Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра Математического и компьютерного моделирования
Направление подготовки Математика и компьютерные науки

РАБОТА ПРОВЕРЕНА
Рецензент, _____

_____

_____

_____

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой

д.ф.-м.н.доцент

_____/С.А. Загребина

_____

Применение сверточных нейронных сетей для обнаружения объектов на изображение

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ-02.03.01.2018.108.ПЗ ВКР

Руководитель работы доцент кафедры
к.ф.-м.н., доцент
_____/М.А. Сагадеева
«____»_____ 2018 г.

Автор работы студент группы ET - 411

_____/В.В. Марков
«____»_____ 2018 г.

Нормоконтролер, доцент кафедры,
к.ф.-м.н.
_____/А.А. Акимова
«____»_____ 2018 г.

Челябинск, 2018

АННОТАЦИЯ

Задача обнаружения объекта является одной из классических задач компьютерного зрения. Решение данной задачи используется в различных сферах жизни человека и находят широкое применение в бизнесе.

В данной работе будет рассмотрено решение задачи обнаружения ценника на изображение. Первая глава посвящена постановке задачи машинного обучения, оценки качества модели, а так же дан обзор основным алгоритмам обнаружения объекта. Во второй главе представлена постановка задачи от предприятия, алгоритм решения и результаты его работы.

Оглавление

ВВЕДЕНИЕ

Компьюторное зрение является дисциплинной, которая включает в себя знания из различных областей науки, таких как биология, компьютерные науки, математика, инженения и физика.

История компьютерного зрения начинается с исследований в нейробиологии, проводимые в 1959 году, и были связанны с учеными Давид Хубель и Торстен Визель. Они совместно установили большинство основных фактов, касающихся работы зрительной системы млекопитающих. Одним из основных результатом их работы является открытие, что нейроны первичных зрительных центров сильнее реагируют на специфические паттерны. Их открытие оказало огромное влияние на современные модели компьютерного зрения [5].

Еще одним важным шагом в области компьютерного зрения стала докторская работа Larry Roberts, в которой обсуждалась возможность извлечения информации о трехмерных объектах с помощью их двумерной проекции. Долгое время исследователи следовали этой работе и изучали компьютерное зрение в контексте , так называемых, блоков.

Позже, пришло понимание, что нужно заниматься образами из реального мира. В связи с этим, было много исследований, посвященных низкоуровневым задачам, такие как нахождения границ объекта и сегментация изображения.

Важным моментом стала концепция исследователя David Marr, которая заключается в восходящем подходе к пониманию изображения. Суть концепции заключается в том, что сначала применяются низкоуровневые алгоритмы для получения первичного эскиза изображения. Далее, используются алгоритмы для обработки эскиза и выделения объектов. В конце, применяется уже структурный анализ и априорные знания об объекте для его распознавания и получении трехмерной модели. Данная работа считается одной из самой влиятельной в области компьютерного зрения. Однако, в по-

следствии, были выявлены некоторые ограничения данной концепции. Дело в том, что концепция чрезвычайно сложна, а для большинства прикладных задач компьютерного зрения нет необходимости в получении трехмерной модели объекта. Например, в задаче автономной навигации транспортного средства важно знать расстояние до впереди стоящего объекта, при этом совсем необязательно выделять его трехмерную модель [17].

Другой группой ученых проводились работы над идеей преставления сложных объектов с помощью более простых примитивов. Например, нахождение скелета человека, то есть нахождение представления человека в виде графа, где вершины отвечают за суставы, а ребра за кости [10].

В 80-е года 20 века исследователь David Lowe показал другой пример того, как можно реконструировать и распознавать объект на изображении с помощью его структуры. В его работе была попытка распознать объекты используя лишь такие примитивы как линии, границы и их комбинации.

Исследователи 60–80-х годов пытались решить сложную проблему в распознавании образов. Основным направлением их исследований заключается в попытках повторить механизмы зрительной системы человека. Но из за сложности задачи распознавания объекта на изображении, данная задача так и не была решена. Вместо нее, исследователи сосредоточились над проблемой сегментации изображения, которая казалась более легкой, по сравнению с распознаванием объекта. Задача сегментации изображения заключается в том, чтобы сгруппировать пиксели, принадлежащие к одному объекту, в один кластер.

Начиная с 2000 года в области компьютерного зрения начали активно применять алгоритмы машинного обучения, такие как svm, boosting, graphical models, а так же нейронные сети. Одно из успешных применений таких алгоритмов было применение AdaBoost для задачи нахождения лиц на изображении [14].

Стоит так же упомянуть алгоритм распознавания SIFT, основанный на

поиске и сопоставлении особых точек объекта. При этом, данный алгоритм устойчив к поворотам и масштабированию объекта [6].

Еще один классический алгоритм компьютерного зрения был разработан в 2005 году - Histogram of Gradients. Основная идея которого, заключается в представлении изображения в виде спектра градиентов [11].

Для развития области компьютерного зрения, начиная с 2006 года начали проводить соревнование PASCAL Visual Object Challenge. Целью соревнования было стимулировать развития области, придумывая все более точные алгоритмы. Суть соревнования заключалась в решении задачи распознавания и нахождении объекта на изображении.

В 2010 году было впервые проведено соревнование ImageNet, которое в свою очередь является более улучшенной версия PASCAL, так как имеет больший объем размеченного датасета.

В 2012 году на соревновании ImageNet победило решение, основанное на сверточных нейронных сетях. Это решение показало качество, заметно лучше, чем предыдущие алгоритмы классификации и нахождении объекта на изображении.

Глубокие сверточные сети вдохнули новую жизнь в область компьютерного зрения. Так, классические алгоритмы, обладают рядом недостатков. Одним из таких является то, что с увеличением выборки было очень тяжело, порой невозможно, улучшить качество работы алгоритма. Глубокие нейронные сети же, наоборот, с увеличением объема выборки способны улучшать качество работы.

Цель данной работы - решить задачу обнаружения объекта на изображении применительно к задаче поиска ценника на фотографии. Решить задачу обнаружения объекта значит локализовать объекта на изображении, т.е. в общем случае, найти минимальный бокс, описывающий объект, а так же классифицировать что за объект находится внутри бокса. На рисунке 1 показан пример результата работы алгоритма обнаружения объекта.

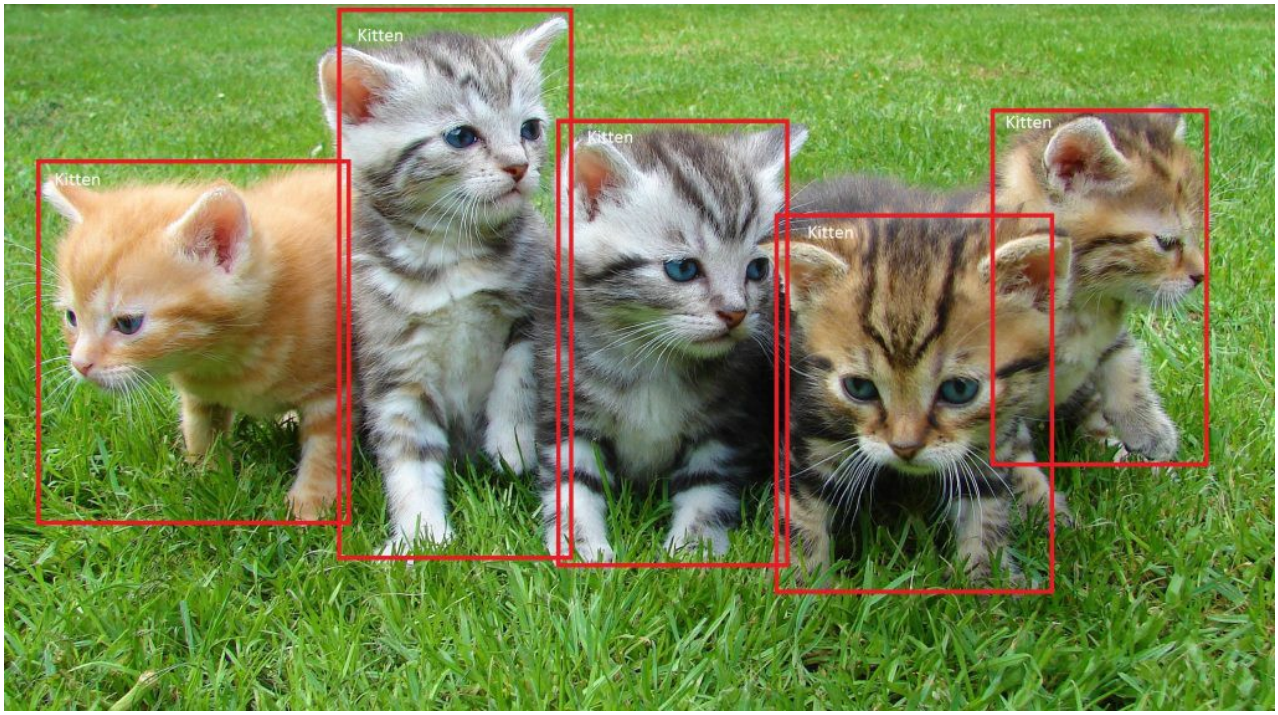

Рис. 1: Пример результата работы алгоритма обнаружения объекта

В связи с этим, при написании выпускной квалификационной работы были поставлены следующие задачи.

1. Провести обзор основных алгоритмов компьютерного зрения для решения задачи обнаружения объекта.
2. Реализовать базовое решение.
3. Провести анализ полученных результатов.

1. ЗАДАЧА И МЕТОДЫ РАСПОЗНАВАНИЯ

1.1. Задача классификации в машинном обучении

Пусть имеется множество описаний объекта $X = \{x_i\}_{i=1}^{l}$ и конечное множество меток классов $Y = \{y_i^m\}_{i=1}^{l}$, где m - количество классов. Имеется неизвестная функция $y* : X \to Y$, значения которой известны только на конечной обучающей выборки $X^l = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$. Необходимо построить алгоритм $a : X \to Y$ способный классифицировать произвольный объект $x \in X$.

Пусть $F_i : X \to \mathbb{R}$, F_i - функция, которая i - ому признаку объекта x ставит в соответствие число.

Таким образом, требуется построить такую поверхность в признаковом пространстве, которая разделяла бы объекты разных классов. Согласно теореме Колмогорова[2] о представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения при любом $n >= 2$ существуют такие определенные на единичном отрезке $E^1[0;1]$ непрерывные действительные функции $\psi^{pq}(x)$, что каждая определенная на n-мерном единичном кубе $E^n[0;1]$ непрерывная действительная функция $g(x_1, x_2, ..., x_n)$ представима в виде

$$g(x_1, x_2, ..., x_n) = \sum_{q=1}^{2n+1} \chi[\sum_{p=1}^{n} \psi^{pq}(x_p)]. \quad (1)$$

Из теоремы Хехт-Нильсена следует представимость любой многомерной функции нескольких переменных с помощью нейронной сети фиксированной размерности. Неизвестными остаются следующие характеристики функций активации нейронов.

