скорости обучения и значения импульса, который используется для учета уровня влияния градиентов, рассчитанных на предыдущих шагах обучения. Данный параметр позволяет обновлять веса более сглажено, что способствует более быстрой оптимизации ошибки модели. Рекомендуется использовать импульс равный 0,9 [11].

Итак, были рассмотрены основные методы и классы, используемые для реализации алгоритма SSD. Весь программный код алгоритма будет приведен в приложении. Для распознавания объектов на изображении был выбран алгоритм Single Shot MultiBox Detector. В данном разделе были описаны основные принципы его работы, кроме того, данный алгоритм был реализован для локализации возгораний с использованием библиотеки машинного обучения TensorFlow.

3 Обучение и оценка точности реализованного алгоритма для распознавания возгораний

3.1 Создание набора изображений для обучения нейронной сети

Для того, чтобы обучить реализованную модель нейронной сети распознавать возгорания, необходимо подготовить обучающий набор данных, на основании которого будут настраиваться весовые коэффициенты. Обучающий набор данных должен состоять из тренировочных и проверочных данных, первые из которых используются в процессе обучения для корректирования параметров модели, а вторые для проверки её точности [9].

Для распознавания возгораний, в качестве классов объектов, требующих обнаружения, были выбраны «огонь» и «дым». Поскольку в открытом доступе сети интернет отсутствуют обучающие наборы данных, включающие изображения этих объектов, то были в вручную собраны 2000 изображений, содержащих огонь и дым. Для того, чтобы после обучения нейронная сеть определяла огонь и дым на изображениях как можно лучше, данные классы объектов в собранном наборе содержаться в разных ракурсах, при различном масштабе и времени суток. Все собранные изображения были размечены с помощью специальной программы LabelImg, которая позволяет создавать аннотации для изображений в формате Pascal VOC. Получаемые аннотации содержат необходимые для обучения данные, а именно список объектов на изображении, их классы и координаты ограничивающих рамок. Большая часть изображений, содержащих огонь и дым, а именно 1500 были добавлены в тренировочный набор данных, а оставшиеся 500 изображений в проверочный. Примеры изображений, содержащихся в обучающем наборе данных представлены на рисунке 3.1.

Рисунок 3.1 – Пример изображений из обучающего набора данных

Таким образом, были собраны и отмечены данные, необходимые для обучения нейронной сети, входящей в состав алгоритма SSD.

3.2 Обучение модели для распознавания возгораний на изображении

В процессе обучения используется несколько параметров, такие как эпоха, батч и итерация. Эпохой называется процесс прохождения всего тренировочного набора данных через нейронную сеть, другими словами, эпоха заканчивается тогда, когда не остается ни одного тренировочного

экземпляра, которого не видела обучаемая модель. Батчем или пакетом обычно называют небольшой набор примеров тренировочных данных, на основании которых вносятся коррективы в обучаемую модель. Итерация обозначает то, сколько таких пакетов должно пройти через модель, чтобы закончилась одна эпоха [16]. Останавливать обучение модели нейронной сети можно тогда, когда достигнута требуемая точность, либо если ошибка нейронной сети перестала уменьшаться.

Модель считается хорошо обученной только тогда, когда с одинаковой точностью определяет объекты как на данных, используемых в процессе обучения, так и на данных, не входящих в обучающую выборку. Если модель нейронной сети производит хорошие прогнозы только для обучающих данных, то это свидетельствует о переобучении.

Переобучение нейронной сети является одной из главных проблем глубоких нейронных сетей и говорит о том, что модель начинает запоминать тренировочные примеры, теряя способность к обобщению [20]. Переобучение может происходить по разным причинам:

– не достаточно примеров обучающих данных для того, чтобы нейронная сеть нашла между ними взаимосвязь;

– слишком много корректируемых параметров;

– плохая архитектура нейронной сети.

Решить проблему переобучения нейронной сети можно с помощью ранней остановки, для этого выбираются сохраненные весовые коэффициенты модели до того, как она начала переобучаться и используют их для совершения прогнозов, точность которых может быть достаточно высокой. Кроме того, во избежание переобучения используется метод под названием «dropout», в котором с определенной вероятностью отключаются различные нейроны. Данный подход позволяет исключить вклад отключенных нейронов в общий результат работы модели и изменить процесс обучения, что может поспособствовать лучшему обучению модели [21].

Обучение модели, входящей в состав SSD, происходило с использованием сервиса Google Colab Notebook, который представляет из себя командную оболочку для работы на языке Python и предоставляет бесплатный доступ к графическим процессорам.

