МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»

Институт (факультет) Институт информатики и телекоммуникаций
Направление 27.04.03 Системный анализ и управление
Магистерская программа Системный анализ данных и моделей принятия решений
Кафедра Системного анализа и исследования операций

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ В ЗАДАЧАХ КЛАССИФИКАЦИИ ВИЗУАЛЬНЫХ ДАННЫХ

Обучающийся _____ М. В. Гордиенко
подпись инициалы и фамилия
Руководитель _____ Е. С. Семёнкин
подпись инициалы и фамилия
Рецензент _____ _____
подпись инициалы и фамилия

Ответственный за нормоконтроль _____ И. С. Масич
подпись инициалы и фамилия

Допускается к защите
Заведующий кафедрой _____ Л. А. Казаковцев
подпись инициалы и фамилия
«_____» _____ 20__ г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»

ИНСТИТУТ ИНФОРМАТИКИ И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА СИСТЕМНОГО АНАЛИЗА И ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _Л.А. Казаковцев_
подпись инициалы, фамилия
« ____ » _____ 20 ___ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме магистерской диссертации

Обучающийся Гордиенко Максим Владимирович

Группа МСД18-01 Направление (специальность) 27.04.03 Системный анализ и управление

«Системный анализ данных и моделей принятия решений»

Тема выпускной квалификационной работы: «Исследование эффективности сверточных нейронных сетей в задачах классификации визуальных данных»

Утверждена приказом по университету от 26.03.2020 № 647

Руководитель ВКР – Е.С. Семёнкин, профессор кафедры системного анализа и исследования операций института информатики и телекоммуникаций, профессор, д-р. техн. наук, СибГУ им. М.Ф. Решетнева

Исходные данные для ВКР Алгоритмы обнаружения областей объектов на изображениях, алгоритмы распознавания объектов на изображениях, сверточные нейронные сети для детектирования и локализации объектов, тестовый материал

Перечень разделов ВКР 1. Анализ сверточных нейронных сетей для решения задач классификации и локализации; 2. Исследование эффективности сверточных нейронных сетей при решении задачи детектирования объектов

Перечень графического материала (с указанием обязательных чертежей при необходимости) _____

Срок сдачи обучающимся первого варианта ВКР – « _04_ » ___05_____ 2020 г.

Срок сдачи обучающимся окончательного варианта ВКР – « _15__ » _06_ 2020 г.

Руководитель ВКР _____ ____Е.С. Семёнкин_____
подпись инициалы и фамилия

Задание принял к исполнению _____ ___М.В. Гордиенко_____
подпись инициалы и фамилия обучающегося

« ___ » _____ 20__ г.

АННОТАЦИЯ

к магистерской диссертации

«Исследование эффективности сверточных нейронных сетей в задачах классификации визуальных данных»

Гордиенко Максим Владимирович

Ключевые слова: сверточные нейронные сети, задачи классификации и локализации.

В магистерской диссертации рассматриваются вопросы развития сверточных нейронных сетей для решения задач классификации и локализации. Рассматривается исследование эффективности архитектур сверточных нейронных сетей для задачи распознавания цветов. Представлена интеллектуальная система определения расстояния системы до детектируемых объектов при помощи стереокамеры.

Работа включает: 40 страницах, 4 таблицы, 23 рисунка. Использованных источников – 27.

ABSTRACT

to the master's thesis

« Research of efficiency of convolutional neural networks in problems of classification of visual data »

Keywords: convolutional neural networks, classification and localization task.

The master's thesis discusses the development of convolutional neural networks to solve classification and localization problems. A study of the effectiveness of convolutional neural network architectures for the task of color recognition is considered. An intelligent system for determining the distance of a system to detected objects using a stereo camera is presented.

The work includes: 40 pages, 4 tables, 23 figures. Sources used – 27.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. Компьютерное зрение – область научных и прикладных исследований, направленных на извлечение и последующее использование информации об объектах внешнего мира, полученной из изображений. Первоначальной задачей компьютерного зрения было восстановление пространственной структуры объекта (3D) по плоским изображениям (2D). Затем, данная задача была расширена в сторону анализа состояния объекта по изображениям

Задачи компьютерной обработки и анализа изображений открывают широкие перспективы автоматизации многих сфер человеческой деятельности. Новые подходы в области задачах классификации, локализации, сегментации объектов берут в курс в сторону быстродействия и точности, что позволяет создавать системы распознавания в режиме реального времени. Усовершенствование быстродействия таких подходов позволяет создать различные системы распознавания, что делает возможным их применение в таких сферах жизни, как маркетинг, где, например, такие подходы используются для поиска определенного бренда одежды, идентификации правонарушителей системой городского видеонаблюдения, диагностики раковых заболеваний на снимках КТ и МРТ в медицине. Также подходы в решении задач классификации и локализации активно используются в робототехнике, например, для использование летательных и сухопутных дронов для анализа энергосетей в труднодоступных для человека местах.

Целью магистерской диссертации является оценка эффективности сверточных нейронных сетей, а также разработка интеллектуальной системы для определения расстояния до детектированных при помощи сверточных нейронных сетей объектов.

Для достижения поставленной цели необходимо решить следующие задачи:

Обзор современных источников литературы по основной теме исследования.

Подготовка данных для обучения сверточных нейронных сетей посредством аннотирования изображений.

Обучение разных архитектур сверточных нейронных сетей и определение их эффективности.

Разработка интеллектуальной системы определения расстояния до локализированных объектов на основе стереокамеры.

Научная новизна заключается в разработке интеллектуальной системы определения расстояния до локализированных объектов на основе методов глубокого обучения, а также использования стереокамеры.

Апробация и публикации. В процессе работы над магистерской диссертацией была опубликована статья:

– в сборнике VI Международной научной конференции «Актуальные проблемы авиации и космонавтики» (г. Красноярск, 2020).

Структура работы. Магистерская диссертация состоит из введения, двух глав, заключения и списка использованных источников, состоящего из 27 наименований. Изложена на 40 страницах, содержит 23 рисунка и 4 таблицы.

1 АНАЛИЗ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧ КЛАССИФИКАЦИИ И ЛОКАЛИЗАЦИИ

1.1 Метрики оценки качества сверточных нейронных сетей

Оценка эффективности решения в задаче детектирования и локализации объектов не является тривиальной, так как измерение должно включать в себя информацию о наличии объекта на изображении (классификация), а также определение места его расположения (локализация). Кроме того, в наборе данных, где распределение классов неравномерно, простая метрика, основанная на точности, будет вводить отклонения. Также важно оценить риск неправильной классификации. Таким образом, необходимо связать показатель достоверности и соотнести предсказанные ограничивающие рамки с истинными.

1.1.1 Точность и полнота

Для оценки эффективности используются такие величины, как точность и полнота. Их можно рассчитать на основании таблицы 1.1.

Таблица 1.1 – Таблица ошибок

		Истинное значение	
		Positive	Negative
Предсказание	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

а) True Positive TP(c): было выдвинуто предложение о наличии объекта класса C и фактически существовал объект класса C;

б) False Positive FP(c): было выдвинуто предложение о наличии объекта класса C и фактически объекта класса C не существовало;

в) True Negative TP(c): было выдвинуто предложение об отсутствии объекта класса C и фактически не существовал объект класса C;

г) False Negative FP(c): было выдвинуто предложение об отсутствии объекта класса C и фактически объект класса C существовал;

Точность ($Precision$) — это доля правильных предсказаний относительно их общего числа:

$$Precision = \frac{TP(c)}{TP(c) + FP(c)} \quad (1)$$

Полнота ($Recall$) измеряет, насколько хорошо детектор находит все положительные образцы. Например, находить 80% от всех возможных положительных предсказаний.

$$Recall = \frac{TP(c)}{TP(c) + FN(c)} \quad (2)$$

1.1.2 График Точности и Полноты

График Precision и recall (PR) на рисунке 1.1 сочетает в себе точность и полноту для каждого из порогов в каждом предсказании. Чем выше ваша кривая по оси Y, тем выше точность вашей модели. Для того чтобы рассчитать Average Precision (AP) необходимо рассчитать площадь под PR кривой.

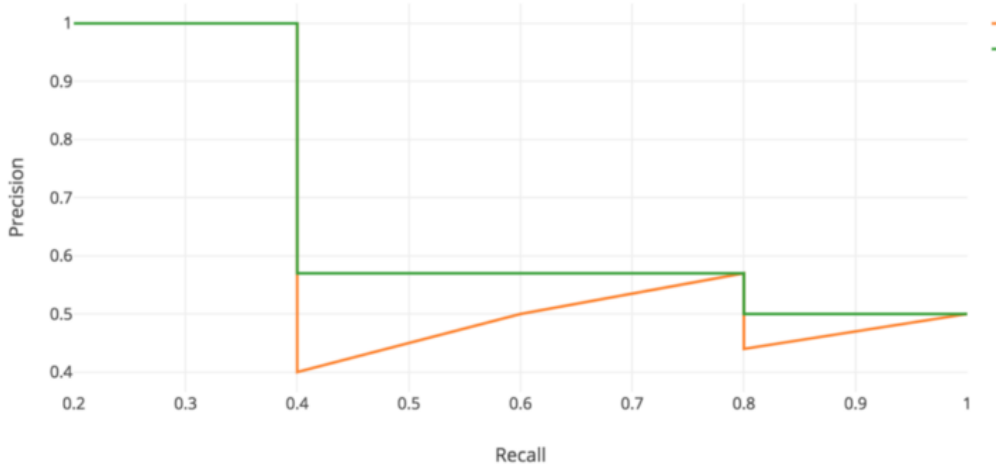

Рисунок 1.1 – График Precision и recall

Прежде чем построить кривую PR, нужно узнать величину шага для расчета значений точности и полноты. Например, введенную в международном соревновании для распознавания изображений Pascal VOC 2007 [1,2] используется усреднение для 11-точечной интерполированной AP – рисунок 1.2.