1. Ограничения области значений (координаты асимптот) сигмоидальных функций активации нейронов скрытого слоя.

2. Наклон сигмоидальных функций активации.

3. Вид функций активации нейронов выходного слоя.

Про функции активации нейронов выходного слоя из теоремы Хехт-Нильсена известно только то, что они представляют собой нелинейные функции общего вида В одной из работ, продолжающих развитие теории, связанной с рассматриваемой теоремой, доказывается, что функции активации нейронов выходного слоя должны быть монотонно возрастающими. Это утверждение в некоторой степени сужает класс функций, которые могут использоваться при реализации отображения с помощью двухслойной нейронной сети [3].

Будем решать задачу апроксимации многомерной разделяющей поверхности суперпозицией нелинейных функций. Другими словами, в случае двухслойной нейронной сети, будем искать классификатор в виде:

$$a^m(x, w) = \sigma_m(\sum_{h=0}^{H} w_{hm}\sigma_h(\sum_{j=0}^{n} w_{jh}F_j(x_i))), \tag{2}$$

где σ - монотонно неубывающая, непрерывная, нелинейная функция, называемая функцией активации, w - вектор весов.

Популярные функции активации представлены ниже.

1. $sigmoid(x) = \frac{1}{1+e^{-x}}$.
2. $reLU(x) = max(0, x)$.
3. $SoftPlus(x) = log(1 + e^x)$.
4. $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

На рисунке 2 представлены графики популярных функций активации.

Ответы классификатора будут отличаться в зависимости от функции активации на финальном слое. В случае softmax функции, область значений функции активации будет [0, 1]. Данная функция обладает вероятностной

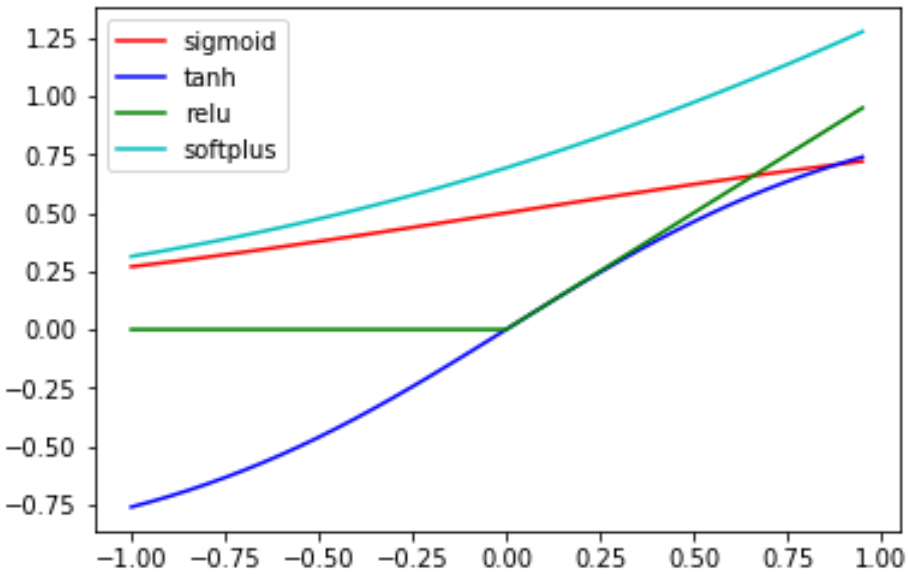

Рис. 2: Функции активации

интерпретацией: значение функции равно вероятности принадлежности объекта к классу. Мы можем выставить порог p, при котором будем говорить, что объект x относится к j - ому классу, если $a^j(x, w) > p$.

Можно показать, что минимизируя данный функционал, будет максимизироваться принцип максимума правдоподобия:

$$Q = -\frac{1}{n}\sum_{i=1}^{L}[y_i \ln a(w, x_i) + (1 - y_i)\ln(1 - a(w, x_i))], \tag{3}$$

$$w = \arg\min_w Q. \tag{4}$$

Данный функционал является непрерывным. Искать решение будем численно с помощью метода стохастического градиентного спуска.

Алгоритм стохастического градиентного спуска является одним из основных алгоритмов оптимизации в машинном обучение. На каждом шаге алгоритма происходит подсчет градиента минимизируемого функционала и переход в точку в направлении антиградиента. Таким образом, происходит поиск локального безусловного минимума. В то же время, существуют алгоритмы, использующие вторые частные производные. Они оказываются более точными в выборе направления локального минимума, но в то же время,

более медленными в подсчетах.

Алгоритм стохастического градиентного спуска состоит в следующем.

1. Вход: X^l - обучающая выборка, ν - длина градиентного шага, ϵ - допустимая погрешность.
2. Выход: w.
3. Задать начальное значение вектора w. Подсчитать значение функционала качества.
4. Повторять.
5. Выбрать объект x_i из обучающей выборки X^l.
6. Вычислить частные производные функционала по вектору w на объекте x_i.
7. Обновить веса согласно формуле $w^{k+1} = w^k - \nu\nabla_w Q$.
8. Повторять, пока $\triangle w > \epsilon$.

Для подсчета частных производных функционала качества Q по весам w удобнее всего представить нейронную сеть в виде так называемого графа вычислений. Граф вычислений - это ациклический направленный граф $G = (V, E)$, вершинами которого являются функции $g \in V$, причем часть вершин не имеет входных ребер, а одна вершина не имеет исходящих ребер. Ребра показывают зависимости между функциями, стоящих в узлах. Для того, чтобы получить значение результирующей функции, стоящей в конце графа, достаточно пройти по ребрам и вычислять каждую функцию в топологическом порядке.

А чтобы узнать частные производные этой функции, достаточно двигаться в обратном порядке. Если нужно найти частные производные функции $f \in V$, тогда в полном соответствие с формулой нахождения произ-

водной сложной функции, можно посчитать $\frac{\partial f}{\partial g}$ для каждого узла $g \in V$ следующим образом.

1. Сначала инициализируем $\frac{\partial f}{\partial f} = 1$.

2. Затем для каждой вершины $g \in V$, у которой все дети, то есть вершины, в которые идет из нее ребра, уже обработаны алгоритмом, вычислим

$$\frac{\partial f}{\partial g} = \sum_{g' \in Children(g)} \frac{\partial f}{\partial g'} \frac{\partial g'}{\partial g}. \tag{5}$$

Когда мы дойдем до истоков графа, до вершин $x_1, x_2, ..., x_n$, то получим частные производные $\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}$, т.е. вычислим градиент $\nabla_x f$. Данный алгоритм подсчета частных производных называется алгоритмом обратного распространения ошибки [4].

В качестве примера рассмотрим один слой нейронной сети с двумя входными нейронами и с одним выходным нейроном, у которого функцией активации будет сигмоида.

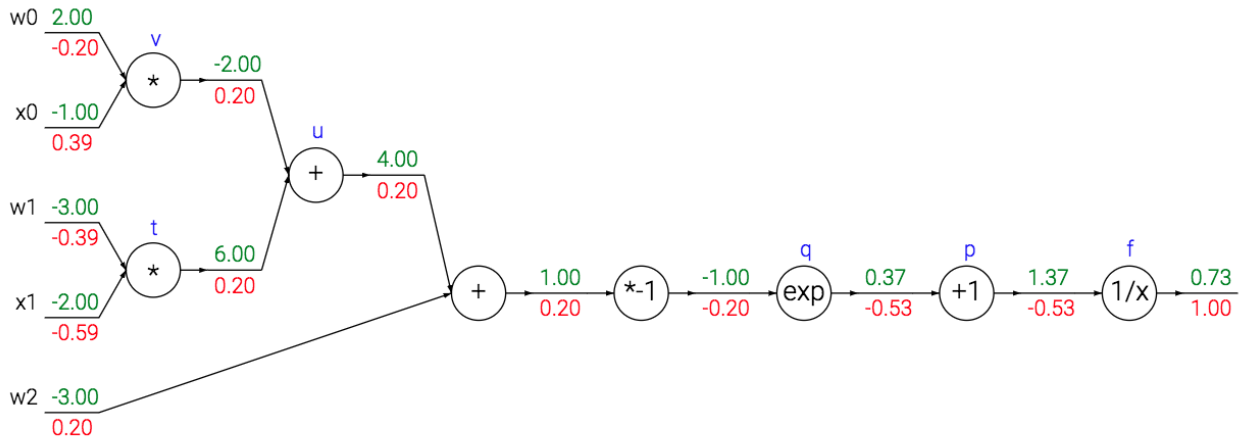

Рис. 3: Граф вычисления простой нейронной сети

На рисунке 3 показан граф вычислений такой сети. Числа стоящие над ребрами показывают значение функции после вершины, из которой исходит ребро. А числа под ребрами показывают значение производной конечной

функции f в вершине. Здесь для примера считается, что $w_0 = 2$, $x_0 = -1$, $w_1 = -3$, $x_1 = -2$, $x_2 = -3$, а все остальные числа, стоящие над ребрами, получены путем движения по графу вычисления. Числа, стоящие под ребрами, получены с помощью движения по графу от результирующей вершины к исходным и подсчете производной сложной функции в соответствующей вершине [18].

1.2. Задача обнаружения объекта в машинном обучение

Задача обнаружения объекта состоит из двух подзадач. Первая подзадача заключается в локализации объекта на изображении, т.е в нахождении минимального бокса, описывающий объект. Вторая подзадача решает задачу классификации локализованной области. Таким образом, результатом решения задачи обнаружения объекта является метка класса для бокса, а так же, четверка чисел (x_1, y_1, x_2, y_2), где (x_1, y_1) - координаты левого верхнего угла бокса, $(x2, y2)$ - координаты правого нижнего угла бокса.

1.3. Оценка качества

Для понимания того, как работает алгоритм, помимо значения функционала качества, которое минимизируется, полезно так же задать ряд метрик, которые расширили бы понимание качества работы алгоритма в целом. Это важно, так как алгоритм может переобучиться, то есть может слишком сильно подогнаться под тренировочный набор данных, при этом он не будет обладать обобщающей способностью.

Так как решение основывается на задаче классификации, то можно добавить следующие метрики.

1. Accuracy - доля правильных ответов алгоритма.
2. Precision - отношение количества положительных ответов алгоритма и количества всех положительных объектов.

3. Recall - отношение количества положительных объектов среди объектов, на которых алгоритм дал положительных ответ, и количества положительных объектов.

4. F1 - мера - средне-гармоническое между precision и recall.

Для их подсчета нужно предварительно составить матрицу ошибок алгоритма. Матрица ошибок представлена в таблице.

Таблица: Матрица ошибок

	$y_i = 1$	$y_i = -1$
$a_i = 1$	$True\ positive(TP)$	$False\ Positive(FP)$
$a_i = -1$	$False\ Negative(FN)$	$True\ Negative(TN)$

Примем следующие обозначения.