Результаты обучения нейронной сети на десяти эпохах представлены на рисунке 3.2.

```
Epoch: 1 Time: 2e+02s | Train loss: 5.2744 Train conf: 3.5243 Train loc: 0.6726 | Val loss: 3.2448 Val conf: 2.5898 Val loc: 0.6550
Epoch: 2 Time: 2e+02s | Train loss: 4.2240 Train conf: 2.5017 Train loc: 0.6467 | Val loss: 3.0579 Val conf: 2.4188 Val loc: 0.6392
Epoch: 3 Time: 2e+02s | Train loss: 4.0390 Train conf: 2.3309 Train loc: 0.6347 | Val loss: 2.8330 Val conf: 2.2031 Val loc: 0.6300
Epoch: 4 Time: 2e+02s | Train loss: 3.9137 Train conf: 2.2171 Train loc: 0.6253 | Val loss: 2.8345 Val conf: 2.2152 Val loc: 0.6193
Epoch: 5 Time: 2e+02s | Train loss: 3.8785 Train conf: 2.1922 Train loc: 0.6171 | Val loss: 2.7481 Val conf: 2.1429 Val loc: 0.6053
Epoch: 6 Time: 2e+02s | Train loss: 3.8071 Train conf: 2.1389 Train loc: 0.6012 | Val loss: 2.6774 Val conf: 2.0873 Val loc: 0.5901
Epoch: 7 Time: 2e+02s | Train loss: 3.7719 Train conf: 2.1203 Train loc: 0.5866 | Val loss: 2.6505 Val conf: 2.0809 Val loc: 0.5695
Epoch: 8 Time: 2e+02s | Train loss: 3.7283 Train conf: 2.0969 Train loc: 0.5684 | Val loss: 2.6397 Val conf: 2.0814 Val loc: 0.5583
Epoch: 9 Time: 2e+02s | Train loss: 3.7216 Train conf: 2.1083 Train loc: 0.5524 | Val loss: 2.8003 Val conf: 2.2532 Val loc: 0.5471
Epoch: 10 Time: 2e+02s | Train loss: 3.6329 Train conf: 2.0317 Train loc: 0.5414 | Val loss: 2.5064 Val conf: 1.9792 Val loc: 0.5272
```

Рисунок 3.2 – Результат обучения нейронной сети на 10 эпохах

Как можно увидеть из рисунка 3.2, общая ошибка распознавания возгораний на тренировочных и проверочных данных постепенно уменьшается. Изменение общей ошибки обучаемой нейронный сети за 10 эпох представлено на рисунке 3.3.

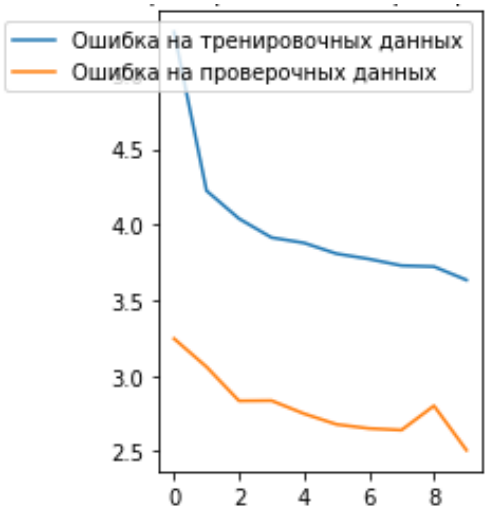

Рисунок 3.3 - Изменение ошибки нейронный сети за 10 эпох

Поскольку подготовленный для обучения набор данных является не слишком большим, то обучение на нем длилось в течении 300 эпох, пока изменение общей ошибки стало незначительным. Минимальная средняя

ошибка на тренировочных и проверочных данных достигла 0,4 и 0,5 соответственно, что свидетельствует о том, что модель не начала переобучаться.

Таким образом, была обучена модель для классификации и локализации объектов на изображении.

3.3 Оценка точности обученной модели для распознавания возгораний на изображении

Оценка точности алгоритма детектирования объектов SSD производится с использованием метрики mAP (Mean average precision), которая высчитывает общее среднее значение точности модели на основании средней точности детектирования объектов каждого класса.

Основная идея этого метода оценки в точности состоит в том, что для каждого класса объекта между всеми правильными ограничивающими рамками объектов и рамками, предсказанными моделью, находится индекс перекрытия, рассчитываемый по формуле (6). В первую очередь с правильными рамками объектов сравниваются те предсказания, которые имеют наибольшую оценку уверенности классификации. Если индекс перекрытия предсказанной рамки и правильной рамки составляет больше 0,5, то такая предсказанная рамка считается истинно-позитивной, иначе истинно-негативной. Если одна из предсказанных ранее рамок уже определялась для конкретной правильной рамки объекта как истинно-позитивная, то следующие предсказания, которые сравниваются с этой же правильной рамкой объекта будут являться ложно-позитивными вне зависимости от значения индекса пересечения. После получения истинно-позитивных и истинно-негативных рамок объекта рассчитываются значения точности (precision) и отзыва (recall) первая из которых показывает точность предсказаний, а вторая то, насколько много совершено положительных предсказаний [8]. Данные показатели рассчитываются по формуле (12).

$$Precision = TP : (TP + FP), \qquad (12)$$

$$\text{Recall} = TP{:}TB,$$

где TP – истинно-позитивные предсказания;

FP – ложно-позитивные предсказания;

TB – общее количество правильных рамок для всех объектов.