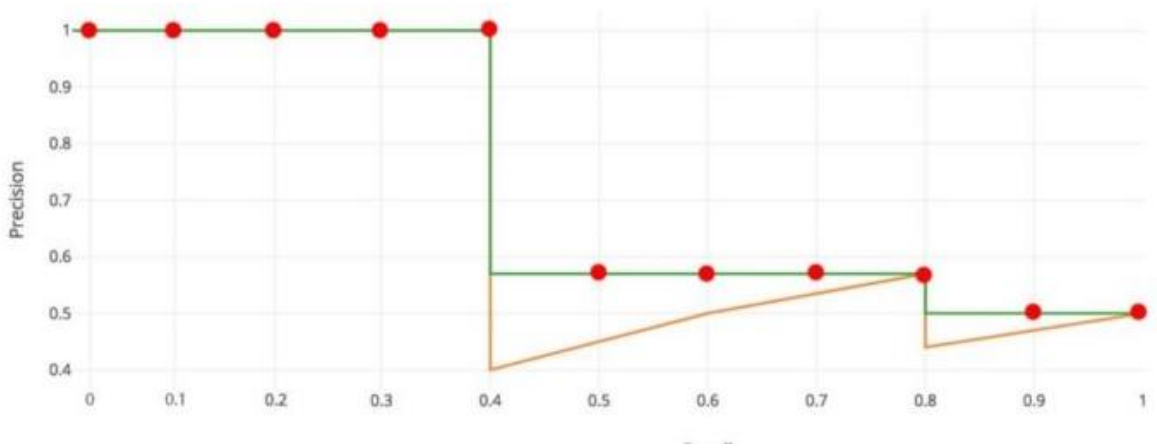

Рисунок 1.2 – График Precision и recall для 11 точек PR

Average precision для 11-точечной интерполированной AP:

$$AP = \frac{1}{11} \sum_{r \in \{0.0,\ldots,1.0\}} p_{int}(r) \tag{4}$$

1.1.3 Метрика степени пересечения областей на изображении «IoU»

Intersection over union (IoU) измеряет перекрытие между двумя областями. Эта величина на используется для определения процента перекрытия истинной и предсказанной областями объекта – рисунок 1.3.

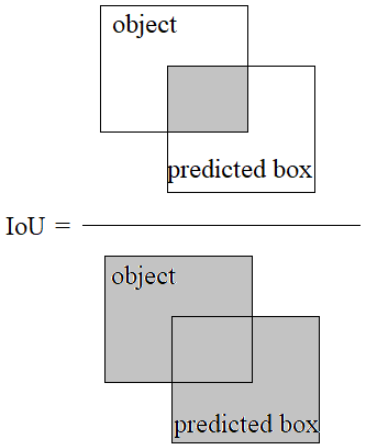

Рисунок 1.3 – Принцип работы IoU

Для таких наборов данных как Pascal VOC и COCO [3] для IoU изначально устанавливается минимальный порог 0.5. Кандидаты-регионы, не прошедшие порога, считаются ошибочными.

$$\mathrm{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{5}$$

1.1.4 Метрика оценки качества ранжирования «mAP»

Mean average precision (mAP) для обнаружения объекта – это среднее значение AP, рассчитанное по каждому из классов. Также важно отметить, что для некоторых наборов данных AP и mAP взаимозаменяемы.

$$mAP = \frac{1}{classes} \sum_{c \in classes} AP(c) \tag{6}$$

Существуют различные методы подсчета AP. Некоторые из таких методов устанавливают пороговое значение ограничивающих рамок на разных IoU:

а) AP с IoU = 0,50: 0,05 – 0,95. Начиная с IoU от 0,5 до 0,95 AP вычисляется с шагом 0,05 и усредняется. Таким образом AP вычисляется при 10 разных IoU и используется для определения качества локализации детектора;

б) AP@IoU=0.5 (метод вычисления описанный выше);

в) AP@IoU=0.75 (IoU ограничивающих рамок должен быть больше 0.75);

В COCO объекты малого размера преобладают над объектами большего размера. Приблизительно 41% объектов имеют малый размер (площадь меньше 32^2 пикселя), 34% объектов имеют средний размер (области между 32^2 и 96^2 пикселями) и 24% большой (область больше 96^2 пикселей)

Методы вычисляют AP по размеру детектированного объекта:

а) AP_{small}: AP для малых объектов: область меньше 32^2 пикселя;

б) AP_{medium}: AP для средних объектов: области между 32^2 и 96^2 пикселями;

в) AP_{large}: AP для больших объектов: область больше 96^2 пикселей;

1.2 Детектирование объектов на основе сверточных нейронных сетей YOLO

Существует два основных подхода в задачи локализации объектов на изображении – one stage detectors и two stage detectors. Идея первого подхода состоит в том, что задача поиск регионов и классификация происходит одновременно и рассматривается, как задача регрессии, в том время как работу второго подхода можно разделить на две части: первая – найти регионы-кандидаты на картинке, которые соответствуют объектам, а вторая – провести классификацию для каждой области, чтобы определить какой именно объект там содержится [4,5].

1.2.1 Архитектура сети YOLO

You only looked once (YOLO) – one stage detector, один из первых скоростных детекторов, позволяющий осуществлять детектирование в режиме реального времени. Он был разработан Джозефом Редмоном. На момент первой публикации (2016 г.) [6,7] в сравнении с такими подходами, как Faster R-CNN [8,9] и DPM [10], модель YOLO немного проигрывала в точности mAP на датасете PASCAL VOC 2007, однако существенно превосходила в скорости детектирования все ранее существующие подходы, и могла проводить детектирование в режиме реального времени. В других моделях того периода в основном использовался метод скользящего окна по всему изображению (DPM – модели деформируемых деталей). Суть метода заключается в том, что существует окно, которое скользит по изображению, далее составляется градиентная гистограмма и на ее основе происходит предсказание объекта классификатором. Модель R-CNN использовала метод предложения региона. Этот метод сначала генерировал потенциальные кандидаты-регионы, затем для них был запущен классификатор и применена постобработка для удаления дубликатов обнаружений и уточнения ограничивающих рамок.

Первой частью модели YOLO является сверточная нейронная сеть классической архитектуры, применяемая для решения задачи классификации изображения. Ее называют базовой сетью [11]. После базовой сети идут 2 полносвзязных слоя рисунок 1.4, которые отвечают за формирование предсказаний ограничивающих рамок, потенциально содержащих объекты, а также определяют вероятности принадлежности этих объектов к заданным классам.

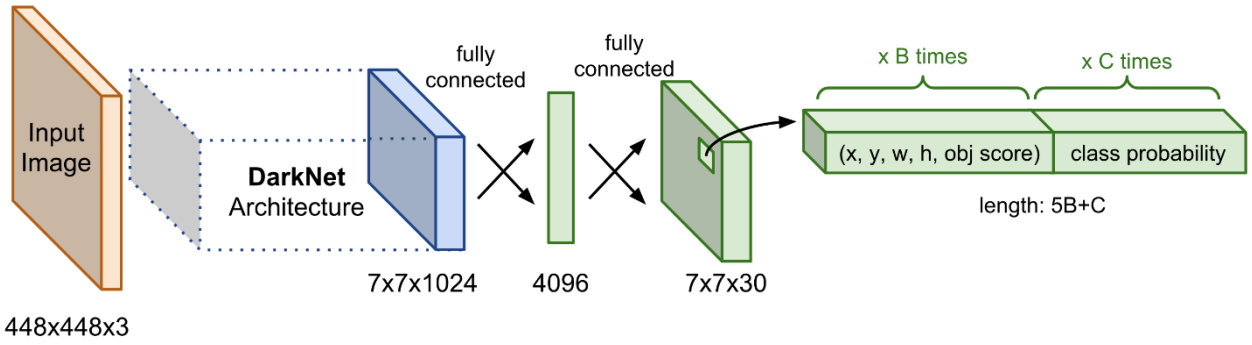

Рисунок 1.4 – Архитектура детектора YOLO

Принцип работы:

Используется предварительно обученная базовая СНС Darknet [7] для классификации входных изображений. На входное изображение накладывается сетка S × S. Если центр объекта попадает в ячейку, эта ячейка «отвечает» за обнаружение существования этого объекта. Каждая ячейка предсказывает местоположение ограничивающих рамок, показатель достоверности и вероятность класса объекта, обусловленную наличием объекта в ограничивающем прямоугольнике рисунок 1.5.

а) Координаты ограничивающей рамки определяются набором из 4 значений (x, y – координаты центра, ширина – w, h - высота), где x и y задаются со смещением местоположение ячейки. Кроме того, x, y, w, h нормализуются по ширине и высоте изображения, таким образом, что все значения лежат в интервале между (0, 1].

б) Показатель достоверности указывает на вероятность того, что ячейка содержит объект: вероятность наличия объекта умножается на его IoU.

в) Если ячейка содержит объект, она предсказывает вероятность того, что этот объект принадлежит каждому классу $c_i, i = 1, k$. На этом этапе модель предсказывает только один набор вероятностей классов на ячейку, независимо от количества ограничивающих рамок.