1. TP - алгоритм правильно классифицировал положительный класс.

2. TN - алгоритм правильно классифицировал отрицательный класс.

3. FN - алгоритм неправильно классифицировал отрицательный класс.

4. FP - алгоритм неправильно классифицировал положительный класс.

Основываясь на матрице ошибок, выведем следующие формулы.

1. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$.

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

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

4. $F1 = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$.

Стоит отметить, что значения этих метрик зависит от величины порога p, выше которого считается, что объект относится к определенному классу.

Для подбора оптимального порога используют, так называемую, precision-recall curve. Данная кривая представляет собой наглядное представление точности и полноты при различных порогах, что позволяет исследователю найти оптимальное значение метрик. Пример precision-recall curve представлен на рисунке 4.

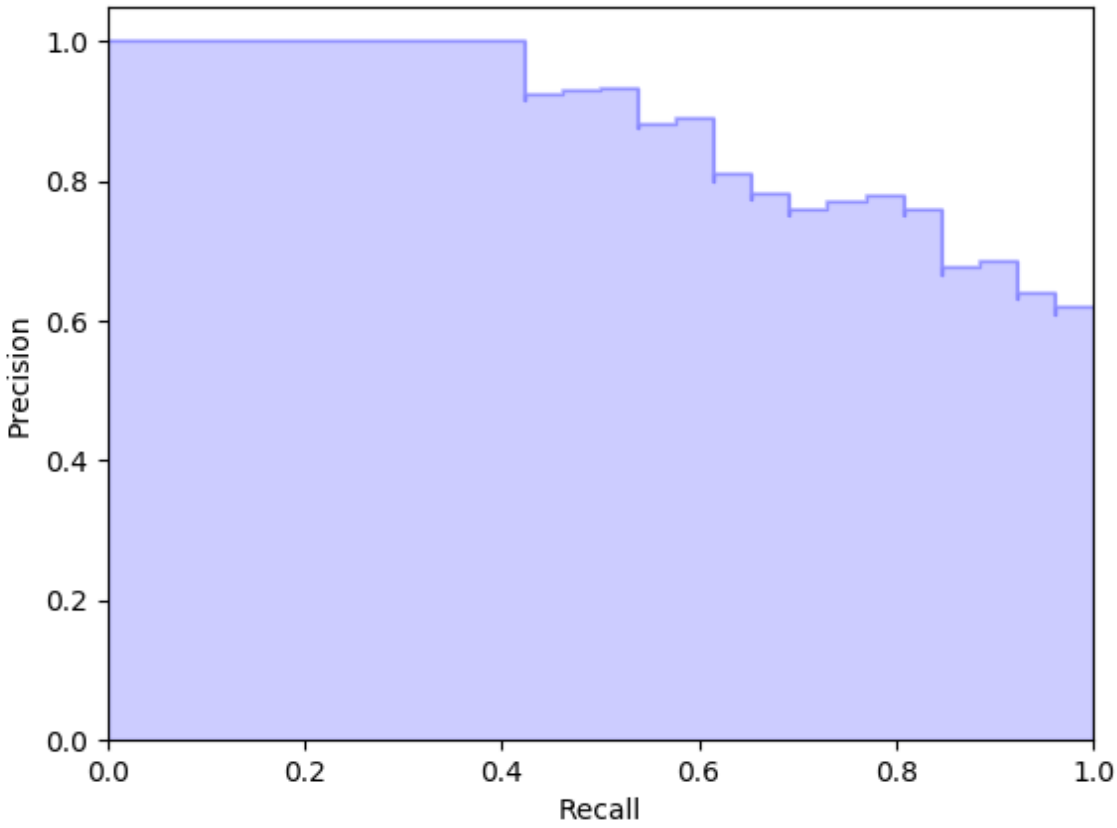

Рис. 4: Пример Precision - Recall curve

Перечисленные метрики являются базовыми метриками в задаче классификации объекта.

В задачах обнаружения объекта можно выделить такую метрику как IoU (intersection over union). Она означает отношение площадей пересечения найденной области и истинной области на объединение этих областей (рис. 5).

Для подсчета метрик не правильно использовать тот же набор данных, на которых модель обучалась, так как модель "видела"эти данные. Для решения данной проблемы тренировочный датасет, как правило, делят на три

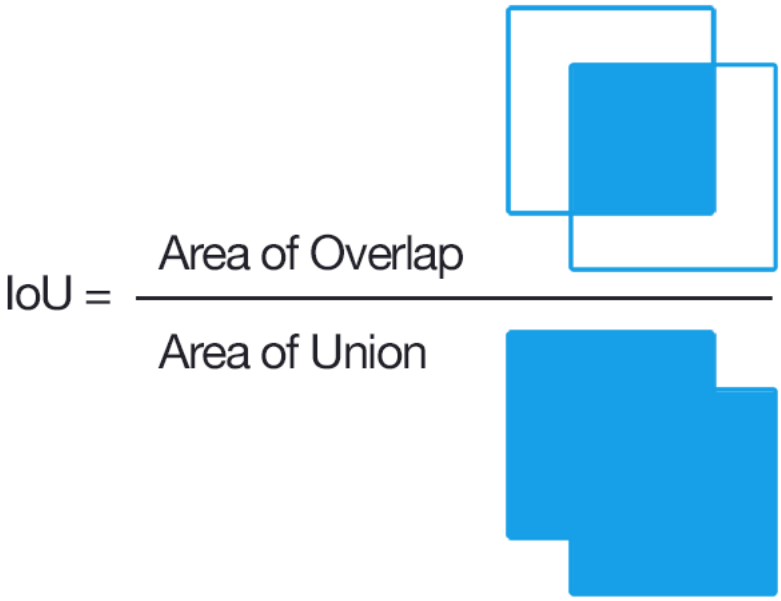

Рис. 5: Отношение пересечения и объединения.

части. Первая часть, доля которой составляет 60% тренировочных данных, служит для обучения модели, вторая часть, 20% - для настройки гиперпараметров модели, такие как шаг градиентного спуска ν или предел погрешности ϵ. На оставшейся части данных, проводят окончательное сравнение различных моделей и выбирают лучшую.

1.4. Обзор методов распознавания объектов без использования нейронных сетей

1.4.1. *Каскад Хаара*

Каскад Хаара представляет собой алгоритм, использующий так называемые признаки Хаара (рис. 6). Признаки Хаара являются набором прямоугольных областей. Во время работы алгоритма, вычисляется вес для каждой области. Данный вес представляет собой сумму яркостей пикселей области. Для подсчета яркости области изображения используют интегральное представление изображения. Для обучения алгоритма требуется относительно большой датасет с положительными и отрицательными примерами [14].

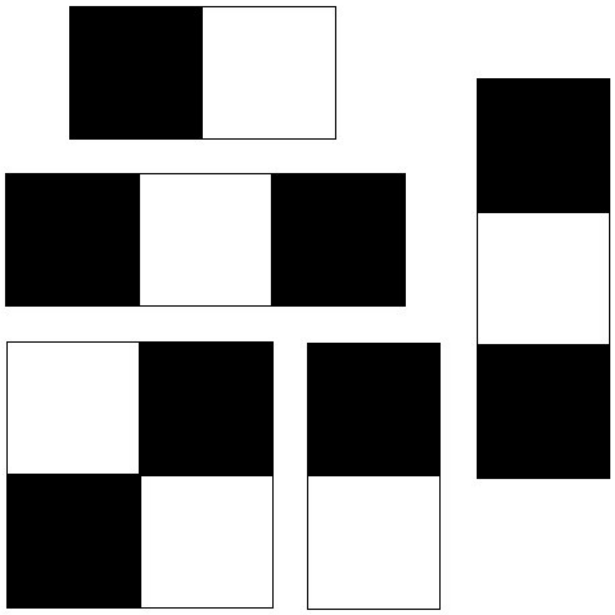

Рис. 6: Признаки Хаара

В ходе экспериментов на данных работодателя, было выявлено, что данный алгоритм показывает качество заметно хуже, чем нейросетевой подход. Связано это может быть с тем, что в целом, ценники порой сильно отличаются друг от друга, в то время, как каскад Хаара хорошо работает, когда объект не сильно меняет свои признаки.

1.4.2. *HOG/SVM*

Связка HOG/SVM представляет собой два алгоритма, первый из которых, выделяет признаки из изображения, а второй, на основании выделенных признаков пытается классифицировать изображение.

HOG - (Histogram of Oriented Gradients) дескрипторы особых точек, которые используются в компьютерном зрении и обработке изображений с целью распознавания объектов. Данная техника основана на подсчете количества направлений градиента в локальных областях изображения. Основная идея состоит в том, что форма объекта и его внешний вид могут описаны с помощью градиента интенсивности [11]. Результаты работы алгоритма HOG показаны на рисунке 7.

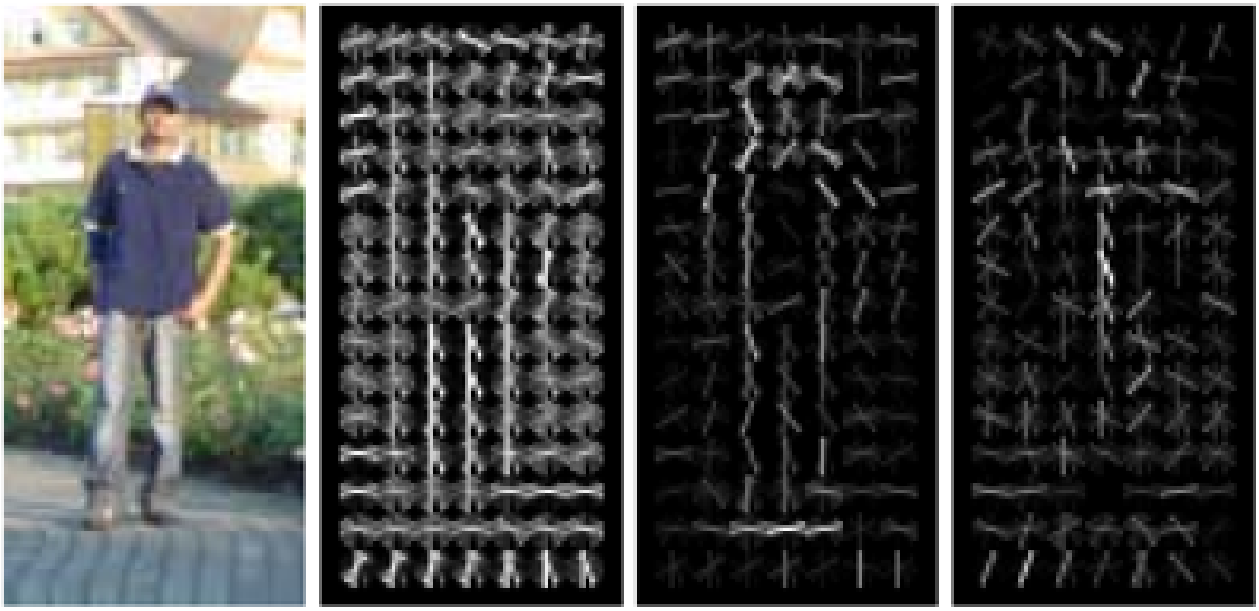

Рис. 7: Пример результата работа HOG дескриптора

SVM - (Support Vector Machine) алгоритм машинного обучения, в основе которого стоит идея построения такой разделяющей гиперплоскости между классами, чтобы отступ объектов от нее был максимальный [1]. На рисунке 8 представлена визуализация работы SVM дял решения задачи бинарной классификации.