Показатели точности и отзыва считаются для каждого изображения и выступая в роли значений на осях координат, образуют кривую. Для того, чтобы получить среднее значение точности детектирования объекта по классу, необходимо сгладить данную кривую по максимальному значению справа и вычислить под ней площадь.

Программный код функции, вычисляющей среднюю точность по классам и общую среднюю точность модели, будет приведен в приложении. Результат оценки точности обученной модели на проверочных данных приведен на рисунке 3.4.

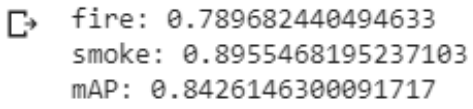

Рисунок 3.4 – Результат оценки точности детектора по метрике mAP

Как видно на изображении 3.4, точности обученной модели нейронной сети алгоритма SSD на проверочных данных составляет 80% и 90% в определении огня и дыма соответственно. Общая точность детектора на тренировочных данных составила 84%, что является довольно хорошим результатом, учитывая то, что огонь и дым не имеют постоянной формы и сложны для распознавания. Результаты работы детектора приведены на изображениях 3.5 – 3.8.

Рисунок 3.5 – Результаты работы детектора №1

Рисунок 3.6 – Результаты работы детектора №2

Рисунок 3.7 – Результаты работы детектора №3

Рисунок 3.8 – Результаты работы детектора №4

По результатам работы детектора видно, что локализация и классификация таких классов, как «дым» и «огонь» позволяют намного лучше находить пожары на изображении, так как отсутствие объектов одного из классов по результатам детектирования, не исключают присутствия возгораний.

Таким образом, с помощью метрики mAP была произведена оценка точности распознавания огня и дыма на изображениях. Общая средняя точность распознавания обученной модели по данным классам составила 84%.

Итак, в данном разделе был описан процесс создания обучающего набора данных, с помощью которых было произведено обучение модели нейронной сети, входящей в состав алгоритма SSD. В заключение данной работы, с помощью метрики mAP была произведена оценка точности распознавания возгораний реализованным алгоритмом.

Заключение

В данной выпускной квалификационной работе исследовался вопрос локализации возгорания на изображении. Для этого было необходимо выбрать и реализовать алгоритм на основе сверточных нейронных сетей, способный распознавать такие классы объектов, как дым и огонь.

Был проведен анализ существующих методов распознавания объектов на изображении и сделан вывод, что для локализации возгораний наиболее подходящими являются методы, основанные на машинном обучении, а именно сверточные нейронные сети. Были рассмотрены основные принципы работы искусственных нейронных сетей, особенности их построения, а также методы, с помощью которых происходит их обучение. Кроме того, был подробно разобран процесс применения сверточных нейронных сетей для классификации объекта на изображении.

Для локализации и классификации объектов на изображении было решено использовать алгоритм Single Shot MultiBox Detector, в основу которого входят сверточные нейронные сети. Данный алгоритм был выбран поскольку он обнаруживает объекты за один прямой проход нейронной сети и достигает наилучшего баланс между точностью и быстродействием, что является важным при распознавании возгораний в режиме реального времени. Также были описаны основные принципы работы алгоритма SSD и произведена его реализация для локализации возгораний на изображении на языке Python с помощью инструментов библиотеки машинного обучения TensorFlow.

Для обучения модели сверточной нейронной сети, входящей в состав алгоритма SSD был вручную собран и размечен обучающий набор данных, состоящий из 2000 изображений огня и дыма. Обучение сверточной нейронной сети проходило в течении 300 эпох, в результате чего минимальная средняя ошибка предсказания составила 0,4. Для оценки точности распознавания возгораний на изображении использовалась метрика

mAP которая показала, что общая средняя точность нахождения огня и дыма на изображении составила 84%.

Таким образом, был реализован алгоритм на основе сверточных нейронных сетей, способный обнаруживать на изображении такие классы объектов, как дым и огонь с довольно высокой точностью. Тем не менее, данная точность распознавания в дальнейшем может быть улучшена путем увеличения обучающего набора изображений и изменения таких параметров обучения, как размер батча или количество эпох. Использование данного алгоритма для анализа видеопотоков, получаемых с беспилотных транспортных средств, может позволить обнаруживать возгорания в городах, лесах, заповедниках или другой местности без участия человека, а также поспособствует ликвидации пожаров на раннем этапе.