г) Каждое изображение содержит ограничивающие рамки S×S×B, каждый из которых соответствует 4-м координатам местоположения, 1 значению достоверности и K условных вероятностей для классификации объектов.

Суммарные значения прогнозирования для одного изображения составляют S×S×(5B+K), что является тензорной формой конечного сверточного слоя модели.

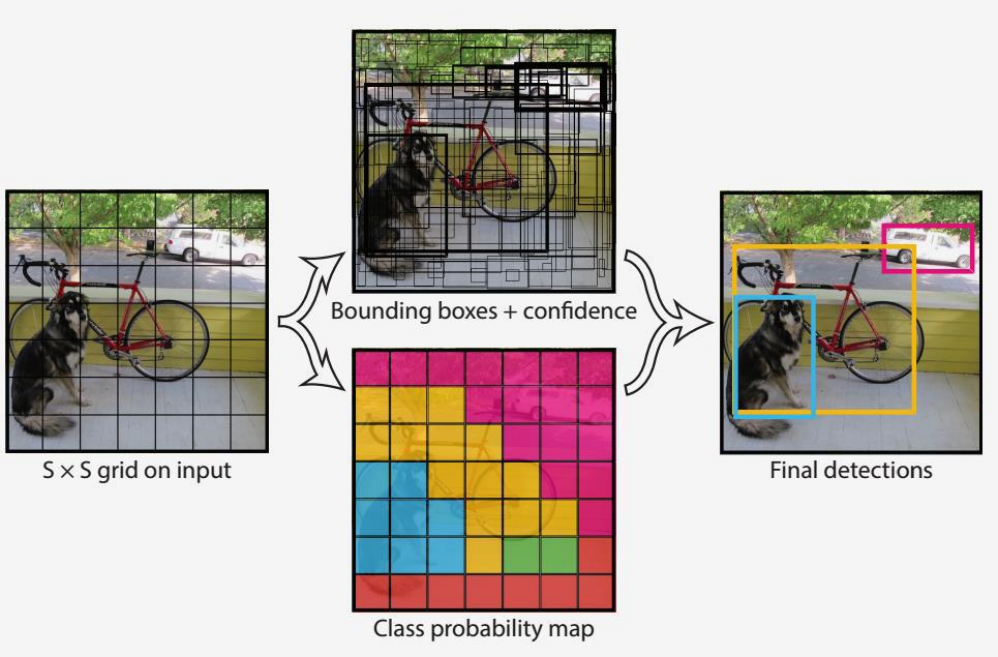

Рисунок 1.5 – Принцип работы YOLO

Далее используя алгоритм Non Maximum Suppression выбираются наиболее подходящие кандидаты – регионы.

Функция потерь состоит из двух частей: функции потерь локализации для прогнозирования смещения ограничивающей рамки и функции потерь классификации для условной вероятности класса. Обе части вычисляются как сумма квадратов ошибок. Два масштабируемых параметра используются для управления показателями того, насколько важно учитывать вклад ошибок предсказаний координат ограничивающей рамки (λ_{coord}) и насколько необходимо снизить вклад ошибку для ограничивающих рамок, не включающих в себя объектов (λ_{noobj}). В оригинальной статье [7] модель устанавливает λ_{coord} = 5 и λ_{noobj} = 0.5.

$$L_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \quad (7)$$

$$L_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left(1_{ij}^{obj} + \lambda_{coord}\left(1 - 1_{ij}^{obj}\right)\right) \left(C_{ij} - \hat{C}_{ij}\right)^2 + \sum_{i=0}^{S^2} \sum_{c \in C} 1_i^{obj} \left(p_i(c) - \hat{p}_i(c)\right)^2 \quad (8)$$

$$L = L_{cls} + L_{loc} \quad (9)$$

где 1_i^{obj} – функция индикатор, сигнализирующая о наличии ячейки в объекте;

1_{ij}^{obj} – указывает, отвечает ли j ограничивающая рамка ячейки i за объект;

C_{ij} – показатель достоверности ячейки i ;

$\widehat{C_{ij}}$ – предсказанный показатель достоверности;

C – набор всех классов;

$p_i(c)$ – условная вероятность того, содержит ли ячейка объект класса $c \in C$;

$\widehat{p_i}(c)$ – предсказанная условная вероятность класса;

Функция потерь только штрафует ошибку классификации, если функция индикатор, информирующий о наличии ячейки в объекте, 1_i^{obj} = 1. Также ошибку координат ограничивающей рамки штрафуется, если этот предиктор указывает, отвечает ли *j*-ю ограничивающую рамки ячейки i за объект, 1_{ij}^{obj} = 1.

Как одноступенчатый детектор объектов, YOLO является сверхбыстрым, однако плохо распознает объекты неправильной формы или группу небольших объектов из-за небольшого числа кандидатов-регионов.

1.2.2 Архитектура сети YOLOv2

YOLOv2 является улучшенной версией YOLO вышедшей в 2017 году [12]. Для повышения точности и скорости детектирования к YOLO применяются различные модификации:

а) Добавление Batch Normalization на всех сверточных слоях;

б) Использование изображений более высокого разрешения для обучения базовой CHH;

в) Для того чтобы улучшить детектирование объектов, которые находятся рядом друг с другом YOLOv2 отказывается от полносвязных слоев и заранее определяет базовые прямоугольники (якоря) путем кластеризации исходных данных алгоритмом K-средних: в отличие от faster R-CNN, которая использует отобранные вручную размеры якорей, YOLOv2 выполняет кластеризацию тренировочных данных, чтобы найти наиболее подходящие размеры якорей и их количество. Метрика расстояния рассчитывается при помощи значения IoU:

$$d(x, c_i) = 1 - IoU(x, c_i), i = 1, \dots, k \quad (10)$$

где x – это истинная ограничивающая рамка;

c_i - один из центроидов. Наилучшее количество центроидов k можно выбрать, например, методом локтя;

г) YOLOv2 формулирует предсказание ограничивающей рамки таким образом, чтобы она не слишком сильно расходилось с местоположением центра предсказанного объекта. Если предсказание местоположения ограничивающей

рамки может сместить ее в любую часть изображения, как например в подходе regional proposal network [8, 9], обучение модели может стать нестабильным. Пусть существует якорь размера (p_w, p_h,) в ячейке сетки с ее верхним левым углом в (c_x, c_y), модель предсказывает смещение и масштаб (t_x,t_y, t_w, t_h), а соответствующая предсказанная ограничивающая рамка b имеет центр (b_x, b_y) и размер (b_w, b_h) – рисунок 1.6. Доверительный балл – это сигмоида (σ) выхода t_0.

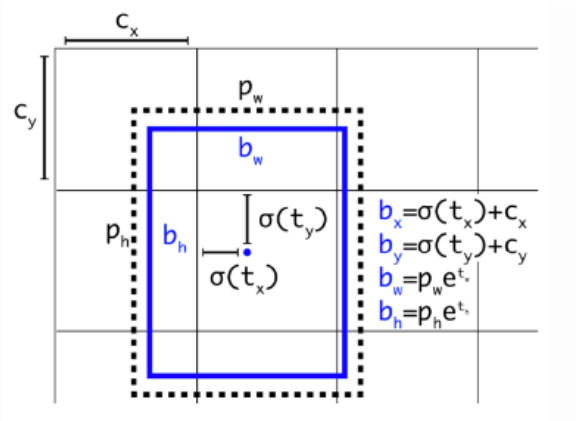

Рисунок 1.6 – Предсказание ограничивающей рамки в YOLOv2

$$b_x = \sigma(t_x) + c_x, \quad (11)$$

$$b_y = \sigma(t_y) + c_y, \quad (12)$$

$$b_w = p_w e^{t_w}, \quad (13)$$

$$b_h = p_h e^{t_h}, \quad (14)$$

$$\sigma(t_o) = pr(object) * IoU(b, object) \quad (15)$$

д) YOLOv2 добавляет слой passthrough, чтобы перенести полезные признаки ранних слоев на последний выходной слой сети. Это приводит к увеличению точности на 1%.

е) Многомасштабное обучение: для того чтобы обучить модель быть устойчивой к входным изображениям различных размеров, новый размер входного измерения случайным образом меняет разрешение каждые 10 эпох. Поскольку сверточные слои YOLOv2 уменьшают размер входного сигнала в 32 раза, то вновь отобранный размер кратен 32.

ж) Облегченная базовая модель. Чтобы увеличить скорость работы сети YOLOv2 использует облегченную базовую модель DarkNet – 19, которая имеет 19 сверточных слоев.

1.2.3 Архитектура сети YOLOv3

YOLOv3 одно из последних обновлений архитектуры YOLO [13]. Модель состоит из 106-ти сверточных слоев. Особенностью v3 является то, что обнаружение происходит в трех различных масштабах. В YOLOv3 обнаружение выполняется путем применения ядер обнаружения 1×1 на картах признаков трех разных размеров в трех разных местах сети – рисунок 1.7.

Размер ядра обнаружения составляет 1×1×(B×(5+ C)). Здесь B – количество якорей, которые может предсказать ячейка на карте признаков, 5 – для 4 координат ограничивающей рамки и значения достоверности, C – количество классов. Так на обученном датасете COCO, B = 3 и C = 80, поэтому размер ядра равен 1×1×255.

Входное изображение сжимается на первых 81 сверхточных слоях, таким образом, что на 81 слое шаг сжатия будет равен 32. Если у нас есть изображение разрешения 416×416, то результирующая карта признаков будет иметь размер 13×13. Первое детектирование производится на 82 слое, что дает нам карту признаков размера 13×13×255.