Чаще всего данная связка применяется для задачи классификации изоб-

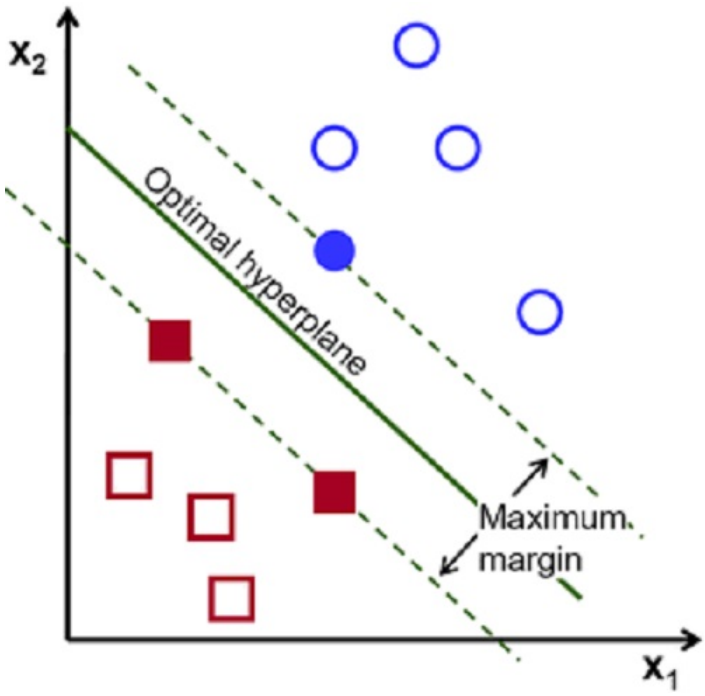

Рис. 8: Максимизация отступа в SVM

ражения, без задачи локализации объекта на изображении. В задачах обнаружения объекта поступают следующем образом. В цикле проходят скользящим окном фиксированного размера по изображению с самоперекрытием. Каждое такое окно классифицируют. В конце, смежные области, относящиеся к одному объекту - объединяют. При таком подходе возникают проблемы в выборе размера окна, в скорости работы алгоритма.

1.5. Обзор методов распознавания объектов с использованием нейронных сетей

1.5.1. *Полносвязная сеть прямого распространеннния*

На рисунке 9 представлена архитектура полносвязной нейронной сети в общем виде.

Полносвязная нейронная сеть прямого распространения применялась для задачи классификации изображения. Для задачи обнаружения объекта можно поступить так же, как и в случае HOG/SVM: в цикле пройти скользящим окном фиксированного размера с самоперекрытием, каждое окно клас-

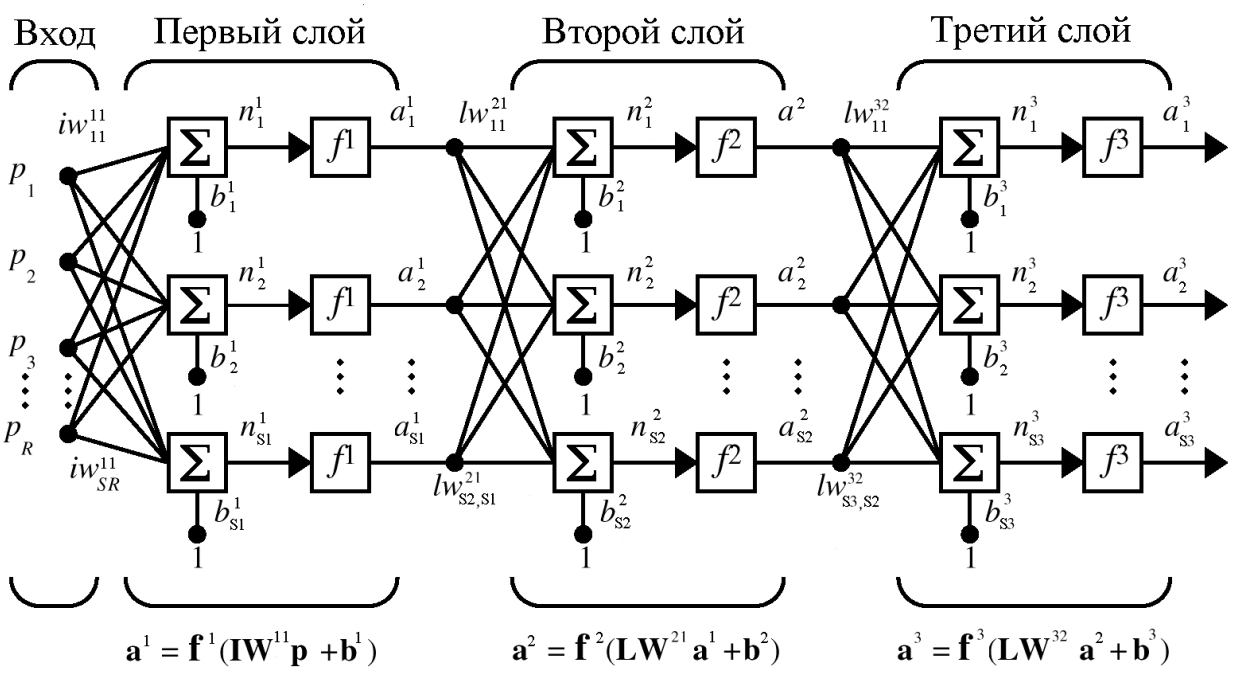

Рис. 9: Полносвязная нейронная сеть

сифицировать, в конце, смежные окна одного класса объединить. При этом минусом такого подхода является проблема выбора размера окна, а так же скорость работы алгоритма.

Однако, сейчас данных подход для классификации изображения считается нежелательным. Связано это с тем, что такая архитектура нейронной сети, принимает в качестве входных признаков вектор размера x^{n*m}, состоящий из пикселей изображения. При таком подходе, совсем не учитывается информации положении пикселя на изображении: смежные пиксели на изображении, имеющие сильную друг от друга зависимость, перестают находится рядом. Как следствие, теряется геометрическая структура изображения. Из за такого подхода, сеть имеет большое количество настраиваемых параметров, а это сильно увеличивает время обучения сети. При этом требуется большой тренировочный датасет.

1.5.2. *Сверточная нейронная сеть*

Сверточная нейронная сеть представляет собой алгоритм машинного обучения, предназначенный, в основном, для работы с изображениями, однако, встречаются работы, в которых данный алгоритм применяется для обработки последовательностей. Сам по себе алгоритм применяется для задачи классификации изображения, при этом, он не решает задачу локализации объекта на изображении. Для решения задачи локализации объекта поступают так же, как и в случае HOG/SVM: в цикле проходят скользящим окном фиксированного размера по изображению с самоперекрытием, каждое окно классифицируют, смежные окна одного класса объединяют в одну область.

Архитектура такой нейронной сети условно состоит из двух частей. Первая часть состоит из сверточных слоев и слоев субдискретизации. Вторая часть является полносвязной нейронной сетью, где входными признаками является карта признаков. Пример сверточной нейронной сети изображен на рисунке 10.

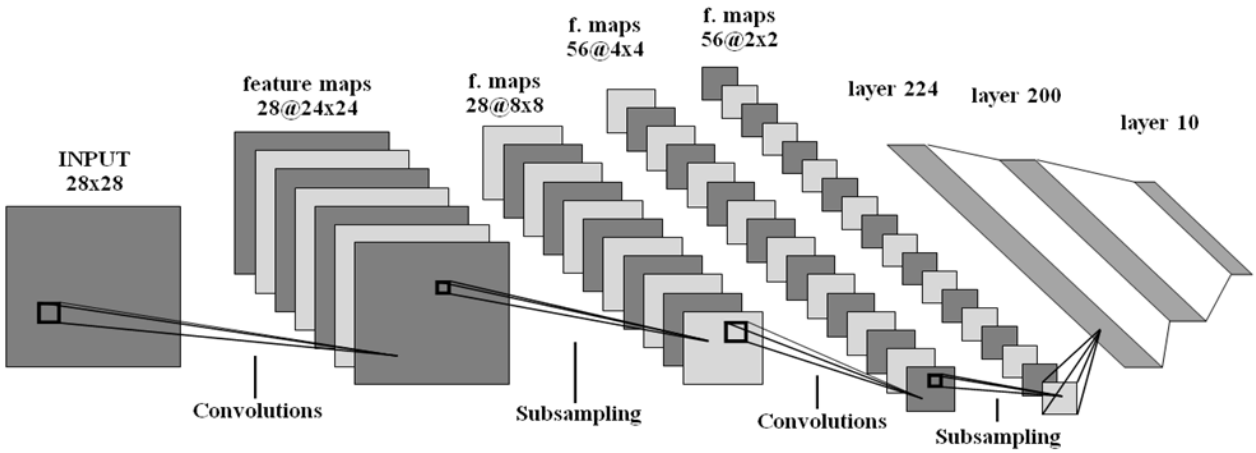

Рис. 10: Сверточная нейронная сеть LeNet

Сверточный слой представляет собой применение операции свертки на изображение как показано на 11. При этом, значение ячеек самой свертки не являются фиксированными, а настраиваются по ходу обучения алгоритма,

что позволяет выделять необходимые признаки для фотографий различной природы.