Список используемых источников

1. Бурков А. Машинное обучение без лишних слов / Андрей Бурков - Питер СПб, 2020. – 192 с.
2. Вьюгин В. Математические основы машинного обучения и прогнозирования / Владимир Вьюгин. - МЦНМО, 2014. - 304 с.
3. Гелиг А., Матвеев А. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. Учебное пособие / Аркадий Гелиг, Алексей Матвеев - Издательство СПбГУ, 2014. – 224 с.
4. Документация по библиотеке машинного обучения TensorFlow 2.0 // TensorFlow [Электронный ресурс]: официальный сайт TensorFlow. URL: https://www.tensorflow.org/versions/r2.0/api_docs/python/ (дата обращения 17.03.2020).
5. Документация по языку программирования Python // Python [Электронный ресурс]: официальный сайт Python. URL: https://www.python.org/doc/ (дата обращения 12.03.2020).
6. Christian Szegedy. Scalable, High-Quality Object Detection, 2015 // arXiv [Электронный ресурс]: открытый архив научных статей. URL: https://arxiv.org/abs/1412.1441 (дата обращения 10.04.2020).
7. Ryo Takahashi. Data Augmentation using Random Image Cropping and Patching for Deep CNNs, 2019 // arXiv [Электронный ресурс]: открытый архив научных статей. URL: https://arxiv.org/abs/1811.09030 (дата обращения 10.04.2020).
8. Wei Liu. SSD: Single Shot MultiBox Detector, 2016 // arXiv [Электронный ресурс]: открытый архив научных статей. URL: https://arxiv.org/abs/1512.02325 (дата обращения 10.04.2020).
9. Abu-Mostafa Y. Learning From Data / Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin – AMLBook. – 2012.-Jan. -C. 213.
10. Bishop C. Pattern Recognition and Machine Learning (Information Science and Statistics) / Christopher M. Bishop - Springer-Verlag New York Inc. - 2007.-Feb. -C. 738.

11. Chollet F. Deep Learning with Python / Francois Chollet - Manning Pubns Co. – 2017.-Nov. -C. 361.

12. Eddison L. Python Machine Learning: A Technical Approach To Python Machine Learning For Beginners / Leonard Eddison - CreateSpace Independent Publishing Platform. – 2018.-Mar. -C. 292.

13. Geron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow / Aurelien Geron – O`Reilly Media. – 2017.-Mar. -C. 566.

14. Goodfellow I. Deep Learning (Adaptive Computation and Machine Learning series) / Ian Goodfellow, Yoshua Bengio, Aaron Courville - The MIT Press. -2016.-Nov. -C. 800.

15. Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists / Sarah Guido, August Mueller - O`Reilly Media. – 2016.-May. -C. 400.

16. Harrington P. Machine Learning in Action / Peter Harrington - Manning Publications. – 2012.-April. -C. 384.

17. Kelleher J. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies / John D. Kelleher, Brian Mac Namee, Aoife D`Arcy - The MIT Press. – 2015.-July. – C. 624.

18. Mitchell T. Machine Learning / Tom Mitchell – Mc Graw Hill India. - 2017.– Mar. – C. 432.

19. Raschka S. Python Machine Learning / Sebastian Raschka - Packt Publishing. – 2015.-Sep. -C. 454.

20. Rashid T. Make Your Own Neural Network / Tariq Rashid - CreateSpace Independent Publishing Platform. – 2016. – C. 222.

21. Rojas R. Neural Networks: A Systematic Introduction / Raul Rojas, Peter Varga - Springer Berlin Heidelberg. – 1996.-Jul. -C. 522.

22. Russell S. Artificial Intelligence: Pearson New International Edition: A Modern Approach / Stuart Russel, Norvig Peter – Pearson. – 2013.-Aug. -C. 1104.

23. Shalev-Shwarthz S. Understanding Machine Learning: From Theory to Algorithms / Shai Shalev-Shwartz, Shai Ben-David - Cambridge University Press. -2014.-May. -C. 415.

24. Shukla N. Machine Learning with TensorFlow / Nishant Shukla – Manning Publication. – 2018.-Jan. -C. 272.

25. Witten I. Data Mining: Practical Machine Learning Tools and Techniques / Ian H. Witten, Eibe Frank, Mark A. Hall - Morgan Kaufmann. – 2011.-Jan. -C. 664.

Приложение А

Программный код реализованного алгоритма

Программный код реализованного алгоритма находится на диске, приложенном к ВКР.