Затем карту объектов из слоя 79 пропускают через несколько сверточных слоев, далее она будет увеличена в 2 раза до размеров 26×26. Затем эта карта признаков будет объединена по глубине с картой признаков из слоя 61 и комбинированные карты признаков снова пропускаются через несколько сверточных слоев размером 1×1, чтобы объединить признаки раннего 61 слоя. Затем второе обнаружение производится на 94 слое, что дает карту признаков размером 13×13×255.

Аналогичная процедура повторяется снова, когда карта признаков из слоя 91 пропускается далее по сверточным слоям, и объединяется с картой признаков из слоя 36. Как и раньше, несколько сверточных слоев размером 1×1 следуют друг за другом, чтобы слить информацию с предыдущего 36 слоя. Финальное предсказание получается на 106-м слое, получая признаков объектов размером 52×52×255.

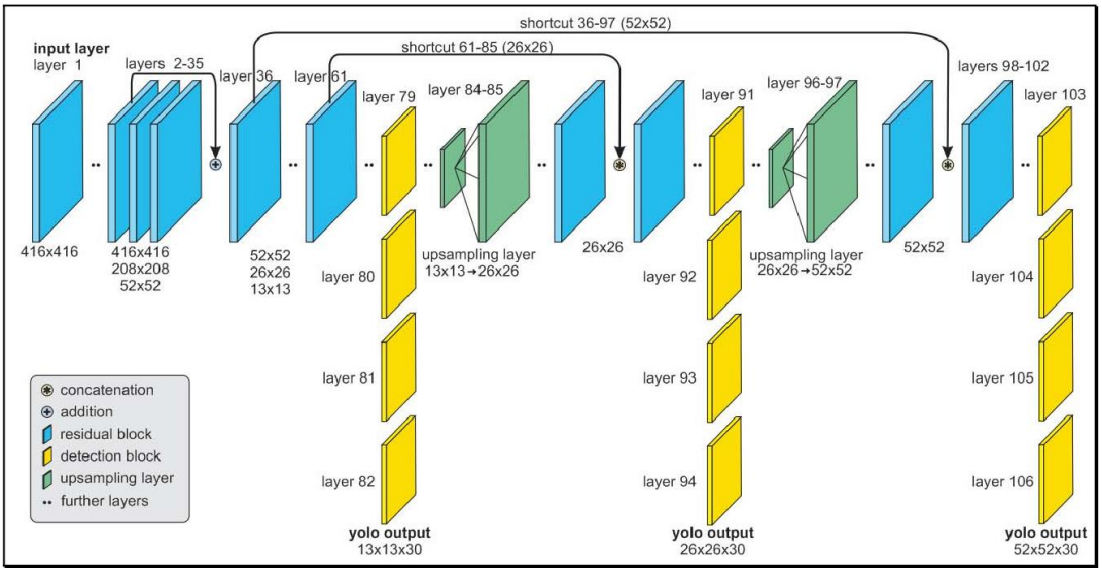

Рисунок 1.7 – Архитектура детектора YOLOv3

Также в YOLOv3 были внесены следующие изменения:

а) YOLOv3 предсказывает оценку достоверности для каждой ограничивающей рамки с помощью логистической регрессии, в то время, как YOLO и YOLOv2 используют сумму квадратов ошибок для обучения. Линейная регрессия прогноза смещения приводит к уменьшению mAP.

б) При прогнозировании уверенности класса YOLOv3 использует несколько независимых логистических классификаторов для каждого класса, а не один слой softmax.

в) Наличие урезанной модели Yolov3 – Yolov3-tiny. Она состоит из меньшего количества слоев и делает предсказания двух разных размеров 13×13и 26×26. Имеет меньшую точность и хуже распознает маленькие объекты, однако может выдавать большие значения обработки кадров в секунду и может быть использована на «слабых» компьютерах и мобильных устройствах.

В целом YOLOv3 работает точнее и быстрее, чем архитектура SSD [14,15,16], и хуже по точности, чем RetinaNet [17], но в 3,8 раза быстрее.

1.3 Детектирование объектов на основе сверточной нейронной сети RetinaNet

RetinaNet – это одноступенчатый детектор, состоящий из базовой сети и двух специфичных для конкретных задач подсетей. Базовая сеть отвечает за вычисление карт признаков по всему входному изображению и представляет собой автономную сверточную сеть. Первая подсеть выполняет классификацию на выходе базовой сети, вторая подсеть выполняет поиск ограничивающей рамки. RetinaNet использует такие подходы как Focal loss и Featurized Image Pyramid, изображенные на рисунках 1.8 и 1.9.

1.3.1 Функция потерь Focal loss

Focal loss – это одна из модификаций cross entropy loss [18], она понижает вклад в обучение случаев, содержащих в себе только фон (easy negative) и наоборот фокусируется на случаях присутствия объекта или его маленькой части (hard negative).

$$FL(p_t) = -a(1 - p_t)^{\gamma} \log p_t \quad (16)$$

где a – балансирующий параметр, который уравновешивает вклад положительных и отрицательных примеров;

γ – фокусирующий параметр, который уравновешивает вклад easy negative и hard negative;

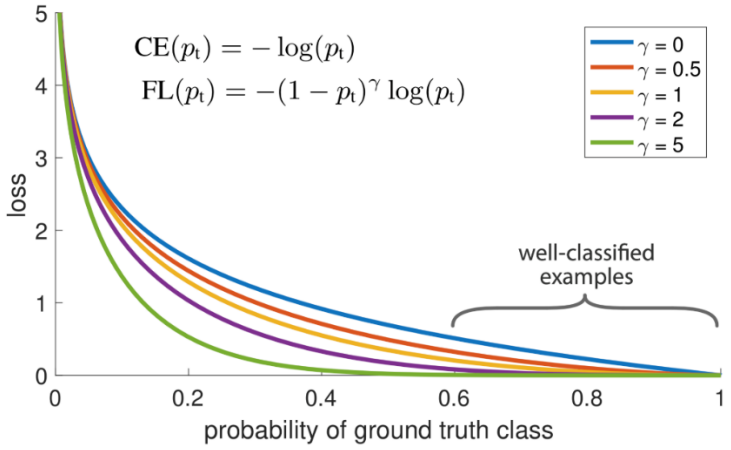

Рисунок 1.8 – График Focal loss при различных значениях γ

1.3.2 Функциональная пирамидальная сеть

Featurized Pyramid Network (FPN) [17] – это пирамидальная структура для извлечения карт признаков разных масштабов, полученных при сжатии размера входного изображения для дальнейшего их объединения с целью детектирования больших и малых объектов.

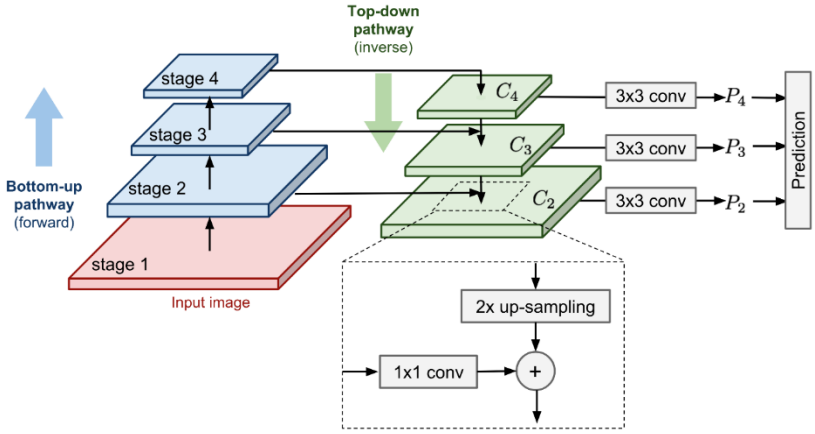

Рисунок 1.9 – Структура Featurized Image Pyramid

Путь снизу-вверх использует ResNet [19] для построения пути снизу-вверх. Он состоит из нескольких модулей свертки, каждый из которых имеет много слоев свертки. По мере продвижения вверх пространственное измерение уменьшается вдвое (т. е. удваивается шаг). Выходные данные каждого модуля свертки помечаются как C_i и затем используются в нисходящем пути.

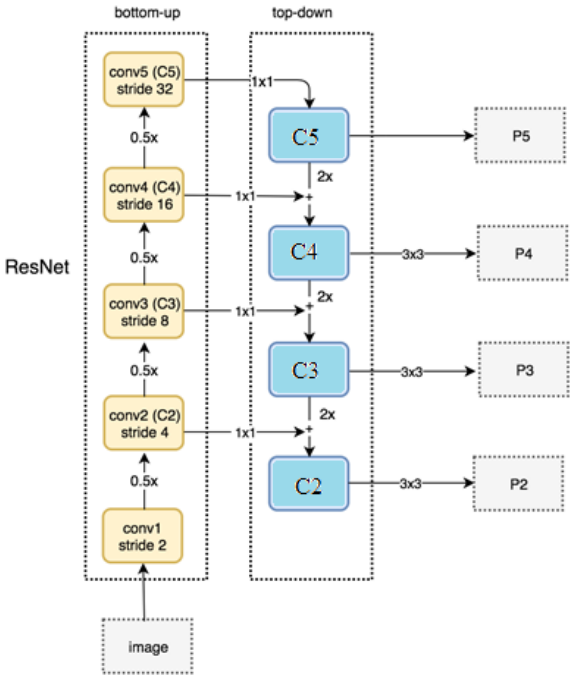

Рисунок 1.10 – Структура Featurized Image Pyramid

При движении сверху-вниз, расширяя карту признаков предыдущего слоя на 2, используется метод ближайших соседей. Далее происходит поэлементное соединение расширенного слоя из пути сверху-вниз с соответствующим слоем из пути сверху-вниз, к которому свертка 1×1 для уменьшения размерности каналов. Объединенные карты признаков называются C_2, C_3, C_4, C_5.