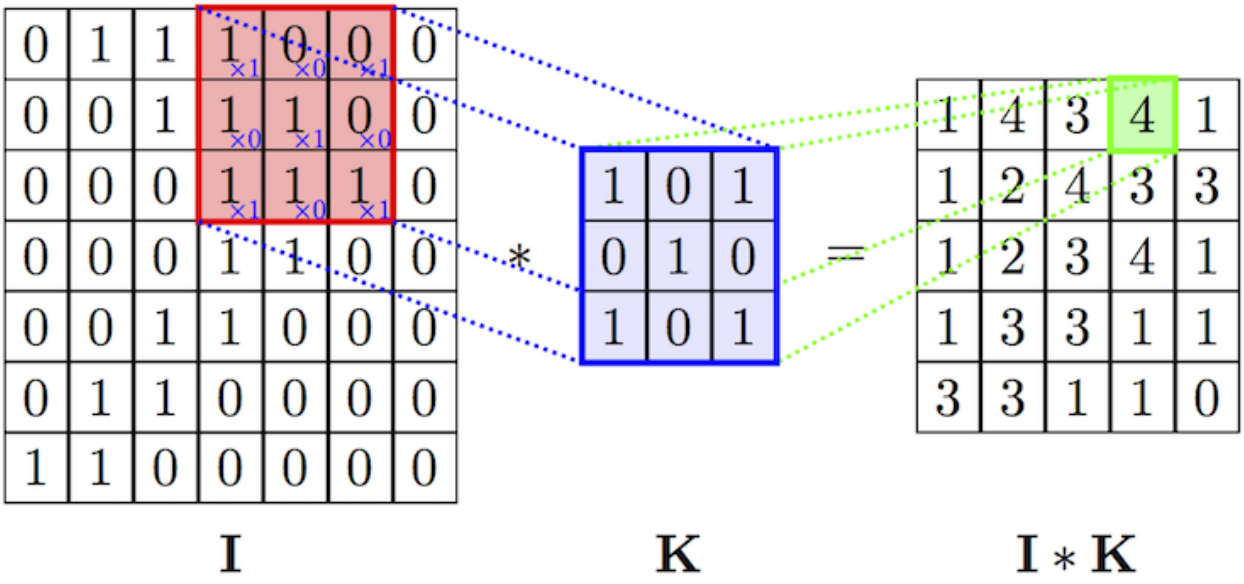

Рис. 11: Операция свертки

Слой субдискретизации представляет собой операцию уменьшения размера изображения. Часто используемые слои это MaxPooling и MeanPooling. Идея следующая: в цикле идут окном размером W^{n*n} по изображению без самоперекрытия, область W^{n*n} заменяют, в случае MaxPolling, на максимальное значение пикселя в этом окне, в случае MeanPooling - на среднее. Пример операции MaxPooling изображен на рисунке 12.

1.5.3. *Fully convolution network*

Архитектура Fully convolution network, в отличие от Convolution Neural Network, состоит только из сверточный слоев, слоев субдискретизации и слоев unsampling. Как правило, такую архитектуру применяют в задаче сегментации изображения, т.е. определения для каждого пикселя изображения к какому классу он относится [9]. Разница между задачей сегментации и обнаружения объектов состоит в том, что когда на изображении находятся два объекта одного класса, то в первом случае, не стоит вопрос о том, к какому конкретному объекту относится пиксель, в то время как во втором случае,

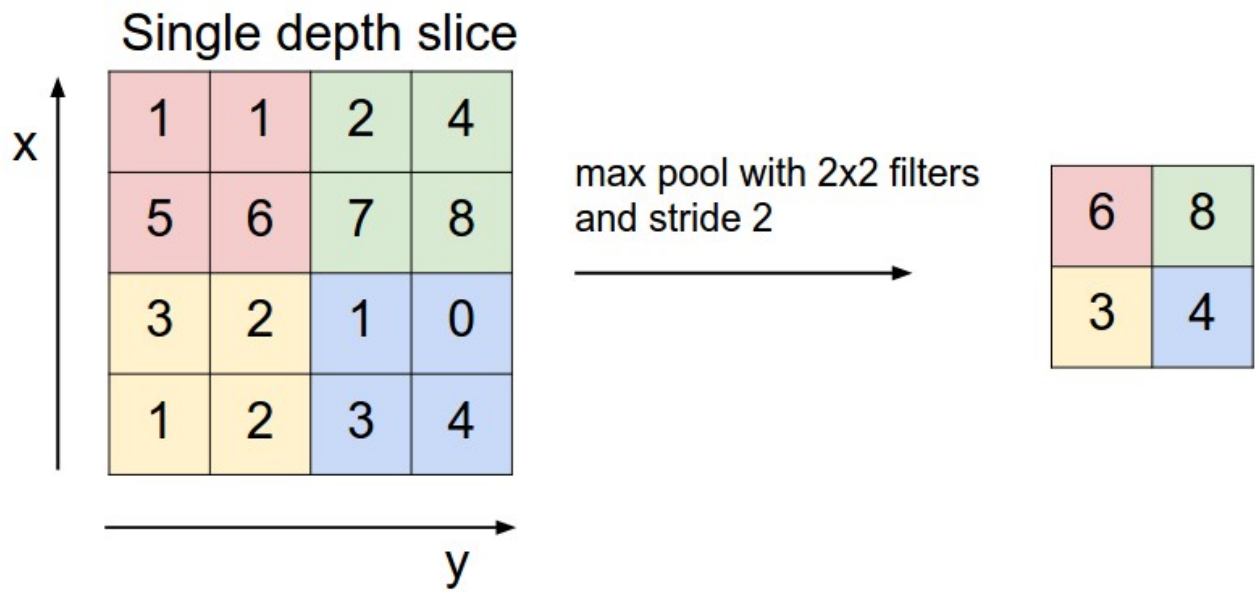

Рис. 12: MaxPooling

этот вопрос важен. Поэтому, одним из артефактов работы fully convolution network состоит в том, что близко расположенные объекты одного класса объединяются в одну область.

Один из плюсов такой архитектуры заключается в том, что без полносвязных слоев, такая архитектуры обладает меньшим количеством параметров для обучения, что ускоряет процесс обучения. При этом качество обнаружения объекта порой даже лучше.

На рисунке 13 представлен пример fully convolution network архитектуры, а на рисунке 14 показан пример работы операции MaxUnpooling.

Помимо слоев субдискретизации, в данной архитектуре есть unsampling слои. Если слои субдискретизации уменьшают размерность изображения, то unsampling слои увеличивают размерность.

1.5.4. *Сверточная нейронная сеть для классификации пикселей изображения*

CNN, у которой выход представляет собой изображение размером $w * h * k$, где w - ширина изображения, h - высота изображения, k - количество классов (как правило $w = h$). Минус такого подхода заключается в том, что

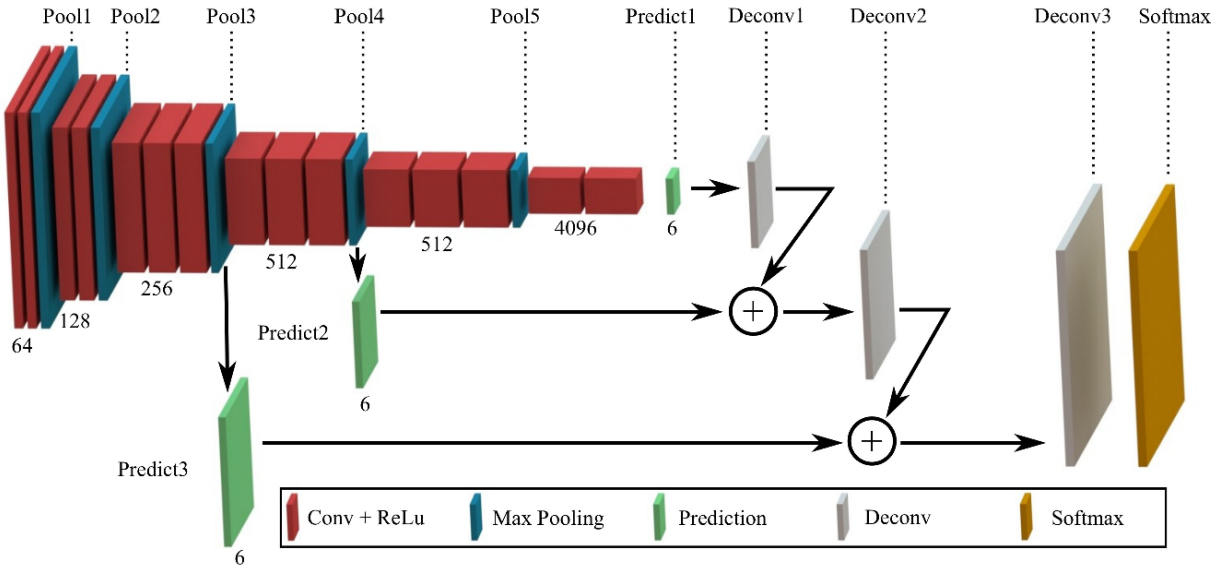

Рис. 13: Fully convolution network

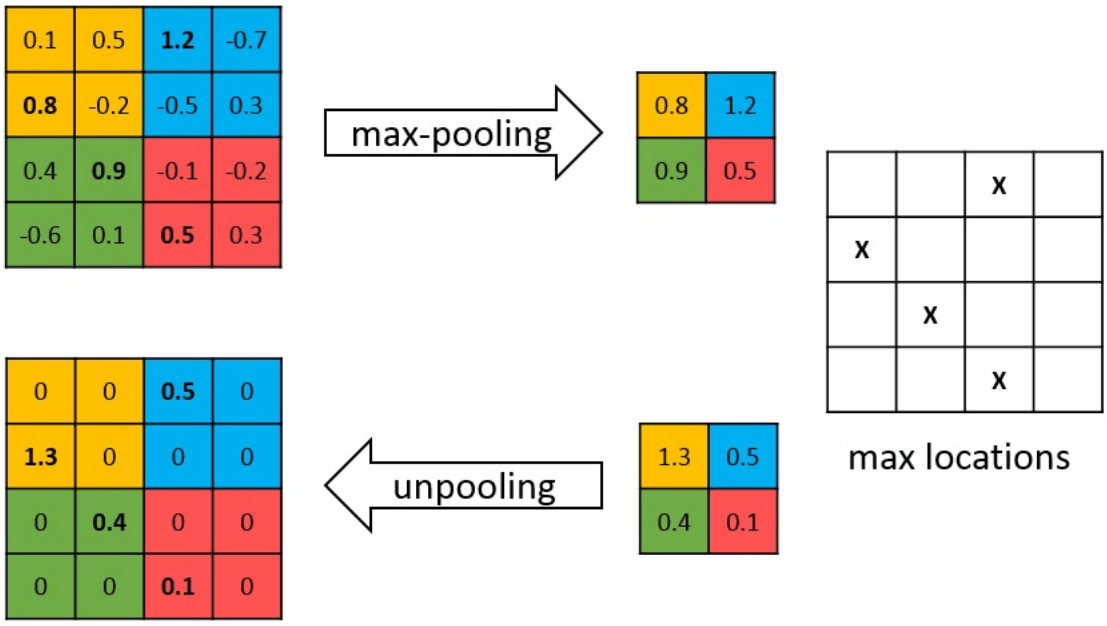

Рис. 14: MaxPooling и Max Unpooling

в случае большого количество классов, размер выходного слоя оказывается слишком большим, в следствие чего, требуется обучить большое количество параметров, что сказывается как на скорость обучения, так и на качество работы алгоритма в целом.

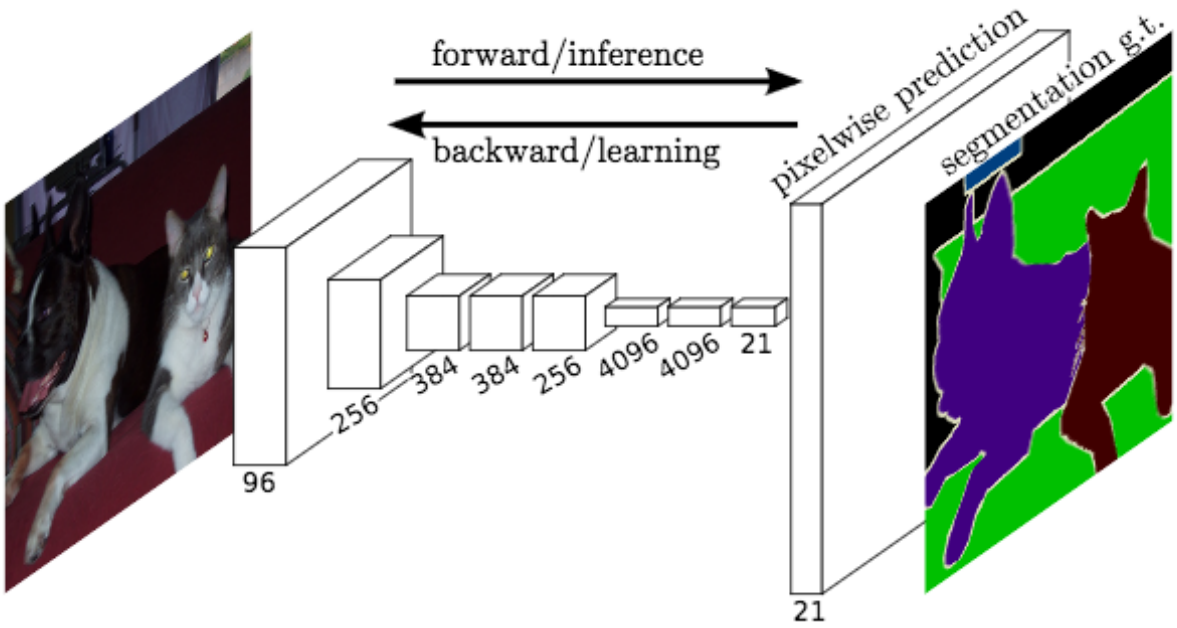

Рис. 15: Сверточная нейронная сеть для классификациии пикселей

1.5.5. *RCNN*

Нейросетевой подход для поиска объектов на изображении. Алгоритм состоит из нескольких шагов. На первом шаге с помощью детерминированного алгоритма происходит генерация гипотез того, где находится объект. В исходной статье таким алгоритмом был Selective Search. Результаты работы алгоритма можно посмотреть на рисунке 16.

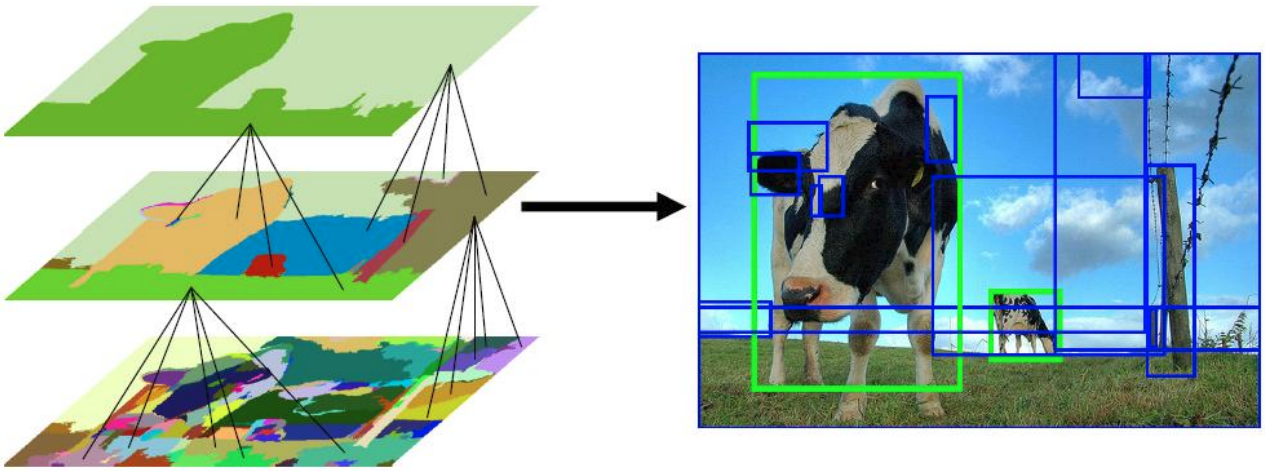

Рис. 16: Результат работы алгоритма Selective Search

Найденные области подаются на вход тяжелому алгоритму для даль-

нейшей классификации. Примером такого алгоритма может быть SVM или CNN. Помимо алгоритма для классификации используется линейный алгоритм для нахождения вещественных координат бокса, покрывающий нужный объект. Процедуры обучения алгоритма состоит из трех шагов. На первом происходит тренировка классификатора. На втором настройка параметров классификатора для уже задачи обнаружения объекта. На третьем шаге происходит обучение линейного регрессора[15].

Данный алгоритм обладает рядом недостатков.

1. Требуется три шага обучения, в отличие от Fast - RCNN и Faster - RCNN, где обучение происходит end-to-end.

2. RCNN зависит от выбора детерминированного алгоритма, использующийся для генерации гипотез. Такого рода алгоритмы могут работать по разному в зависимости от качества фотографии, размеров объектов на фотографии. Такие недостатки входных данных как раз свойственны реальным данным.

3. Детерминированный алгоритм генерации гипотез обладает высокой полнотой, но низкой точностью.

4. RCNN работают медленно. Связано с тем, результатом работы алгоритма для генерации гипотез может достигать до 2000 различных районов нахождения объекта. При этом каждую область необходимо классифицировать и выделить в бокс.

5. Классификатору требуется подавать изображения фиксированного размера. При изменении размера входного изображения качество фотографии может ухудшиться, что приведет к потере информативности.

Для решения поставленной задачи данный алгоритм является слишком тяжелым в обучении, требуется большой размеченный датасет. Так же RCNN не подходит из за времени работы.

1.5.6. *Fast RCNN*

Fast RCNN является следующим алгоритмом в семействе RCNN [7]. В отличие от RCNN, Fast RCNN сначала пропускает изображение через глубокие сверточные слои, затем на получившуюся карту признаков применяют алгоритм генерации гипотез, например Selective Search. Далее, для каждого региона применяют алгоритм Spatial Pyramid Pooling, идея которого схожа с "мешком слов" [16](рис. 17).

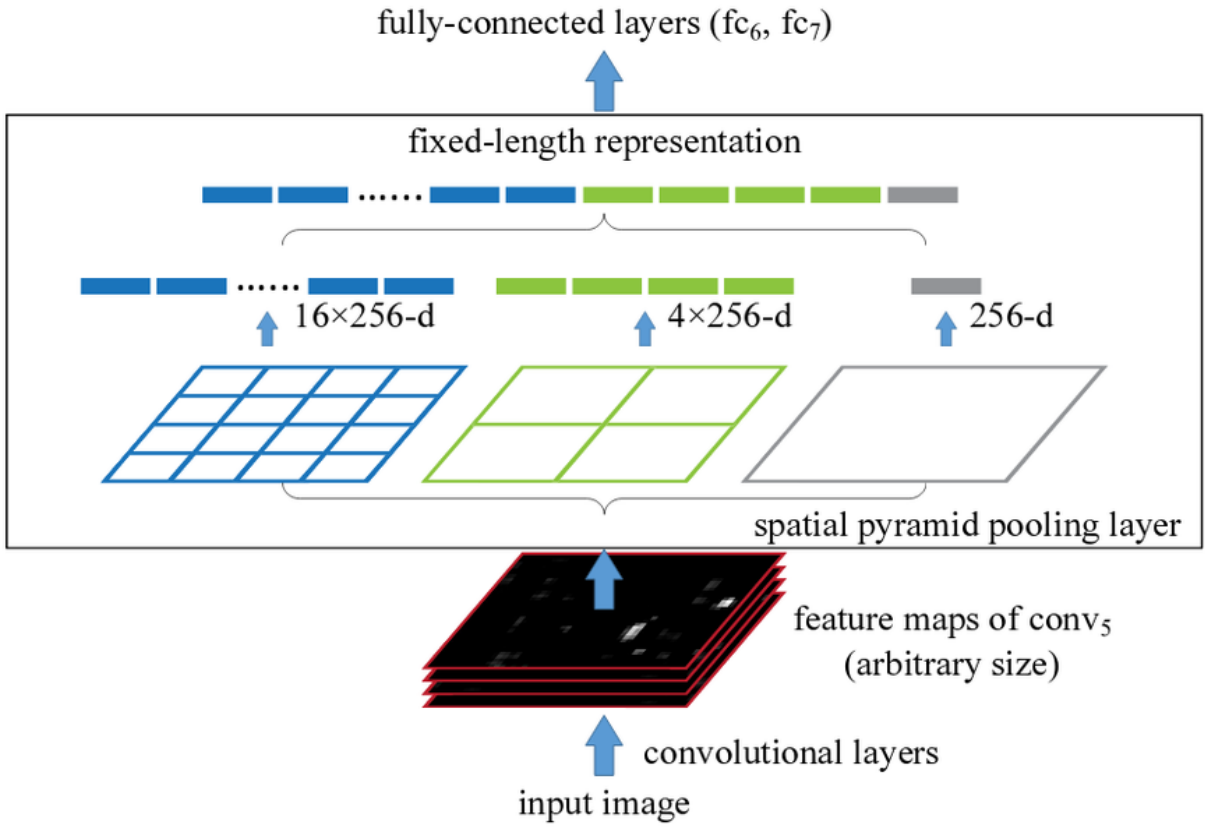

Рис. 17: Spatial Pyramid Pooling

Результатом работы SPP является вектор фиксированной длины, который характеризует регион. Этот вектор подается на вход полносвязной нейронной сети для дальнейшей классификации и локализации объекта. Для классификации, как правило, используют полносвязную нейронную сеть с функцией активации softmax, а для локализации - bbox regressor.

Таким образом, были решены следующие проблемы RCNN.

1. Обучение происходит end-to-end, т.е. обучение Fast RCNN происходит за один набор проход данных, вместо трех, как это было в RCNN.

2. Из за особенности архитектуры, обучение происходит быстрее.

3. Так же быстрее происходит обработка изображения обученным алгоритмом. Связано это с тем, что теперь изображение проходит через глубокие сверточные слои только один раз, вместо того, чтобы пропускать через глубокие сверточные слои каждую из гипотез.

4. Применение SPP повысило точность алгоритма.

Таким образом, Fast RCNN является более улучшенной версией RCNN, исправляя основные проблемы. При этом, все равно остается проблема в выборе алгоритма генерации гипотез и его настройки, так как он не настраивается в процессе обучения.