Наконец, к каждой объединенной карте применяется свертка 3×3 для создания финальной карты признаков, которая должна уменьшить эффект сглаживания при расширении. Конечный набор карт признаков P_2, P_3, P_4, P_5, соответствующий C_2, C_3, C_4, C_5, которые соответственно имеют одинаковые пространственные размеры.

1.3.3 Архитектура сети RetinaNet

Базовой сетью для RetinaNet могут служить такие сети как Resnet50 или Resnet101, поверх которой используется структура FPN – рисунок 1.11.

Подсеть классификации предсказывает вероятность наличия объекта в каждой позиции для каждого из *A* якорей *K* классов объектов. Она представляет собой небольшую FCN [17], присоединенную к каждому уровню FPN. Параметры этой подсети являются общими для всех уровней пирамиды.

Одновременно с подсетью классификации работает подсеть регрессии. Она также представляет собой FCN и присоединена к каждому уровню FPN с целью коррекции смещения ограничивающих рамок. Имеет аналогичную подсети классификации структуру за исключением выходного слоя, который

заканчивается 4 линейными выходами для каждой позиции ограничивающих рамок. Для каждого из якорей A для каждого пространственного местоположения эти 4 выхода предсказывают относительное смещение между якорем и истинной ограничивающей рамкой.

Так же, как и в архитектуре SSD, обнаружение происходит на всех уровнях пирамиды путем прогнозирования каждой объединенной карты признаков. Поскольку прогнозы используют один и тот же классификатор и блок-регрессор, все они сформированы так, чтобы иметь одинаковое измерение канала d = 256.

На каждом уровне пирамиды присутствует 9 якорей. Размер якорей соответствует областям от 32^2 до 512^2 пикселей на P3 – P7 уровнях соответственно. Существует три соотношения размеров: $\{2^0, 2^{1/3}, 2^{2/3}\}$. Для каждого размера есть три соотношения сторон {1/2, 1, 2}.

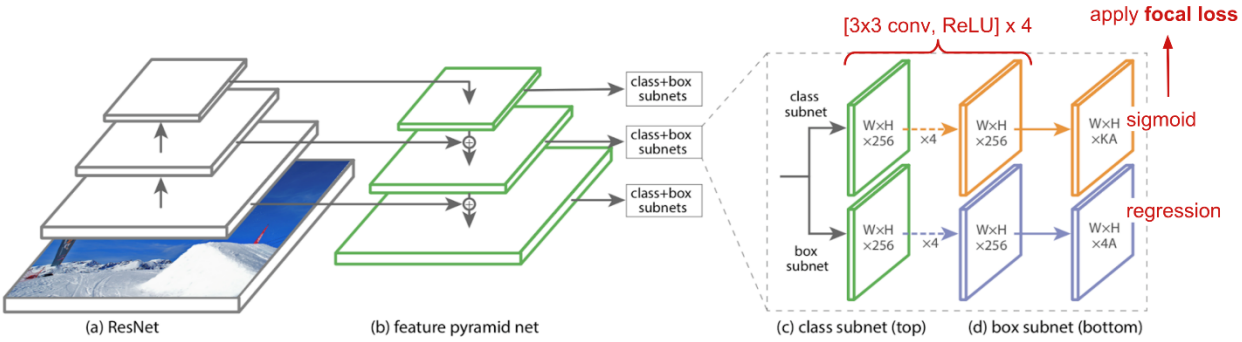

Рисунок 1.11 – Архитектура детектора RetinaNet

1.4 Архитектура сверточной нейронной сети ResNet

ResNet, сокращение от Residual Networks - это классическая нейронная сеть, используемая в качестве основы для многих задач компьютерного зрения. Эта модель стала победителем конкурса ImageNet [20] в 2015 году. Принципиальным достижением ResNet стало то, что оно позволило успешно обучать чрезвычайно глубокие нейронные сети с более чем 150 слоями. До обучения в ResNet очень глубокие нейронные сети были сложными из-за проблемы исчезновения градиентов.

AlexNet [21], победитель ImageNet 2012 и модель, которая положила начало глубокому обучению, имела только 8 сверточных слоев, сеть VGG имела 19, а Inception или GoogleNet – 22 слоя, а ResNet 152 – 152 слоя. В этом блоге мы будем кодировать ResNet-50, который является уменьшенной версией ResNet 152 и часто используется в качестве отправной точки для обучения передаче.

Тем не менее, увеличение глубины сети не работает просто, объединяя слои. Глубокие сети трудно тренировать из-за печально известной проблемы исчезающего градиента – поскольку градиент обратно распространяется на более ранние слои, повторное умножение может сделать градиент чрезвычайно малым. В результате, по мере углубления сети ее производительность становится насыщенной или даже начинает быстро снижаться.

ResNet впервые представила концепцию пропуска соединения – рисунок 1.12. На приведенной ниже схеме показано пропущенное соединение. Фигура слева объединяет слои свертки вместе один за другим. Справа по-прежнему укладываются слои свертки, как и раньше, но теперь также добавляется исходный ввод к выводу блока свертки. Это называется пропуском соединения.

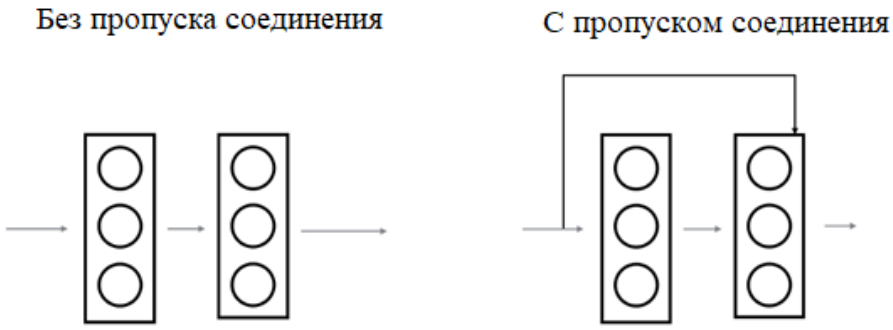

Рисунок 1.12 – Принцип работы подхода с пропуском соединения

Resnet строится на основе простой сети, состоящей из прямых связей, которые вдохновлены философией сетей VGG [22] (слева на рисунке 1.13). Слои свертки имеют 3×3 фильтры и используют следующие правила:

а) Для одной и той же полученной карты признаков слои имеют одинаковое количество фильтров;

б) Если размер карты признаков уменьшается вдвое, количество фильтров удваивается, чтобы сохранить временную сложность каждого слоя;

Также сужение карт признаков выполняется непосредственно с помощью сверточных слоёв с шагом 2. Сеть заканчивается global average pooling layer и полносвязным слоем с функцией активации softmax. Общее количество слоев – 34 рис. 3 (середина).