1.5.7. *Faster RCNN*

Faster RCNN является улучшением архитектуры Fast RCNN [8]. Улучшение заключается в том, что вместо детерминированного алгоритма генерации гипотез, требующий настройки, используется алгоритм, называемый RPN - Region Proposal Network, который обучается в процессе обучения модели. RPN является простой fully convolution network с multiloss.

Таким образом, Faster RCNN является алгоритмом Fast RCNN, где вместо детерминированного алгоритма генерации гипотез используется Region Proposal Network.

1.6. Вывод по первому разделу

В данной главе была поставлена задача машинного обучения. Показан подход к ее решению и оценке адекватности построенной модели. Так же был проведен обзор некоторых алгоритмов для решения поставленной задачи.

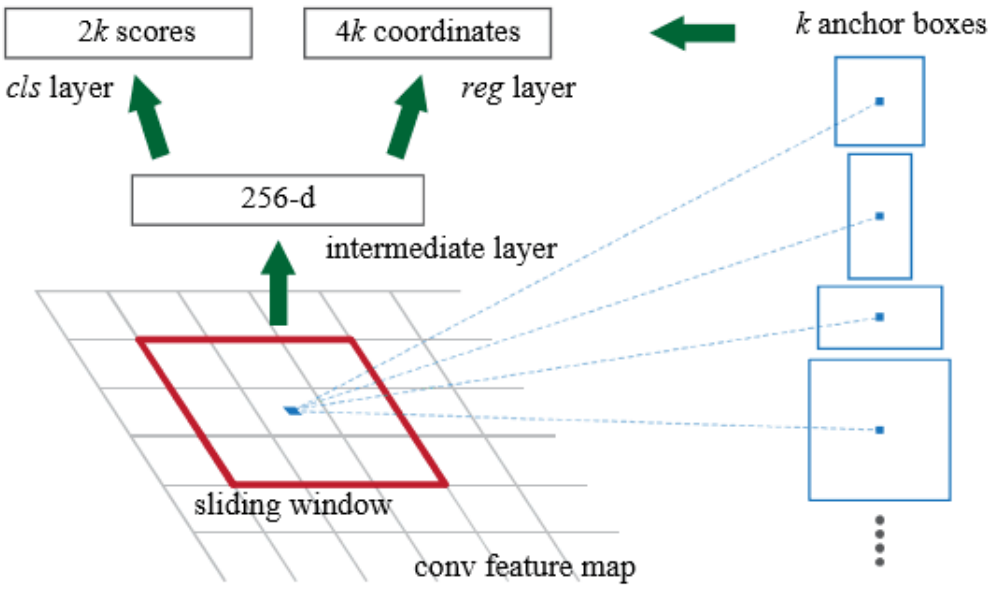

Рис. 18: Ragion Proposal Network

В результате обзора, было решено использовать сверточную нейронную сеть для классификации принадлежности пикселя к ценнику, так как данный алгоритм быстро обучается, не требователен к большим наборам данных и показывает достаточную точность. Следует отметить, что в перспективе, так же хорошей архитектурой для решения поставленной задачи является fully convolution network. По сравнению с решением она имеет ряд преимуществ. Во первых, выход такой сети соответствует размеру входного изображения, что позволяет не терять информацию при изменении масштаба, а во вторых, так как такая архитектура полностью состоит из сверточных слоев, то количество обучаемых параметров в разы меньше, что сказывается как на качестве работы алгоритма, так и на скорости сходимости. Однако, для такой архитектуры нужен больший по объему тренировочный датасет.

2. РЕШЕНИЕ ЗАДАЧИ ОБНАРУЖЕНИЯ ОБЪЕКТА С ПОМОЩЬЮ СВЕРТОЧНОЙ СЕТИ

2.1. Постановка задачи

На вход подается цветное изображение магазинной полки. Необходимо найти область, в которой находится ценник. При этом известно, что:

1. На изображении находится только один ценник.
2. Ценник занимает в среднем 30% - 60% изображения.
3. Изображение хорошего качества, четкое.

Рис. 19: Пример входного изображения

Пример входного изображения показан на рисунке 19.

2.2. Решение задачи обнаружения объекта

В качестве решения было решено взять сверточные слои у предобученной нейронной сети Mobile-Net, добавить к этим сверточным слоям полносвязный слой, количество нейронов в котором, соответствует количеству пикселей выходного изображения. Функцией активации последнего слоя будет softmax.

Архитектура нейронной сети MobileNet[12], которая использовалась для соревнования ImageNet представлена на рис. 20.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Рис. 20: Архитектура MobileNet

Таким образом, будет решаться задача классификации, где алгоритму необходимо определить относится ли пиксель к ценнику или нет.

Плюсом такого решения является то, что из за предобученных слоев, как правило, такая сеть требует для обучения меньше данных, а сам процесс обучения занимает меньшее количество времени. Это связано с тем, что предобученые сверточные слои уже умеют выделять некоторые признаки из изображения, и не требует грубой настройки.

Небольшую выборку изображений расширяют путем различных поворотов и масштабирования исходных изображений. Данный прием называет-

ся аугментацией данных. Этот прием основывается на идее, что измененные изображения воспринимаются нейронной сетью как новые, что позволяет улучшить обобщающую способность.

Всего в датасете 1966 изображений. Было решено разделить картинки в следующих пропорциях: первые 1300 изображений оставить на тренировочный датасет, с 1300 по 1600 на валидационную выборку, остальные на тест. Прежде чем обучать на них классификатор, необходимо сделать предобработку данных. Предобработка заключается в изменении размера картинки до размера $(224, 224, 3)$, а так же в масштабировании значений пикселей. Операция масштабирования:

$$X' = \frac{X - M(X)}{std(X)}, \tag{6}$$

где X - массив, соответствующий изображению, $M(X)$ - среднее значение пикселей в массиве, $std(X)$ - среднеквадратическое отклонение значений пикселей от их среднего значения.

2.3. Алгоритм решения

Алгоритм решения представлен ниже.

1. Вход: ν - величина шага градиентного спуска, $batch\_size$ - количество картинок, использующихся для стохастического градиентного спуска, $data\_path$ - путь до директории, в которой содержатся изображения, $metadata\_path$ - имя файла, в котором содержатся метаданные.

2. Считывание метаданных, изображений. Предобработка изображений. Разделение датасета на тренировочную и тестовую выборки

3. Создание модели

4. Обучение модели на тренировочной выборке с помощью алгоритма стохастического градиентного спуска.

5. Проверка адекватности модели на тестовой выборке.

6. Выход: обученная модель, результаты работы на тестовой выборке.

В результате работы алгоритма получена обученная сверточная нейронная сеть. На рисунке 21 предоставлена блок-схема алгоритма обучения.

Рис. 21: Блок схема алгоритма обучения

2.4. Обоснования инструментов разработки

Весь код программы был написан с использованием языка программирования Python3.6. На сегодняшний момент, данный язык предоставляет удобные библиотеки для обучения различных алгоритмов машинного обучения, а так же для работы с данными. Для написания и обучения нейронной сети использовалась библиотека Keras, которая предоставляет удобный ин-

терфейс для конструирования своих архитектур, при этом освобождает разработчика от внутренней реализации. В качестве бэкенда для библиотеки Keras использовался tensorflow. Для работы с изображениями использовалась библиотека opencv-python, а для работы с метаданными - библиотека pandas.

Наличие GPU является критическим фактором для скорости обучения глубокий нейронных сетей. Поэтому обучение нейронной сети происходило с использованием Google Colaboratory, в которой дается GPU Tesla K80 с 13 Гб видеопамяти. Google Colaboratory это форк популярной среды Jupyter notebook. Код исполняется на сервере в docker контейнере. Таким образом, обучение модели занимает в среднем 25 секунд на одну эпоху.

Код программы содержится в приложение.

2.5. Результаты работы программы

В результате работы программы были получены графики 22 и 23.

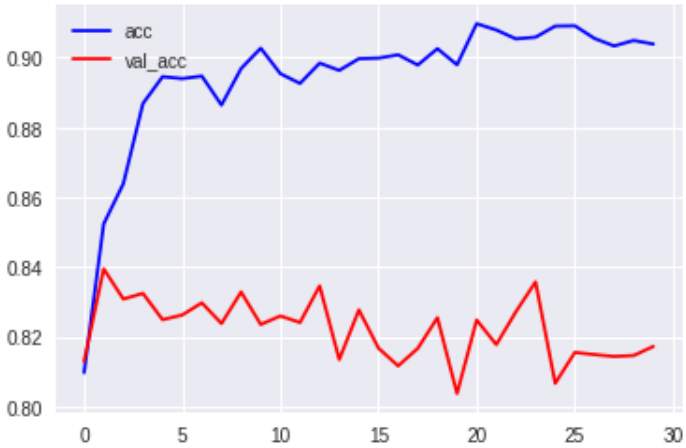

Рис. 22: График значений accuracy от номера эпохи на тренировочной и валидационной выборках

На графике 23 изображена зависимость значения минимизируемого функционала на тренировочной и валидационной выборках от номера эпохи. Видно, что график значений функционала на валидационной выборке быстро

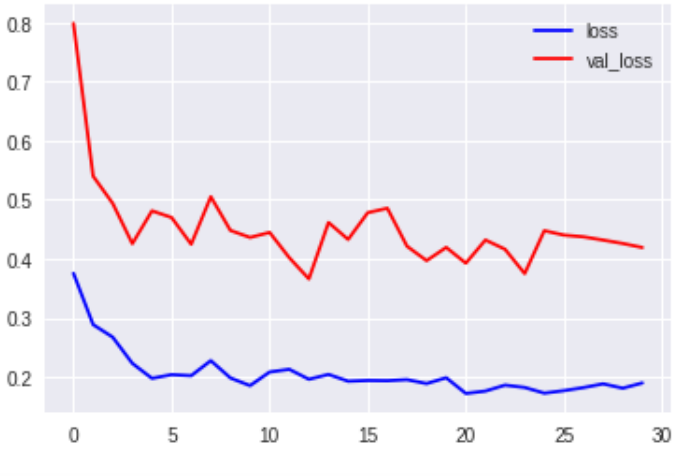

Рис. 23: График значений функции loss от номера эпохи на тренировочной и валидационной выборках

доходит до минимума и, начиная, примерно, с 10-ой эпохи движется почти горизонтально. То же можно сказать и про значение функционала на тренировочной выборке. Так же видно, что значения на тренировочной и валидационной выборках отличаются, что может говорить о том, что модель недообучена. Это значит, что увеличение количества эпох может улучшить качество модели.

На графике 22 изображена зависимость значения accuracy на тренировочной и валидационной выборках от номера эпохи.

В качестве способов улучшения можно выделить следующие направления.