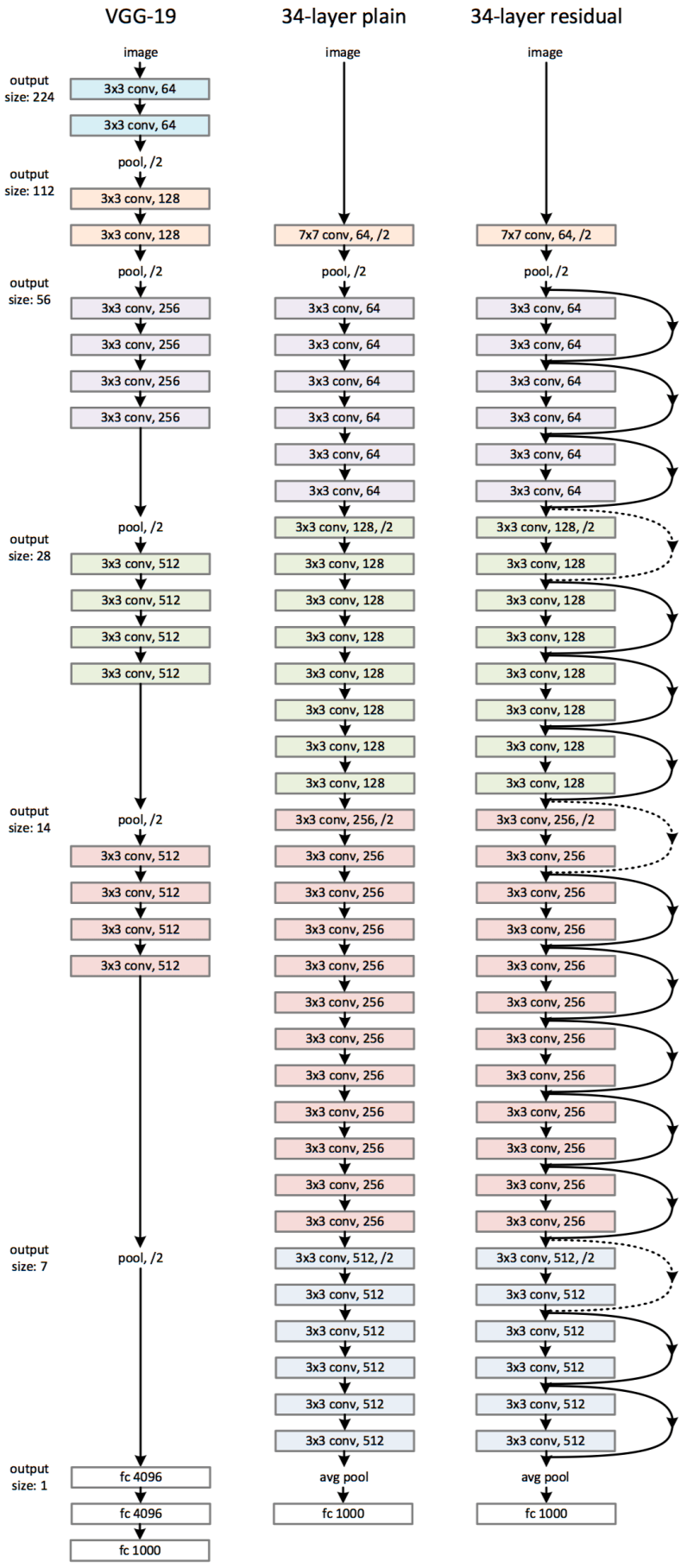

Рисунок 1.13 – Архитектура сетей VGG и ResNet

Структура ResNet основана на структуре простой сети с добавлением подхода пропуска соединения. Идентификационные быстрые соединения $F(x\{W\} + x)$ могут использоваться непосредственно, когда вход и выход имеют одинаковые размерности (быстрые соединения сплошной линией на рис. 2). Когда размерности увеличиваются (пунктирные линии на рис. 2), он рассматривает два варианта:

Быстрое соединение выполняет сопоставление идентификаторов с дополнительными нулями, добавленными для увеличения размерности. Эта опция не вводит никаких дополнительных параметров.

Проекция быстрого соединения в $F(x\{W\} + x)$ используется для сопоставления размерностей (выполнено с помощью 1×1 сверток). Каждый блок ResNet имеет два уровня глубины (используется в небольших сетях, таких как ResNet 18, 34) или 3 уровня (ResNet 50, 101, 152).

Модель ResNet-50 состоит из 5 этапов, каждый с блоком свертки и идентификации. Каждый сверточный блок имеет 3 сверточных слоя, и каждый идентификационный блок также имеет 3 сверточных слоя. ResNet-50 имеет более 23 миллионов обучаемых параметров. Так же существуют версии Resnet-101 и ResNet-151.

На рисунке 1.14 представлены результаты обучения 18-ти и 34-х уровневых сетей простой архитектуры и архитектуры ResNet на датасете ImageNet 2012. ResNet показывает лучшие результаты при большей глубине сети.

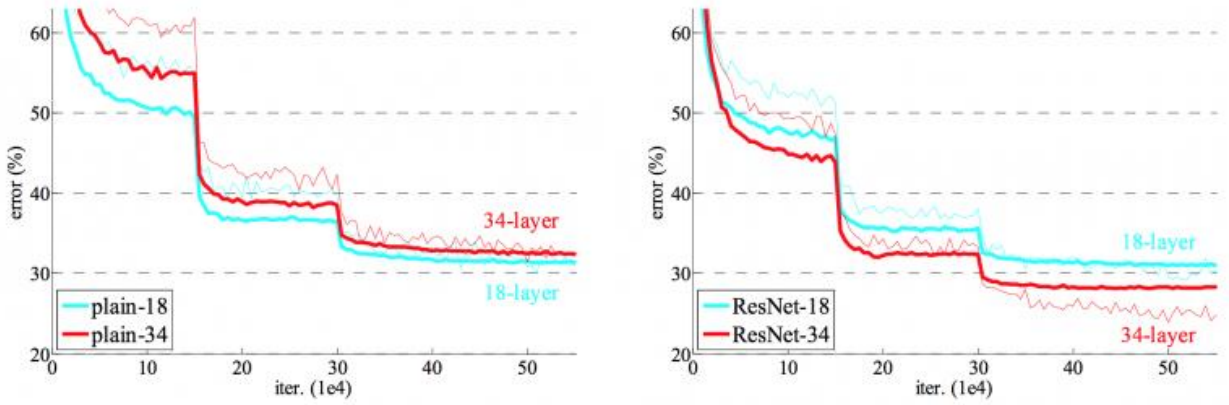

Рисунок 1.14 – Результаты архитектуры ResNet на датасете ImageNet 2012

Сеть ResNet сходится быстрее, чем ее простой аналог. Рисунок 1.15 показывает, что более глубокие ResNet достигают лучших результатов обучения по сравнению с неглубокой сетью.

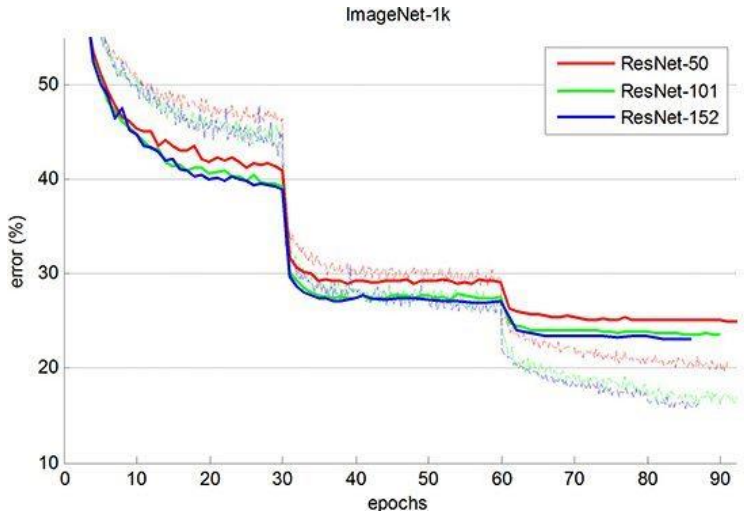

Рисунок 1.15 – Результаты разных архитектур ResNet на датасете ImageNet 2012

ResNet-152 достигает 4,49% в top-5 ошибок валидации. На рисунке 1.16 представлена комбинация из 6 моделей с различной глубиной достигает 3,57% в top-5 ошибок валидации.

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Рисунок 1.16 – ResNet в сравнении с другими архитектурами

1.5 Выводы по главе

В первой главе рассмотрены подходы для задач детектирования и локализации объектов.

Рассмотрены одноступенчатые детекторы семейства YOLO. Для детектирования объектов YOLO использует принцип наложения сетки на входное изображение. Далее для каждой из ячейки сетки предсказывается вероятность наличия в ней объекта и ограничивающие рамки, которые в первой версии определяются при помощи последнего послносвязного слоя, в то время как в последующих версиях используются заранее заданные размеры якорей, которые рассчитываются для различных датасетов, используя алгоритмы кластеризации. Также поздние версии используют YOLO используют подход объединения карт признаков из различных мест сети. Начиная с версии YOLOv2 появляются облегченные базовые модели имеющие меньшее количество сверточных слоев. Это способствует росту производительности детектора и возможности его использования на «слабых» компьютерах и мобильных устройствах.

Рассмотрен одноступенчатый детектор RetinaNet. В качестве базовой сети в RetinaNet используется сеть ResNet, поверх которой используется структура FPN, что позволяет детектировать объекты различного масштаба за счет наличия большого количества якорей на каждом уровне пирамиды, 9 на уровень. Также RetinaNet включает в себя модифицированную функцию потерь focal loss. Используя балансирующий параметр a для уравновешивания вклада положительных и отрицательных примеров, и фокусирующий параметр γ, который уравновешивает вклад easy negative и hard negative.

Рассмотрены методы оценки детекторов. Для определения эффективности работы детекторов используются метрики Average precision или Mean Average Precision. Они вычисляются с учетом процента правильных предсказаний, а также степенью перекрытия предсказанной ограничивающей рамки с истинной. Существует множество методов подсчета Average precision. Далее в работе будет использован метод расчета, который применяется в датасете Pascal VOC 2007.

Рассмотрена архитектура сверточной нейронной сети ResNet. Благодаря подходу с пропуском соединения появилась возможность увеличить количество слоев сети, что привело к увеличению точности.

2 ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ПРИ РЕШЕНИИ ЗАДАЧИ ДЕТЕКТИРОВАНИЯ ОБЪЕКТОВ

2.1 Метод обучения нейронных сетей на основе подхода «Transfer learning»

Подход Transfer learning [23] состоит в том, чтобы применить знания, полученные при решении некоторой задачи, и использовать их на новой, подобной задаче. Например, признаки, полученные при решении задачи классификации автомобилей могут быть использованы для задачи классификации автобусов или других транспортных средств.

Transfer learning часто применяется для задач, в которых изначально содержится слишком мало данных, чтобы обучить полномасштабную модель с нуля.

Наиболее распространенным воплощением Transfer learning в контексте глубокого обучения является следующий подход:

а) Взять сверточные слои ранее обученной модели.

б) Зафиксировать сверточные слои, чтобы избежать изменения любой информации, которую они содержат во время будущих этапов обучения.

в) Добавление нескольких обучаемых слоев поверх зафиксированных слоев. Они научатся превращать старые признаки в прогнозы для нового набора данных.

г) Обучить новые слои в вашем наборе данных.

Так же существует возможность расфиксирования всех слоев модели или ее части, и повторном обучении ее на новых данных с очень низкой скоростью обучения. Это потенциально может привести к значимым улучшениям, путем постепенной адаптации предварительно подготовленных функций к новым данным. Такой подход называется fine-tuning.

В качестве базовых для обучения использовались следующие СНС:

а) RetinaNet – Resnet50, предобученная на датасете COCO

б) YOLOv3 – darknet53, предобученная на датасете imagenet

в) YOLOv3-tiny – darknet19, предобученная на датасете imagenet

2.2 Постановка задачи

Для обучения СНС использовались данные flower recognition [24]. Сбор данных осуществляется на основе данных сервисов flicr, google images, yandex images. Изображения делятся на пять классов: ромашка, тюльпан, роза, подсолнух, одуванчик. Для каждого класса есть около 800 изображений. Они имеют разрешение около 320 x 240 пикселей, а также разные пропорции.

Для обучения СНС использовалось 824 изображения. На рисунке 2.1 представлены изображения обучающей выборки.

Рисунок 2.1 – Примеры изображений обучающей выборки

Для обучения СНС для решения задачи классификации на исходных данных необходимо выделить области с цветами, путем создания аннотации к каждому изображению. Аннотирование изображений осуществлялось при помощи программы LabelImg [25]. При создании наборов данных необходимо выделить все подобласти с цветами, а также указать их принадлежность к классу. Всего было сделано 5 603 аннотаций. На рисунке 2.2 представлен пример аннотации изображения.

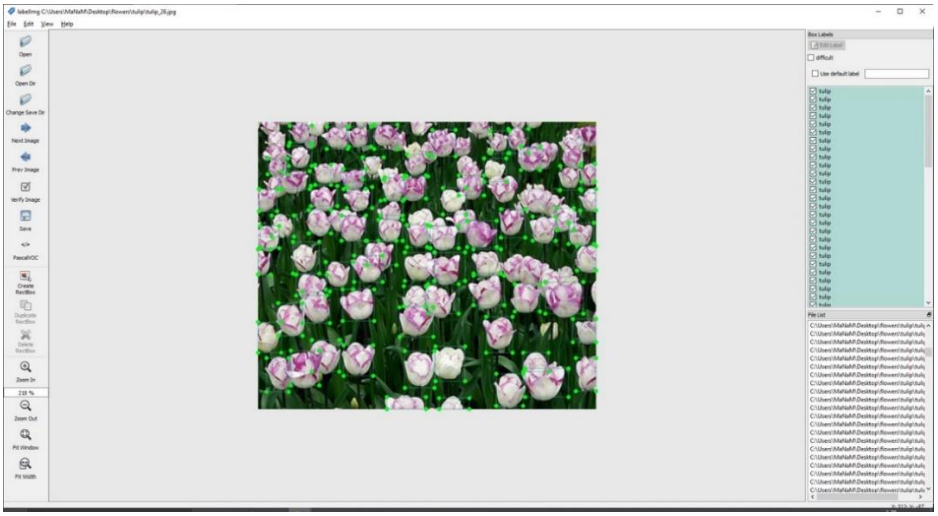

Рисунок 2.2 – Аннотирование изображения в labelimg

Пути к изображениям, координаты выделенных областей и их классы записываются в файлы XML – рисунок 2.2, которые затем преобразуются в формат, пригодный для обучения СНС в соответствующих фреймворках. Каждый XML файл состоит из названия папки, названия изображения, пути до изображения. Также он включает в себя разрешение изображения, глубину цвета, размеры выделенных областей, а именно значения пикселей в левом нижнем и правом верхнем углах областей, принадлежность области к определенному классу

```
- <annotation>
    <folder>chamomile</folder>
    <filename>chamomile_167.jpg</filename>
    <path>C:\Users\MaNaM\Desktop\chamomile\chamomile_167.jpg</path>
  - <source>
      <database>Unknown</database>
    </source>
  - <size>
      <width>320</width>
      <height>233</height>
      <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
      <name>chamomile</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>77</xmin>
        <ymin>32</ymin>
        <xmax>255</xmax>
        <ymax>215</ymax>
      </bndbox>
    </object>
</annotation>
```

Рисунок 2.2 – Аннотация в виде XML файла

2.3 Сравнительный анализ эффективности сверточных нейронных сетей

Для обучения СНС использовались такие фреймворки как imageai [26] – YOLOv3, keras-retinanet [27] – RetinaNet на основе backend Tensorflow на языке Python и Darknet [7] – YOLOv3-tiny на языке C++. Обучение моделей производилось с использованием аппаратного ускорения на GPU Nvidia RTX 2070 super. Оценки скорости работы также производились на указанном GPU. Обучение СНС происходило до тех пор, пока значение average loss не переставало снижаться, то есть последующие эпохи обучения модели не вносили в нее улучшений.

Оценка скорости обработки видео производилась с использованием 10 различных видео с цветами, средняя продолжительность которых равнялась 20 секундам.

В таблицах 2.1, 2.2 представлены значения average precision по каждому классу для каждой архитектуры СНС, mAP, скорость обработки видео в кадрах в секунду (FPS) и объема весовых коэффициентов в МБ.

Таблица 2.1 – AP по каждому из классов

	ромашка	подсолнух	тюльпан	роза	одуванчик
YOLOv3	0.7652	0.7500	0.5728	0.6599	0.6564
RetinaNet	0.7570	0.8078	0.7096	0.6659	0.4381
YOLOv3-tiny	0.3602	0.2643	0.1955	0.3718	0.5727

Таблица 2.2 – Сравнение характеристик СНС

Архитектура СНС	Скорость обработки видео СНС, FPS	mAP, %	Объем весовых коэффициентов СНС, МБ
YOLOv3	35	0.6757	241
RetinaNet	29	0.6809	426
YOLOv3-tiny	65	0.3529	33

Из результатов Таблицы 2.2 можно сделать вывод, что архитектура YOLOv3 уступает RetinaNet по значению mAP, однако затрачивает меньше времени на обработку видео. В свою очередь YOLOv3-tiny намного производительнее по времени обработки видео. На рисунке 2.3 представлен пример детектированных объектов.

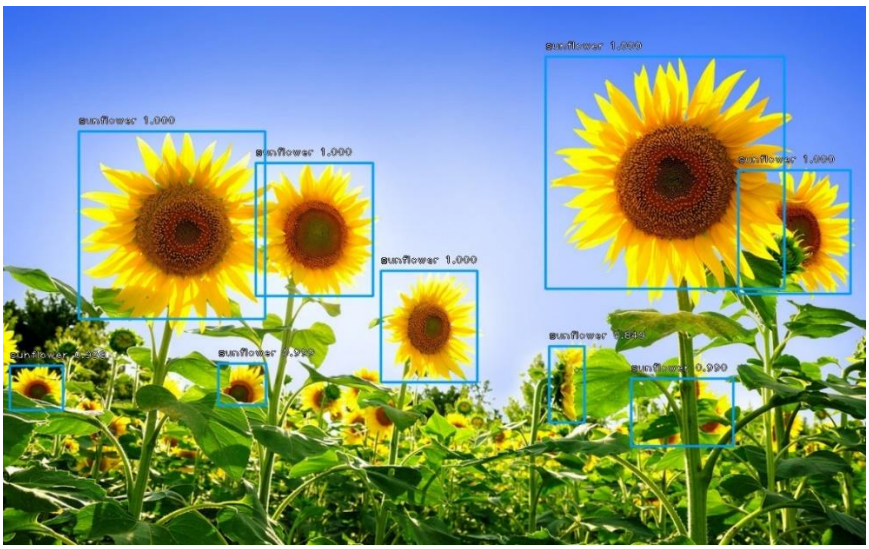

Рисунок 2.3 – Пример детектирования цветов

2.4 Определение расстояния до детектируемых объектов при помощи контроллера Kinect v2

2.4.1 Построение карты глубин

Карта глубины (depth map) – это изображение, на котором для каждого пикселя, вместо цвета, храниться его расстояние до камеры. Карта глубины может быть получена с помощью специальной камеры глубины (например, сенсор Kinect), а также может быть построена по стереопаре изображений. Для каждой точки на одном изображении выполняется поиск парной ей точки на другом изображении. А по паре соответствующих точек можно выполнить триангуляцию и определить координаты их прообраза в трехмерном пространстве. Зная трехмерные координаты прообраза, глубина вычисляется, как расстояние до плоскости камеры.
Парную точку нужно искать на эпиполярной линии. Соответственно, для упрощения поиска, изображения выравнивают так, чтобы все эпиполярные линии были параллельны сторонам изображения (обычно горизонтальны). Более того, изображения выравнивают так, чтобы для точки с координатами (X_0, Y_0) соответствующая ей эпиполярная линия задавалась уравнением $X = X_0$, тогда для каждой точки соответствующую ей парную точку нужно искать той-же строчке на изображении со второй камеры. Такой процесс выравнивания изображений называют ректификацией (rectification). Обычно ректификацию совершают путем ремеппинга изображения и ее совмещают с избавлением от дисторсий. Пример ректифицированных изображений приведен на рисунке 2.4, картинки взяты из базы изображений сравнения различных методов построения карты глубины.