1. Попробовать настроить гиперпараметры, такие как шаг градиентного спуска ν.
2. Увеличить количество эпох обучения нейронной сети.
3. Попробовать другие архитектуры, например fully convolution network или взять другую предобученную сеть вместо MobileNet.
4. Попробовать другие алгоритмы оптимизации.
5. Увеличить количество данных.

6. Обучить свою архитектуру сверточной нейронной сети с меньшим количеством слоев.

В результате анализа, можно сделать вывод, что полученное решение обладает достаточным качеством работы. При этом обладает рядом недостатков, связанных прежде всего, с особенностями датасета, используемого для обучения. В связи с этим, были предложены возможные подходы по улучшению качества работы модели.

2.6. Вывод по второму разделу

Таким образом, в результате выполнения выпускной квалификационной работы был обучен базовый алгоритм для решения задачи обнаружения объекта с помощью сверточной нейронной сетей. Полученное решение простое в реализации и при этом уже достигает, пусть и не идеальных, но достаточных значений итоговых метрик. При этом

Таким образом, в результате выполнения выпускной квалификационной работы была обучена сверточная нейронная сеть, решающая поставленную задачу. Полученное решение имеет необходимое качество работы, при этом обладает рядом недостатков. Среди недостатков можно выделить следующие.

1. Модель недообучена.
2. Изображение, в процессе обработки алгоритмом, уменьшает свои размеры, что приводит к потере информации.
3. Некорректно работает на изображениях, если на изображении нет ценника или их несколько.

При этом были выдвинуты ряд идей, способных решить описанные проблемы и улучшить качество работы алгоритма. Проблему недообучения решается с помощью дополнительного времени на обучении и расширении обу-

чающей выборки. Проблема уменьшения размеров изображения можно решить используя другую архитектуру нейронной сети. Например, можно использовать fully convolution network или уменьшить количество сверточных слоев в исходной модели. Некорректность работы на изображениях, в которых содержится больше чем один объекта, можно исправить расширив обучающую выборку изображениями, в которых содержится больше чем один объект.

ЗАКЛЮЧЕНИЕ

Задача обнаружения объекта на изображении является одной из основных задач компьютерного зрения. Применения нейронных сетей позволило сильно улучшить решения данной задачи.

В работе была поставлена задача обнаружения объекта в терминах машинного обучения и решена в рамках задачи классификации с использованием сверточных нейронных сетей. Для решения поставленной задачи был сделан обзор основных алгоритмов компьютерного зрения и машинного обучения. С помощью обзора была выбрана архитектура нейронной сети. Выбранное решение в итоге было программно реализовано. В процессе обучения модели были собраны значения основных метрик, что позволило провести анализ полученного решения. Таким образом, все поставленные задачи решены, цель достигнута.

В ходе анализа было выяснено, что полученное базовое решение обладает достаточным значением итоговых метрик, но при этом оно обладает рядом недостатков:

1. Полученная модель недообучена.
2. В процессе обработки изображения алгоритмом, исходное изображение сжимается, что приводит к потере информации, которая могла бы улучшить качество работы модели.

Недообученность модели можно исправить добавив количество эпох обучения, а проблему сжатия изображения и потери информативности можно решить с помощью уменьшения количества сверточных слоев. Таким образом, данные проблемы легко устранимы.

Для улучшения работы алгоритма были даны следующие идеи:

1. Попробовать настроить гиперпараметры, такие как шаг градиентного спуска ν.

2. Увеличить количество эпох обучения нейронной сети.
3. Попробовать другие архитектуры, например fully convolution network или взять другую предобученную сеть вместо MobileNet.
4. Попробовать другие алгоритмы оптимизации.
5. Увеличить количество данных.
6. Обучить свою архитектуру сверточной нейронной сети с меньшим количеством слоев.

СПИСОК ЛИТЕРАТУРЫ

[1] Вьюгин, В. Математические основы теории машинного обучения и прогнозирования / В. Вьюгин — МЦМНО, 2013. — 390 с. — ISBN 978-5-4439-0111-4.

[2] Колмогоров, А.Н. О представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения // Докл. АН СССР. Том 114. С. 953-956. 1957.

[3] Круглов, В.В. Искусственные нейронные сети. Теория и практика. - 2-е изд., стереотип. / В.В. Круглов, В.В. Борисов - М.: Горячая линия-Телеком, С. 20-21. 2002.

[4] Николенко, С. Глубокое обучение: погружение в мир нейронных сетей / С. Николенко, А. Кадурин, Е. Архангельская, , СПб.: Питер, 2018. — 480 с.

[5] Goodfellow, I. Deep Learning /Ian Goodfellow, Yoshua Bengio, Aaron Courville, MIT Press, 2016.

[6] David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints [Электронный ресурс]. URL:https://www.cs.ubc.ca/ lowe/papers/ijcv04.pdf (дата обращения: 01.05.2018).

[7] Ross Girshick, Fast R-CNN [Электронный ресурс]. URL: https://arxiv.org/pdf/1504.08083.pdf (дата обращения: 01.05.2018).

[8] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, [Электронный ресурс]. URL:https://arxiv.org/pdf/1506.01497.pdf (дата обращения: 01.05.2018).

[9] Evan Shelhamer, Jonathan Long, Trevor Darrell, Fully Convolutional Networks for Semantic Segmentation [Электронный ресурс]. URL:https://arxiv.org/pdf/1605.06211.pdf (дата обращения: 01.05.2018).

[10] Brooks, Binford, Generalized Cylinder, 1979.

[11] M. Kachouane, S. Sahki, HOG Based fast Human Detection [Электронный ресурс]. URL:https://arxiv.org/pdf/1501.02058.pdf (дата обращения: 01.05.2018).

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Электронный ресурс]. URL:https://arxiv.org/pdf/1704.04861.pdf (дата обращения: 01.05.2018).

[13] W.T. Cathey and E.R. Dowski, New paradigm for imaging systems, Appl. Opt. 41, pp. 6080-6092, 2002.

[14] Paul Viola, Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features [Электронный ресурс]. URL:http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf (дата обращения: 01.05.2018).

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, Rish feature hierarchies for accurate object detection and semantic segmentation [Электронный ресурс]. URL:https://arxiv.org/pdf/1311.2524.pdf (дата обращения: 01.05.2018).

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition [Электронный ресурс]. URL:https://arxiv.org/pdf/1406.4729.pdf (дата обращения: 01.05.2018).

[17] David Marr, Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. New York: Freeman. 1982.

[18] Stanford University, cs231n:Convolutional Neural Networks for Visual Recognition, Spring 2017.

ПРИЛОЖЕНИЕ

```
# Пакет os нужен для работы с файловой системой
import os
# Пакет pandas нужен для работы с csv файлами,
# в которых хранится информация о нужном контуре
import pandas as pd
# Пакет для работы с картинками.
# Позволяет картинки считывать, поворачивать, изменять размер
import cv2
# Пакет для работы с массивами
import numpy as np
# импортируем предобученую модель
# и необходимый для нее препроцессинг
from keras.applications.mobilenet import MobileNet,
                                         preprocess_input
# Генератор картинок, используемый для расширения
# обучающей выборки. Поворачивает, зумит картинку
from keras.preprocessing.image import ImageDataGenerator
# Встроеные стандартные слои
from keras.layers import Dense, GlobalAveragePooling2D
# Алгоритм оптимизации.
from keras.optimizers import SGD
# Класс для создания своей модели
from keras.models import Model

# функция для создания маски: изображения,
# где отмечена область ценника
def create_mask(shape, contour):
    x1, y1, x2, y2 = contour
    mask = np.zeros(shape)
    mask[y1: y2, x1: x2] = 1
    return mask

# загрузка файла с метаданными по каждой картинки
metadata = pd.read_csv('info.csv')
```

```
# получаем список картинок,
# лежащий в директории 'images/'
images = os.listdir('images/')

# пустые списки для записи картинок
X = []
y = []

# список названий нужных колонок для работы с DataFrame
col = ['x_min', 'y_min', 'x_max', 'y_max']
# цикл, внутри которого идем по списку картинок,
# считываем картинку, делаем препроцессинг,
# находим метаданные по картинке,
# по метаданным создаем маску изображения,
# изменяем размер
# исходного изображения и его маски, добавляем в список
    for image in images:
    img = cv2.imread(os.path.join('images/', image))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = preprocess_input(img.astype(float))
    meta_by_img = metadata[metadata.image_name == image]
    contour = meta_by_img.iloc[0][col].tolist()
    mask = create_mask(img.shape, contour)
    img = cv2.resize(img, (224, 224))
    mask = cv2.resize(mask, (32, 32))
X.append(img)
y.append(mask)

X = np.array(X)
y = np.array(y)

class LossHistory(Callback):
    def on_train_begin(self, logs={}):
        self.losses = []

def on_batch_end(self, batch, logs={}):
```

```
    self.losses.append(logs.get('loss'))

history = LossHistory()

X_train = X[:1300]
y_train = y[:1300]
X_val = X[1300:1600]
y_val = y[1300:1600]
X_test = X[1600:]
y_test = y[1600:]

# зададим константы
train_seed = 777
val_seed = 222
batch_size = 64

# создание генераторов для непрерывного
# расширения обучающей выборки
image_generator = ImageDataGenerator(rotation_range=10,
                                     width_shift_range=0.1,
                                     height_shift_range=0.1,
                                     horizontal_flip=True,
                                     vertical_flip=True,
                                     zoom_range=0.25)

mask_generator = ImageDataGenerator(rotation_range=10,
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    horizontal_flip=True,
                                    vertical_flip=True,
                                    zoom_range=0.25)

def get_mask_generator(y, seed):
    for batch in mask_generator.flow(y,
                                     batch_size=batch_size,
                                     seed=seed):
```

```
        shape = (batch.shape[0], batch.shape[1] * batch.shape[2])
        yield batch[:, :, :, 0].reshape(shape)

train_generator = zip(image_generator.flow(X_train,
                                           batch_size=batch_size,
                                           seed=train_seed),
                      get_mask_generator(y_train, train_seed))

val_generator = zip(image_generator.flow(X_val,
                                         batch_size=batch_size,
                                         seed=val_seed),
                    get_mask_generator(y_val, val_seed))

# создадим модель, взяв за основу предобучению модель
model_mobilenet = MobileNet(input_shape=(224, 224, 3),
                            include_top=False)
model_output = GlobalAveragePooling2D()(model_mobilenet.output)
model_output = Dense(32 * 32,
                     activation='sigmoid')(model_output)
MODEL = Model(input=model_mobilenet.input,
                  output=model_output)

# скомпилируем модель
MODEL.compile(loss='binary_crossentropy',
                  optimizer=SGD(),
                  metrics=['binary_crossentropy', 'accuracy'])

# запустим обучение
MODEL.fit_generator(train_generator,
                        steps_per_epoch=len(X_train) / batch_size,
                        epochs=30,
                        verbose=1,
                        validation_data=val_generator,
                        validation_steps=len(X_val) / batch_size,
                        callbacks=[history])
```