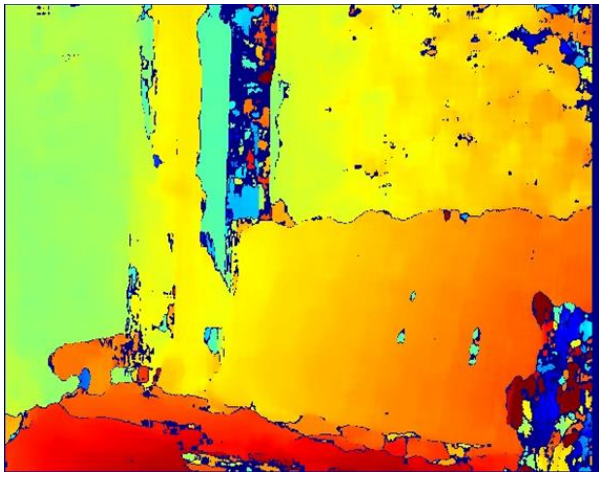

Рисунок 2.4 – Построенная карта глубин по стереопаре

После того как изображения ректифицированы, выполняют поиск соответствующих пар точек. Самый простой способ проиллюстрирован на картинке 4 и состоит в следующем. Для каждого пикселя левой картинки с

координатами (X_0, Y_0) выполняется поиск пикселя на правой картинке. При этом предполагается, что пиксель на правой картинке должен иметь координаты $(X_0 - d, Y_0)$, где d – величина называемая несоответствие/смещение (disparity). Поиск соответствующего пикселя выполняется путем вычисления максимума функции отклика, в качестве которой может выступать, например, корреляция окрестностей пикселей. В результате получается карта смещений (disparity map), пример которой приведен на рисунке 2.5.

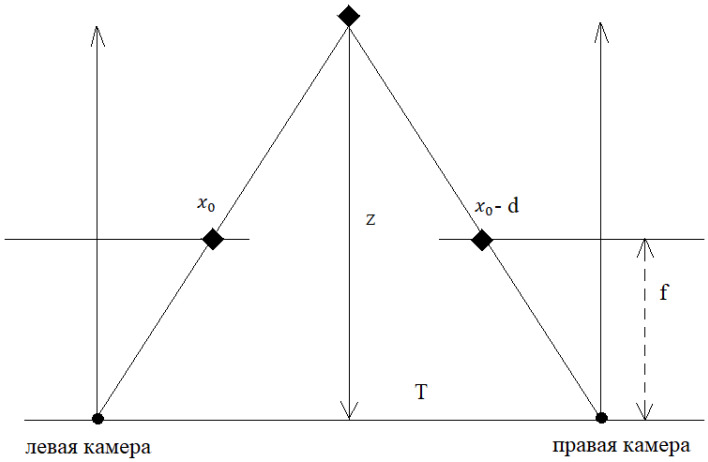

Рисунок 2.5 – Схема работы стереопары

2.4.2 Совмещение детектора для задачи классификации и локализации с игровым контроллером Kinect v2.

Для решения задачи определения расстояния до детектированных объектов было решено использовать стереокамеру Kinect v2.

Kinect – бесконтактный сенсорный игровой контроллер – рисунок 2.6, первоначально представленный для консоли Xbox 360, далее для Xbox One и персональных компьютеров с операционной системой Windows. Kinect включает в себя два датчика глубины, RGB камеру с разрешением 1920 x 1080. Проприетарное программное обеспечение осуществляет полное 3-мерное распознавание движений тела, мимики лица и голоса. Диапазон глубины и программное обеспечение позволяют автоматически калибровать датчик с учётом условий игр и окружающих условий.

Рисунок 2.6 – Kinect v2

Совмещение детектора со стереокамерой Kinect v2 происходило посредством библиотеки PYkinect2 [29] на языке python. Для каждого кадра видеопотока в реальном времени осуществлялось предсказание объектов. Параллельно с детектированием определялась карта глубин кадра. Далее для каждого из предсказанных ограничивающих рамок вычисляется их центр и сопоставляется с картой глубин, в которой уже присутствует вычисленное расстояние до каждого из пикселей изображения. В таблице 3.1 представлены характеристики Kinect v2

Таблица 3.1 характеристики Kinect v2

Разрешение RGB камеры, пикс.	1920 x 1080
Разрешение инфракрасной (ИК) камеры, пикс.	512 x 424
Углы обзора RGB камеры, град.	84.1 x 53.8
Углы обзора ИК камеры, град.	70.6 x 60.0
Диапазон измерений дальности, м.	0.6 — 8.0
Частота съемки RGB камеры, Гц	30
Частота съемки ИК камеры, Гц	30

На рисунке 2.7 представлен принцип и пример работы контроллера Kinect

v2 и архитектуры YOLOv3. Минусами данного подхода являются:

а) слепая зона в 0,5 метра перед контроллером, что объясняется его характеристиками;

б) Определение пикселей, которые будут отвечать за расстояние до детектированного объекта. В разработанной системе ключевыми пикселями являются те, которые находятся в центре ограничивающей рамки. В процессе детектирования центр рамки может сместиться и вместо детектируемого объекта центральные пиксели могут попасть, например, на фон, соответственно рассчитанное расстояние будет не верным. Возможным решением данной проблемы является реализация задачи Instance segmentation [27]. Вместо ограничивающих рамок алгоритмы предсказывают сегментационные маски, покрывающие детектированный объект. Таким образом во время обработки каждого кадра появится возможность определения расстояния до любой части объекта;

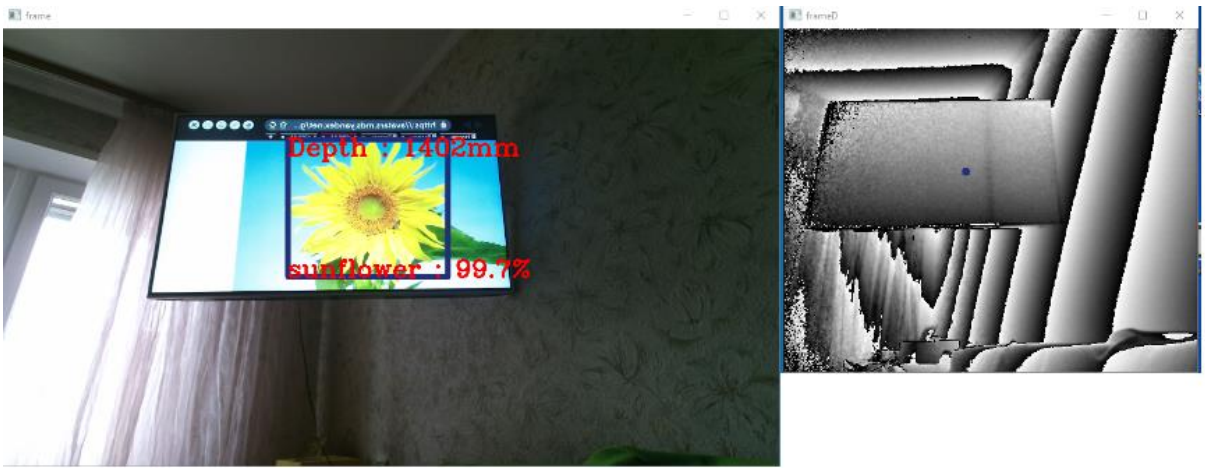

Рисунок 2.7 – Пример нахождения расстояния до детектированного объекта

2.5 Вывод по главе

Во второй главе рассмотрены процессы подготовки данных для обучения, обучения и исследования эффективности сверточных нейронных сетей. Описан способ построения карты глубин, а также работа интеллектуальной системы определения расстояния до локализированных объектов.

Рассмотрен подход передачи модели обучения (transfer learning), когда заранее обученные веса модели, обученные на одной задаче возможно использовать для решения другой, посредством замены последних слоев модели на новые для интерпретирования уже имеющихся признаков в процессе решения задачи.

Для обучения детекторов использовался датасет Flower recognition. Данные включали в себя около 2500 фотографий, из которых для обучения была отобрана треть. Для обучения использовались такие фреймворки как imageai, keras-retinanet и darknet на языках python и C++. Из результатов Таблицы 3 можно сделать вывод, что архитектура YOLOv3 уступает RetinaNet по точности, однако затрачивает меньше времени на обработку видео. В свою очередь YOLOv3-tiny намного производительнее по времени обработки видео.

Рассмотрен подход определения расстояния до детектируемых объектов при помощи контроллера для построения карт глубин. В качестве контроллера был выбран Kinect v2. Он включает в себя датчики глубины и RGB камеры. Проприетарное программное обеспечение включает себя построение карты глубин, распознавание движение тела, мимики лица и голоса.

Во время работы детектора в режиме реального времени при обработке каждого кадра осуществляется детектирование объектов с последующей отрисовкой ограничивающих рамок. Далее происходит поиск центра каждого из ограничивающих рамок и их сопоставление с картой глубины изображения.

ЗАКЛЮЧЕНИЕ

Первым этапом выполнения работы было исследование предметной области. Были рассмотрены архитектуры сверточных нейронных сетей для задачи классификации и локализации YOLO и RetinaNet, а также способы оценки их эффективности.

Вторым этапом выполнения работы был поиск соответствующих фреймворков для обучения на данных о цветах, находящихся в открытом доступе, обучение архитектур сверточных нейронных сетей, а также оценка эффективности распознавания и быстродействия. До начала обучения данные были предварительно обработаны путем аннотирования каждого из изображений. Более точным подходом оказалась архитектура RetinaNet, в то время как обе модели YOLO превосходили по скорости быстродействия.

Последним этапом выполнения работы была разработка интеллектуальной системы определения расстояния по визуальным данным, путем совмещения работы совмещения разных архитектур детекторов, а также бесконтактного игрового контролера Kinect v2. В работе продемонстрирована работа системы и показаны выявленные минусы данного подходы, избавиться от которых можно перейдя от задачи локализации к задаче сегментации изображения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Pascal VOC database. – URL: https://host.robots.ox.ac.uk/pascal/VOC (дата обращения 14.05.2020). – Текст: электронный.

2. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision / Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman // – Jan. 2015. – P. 17. – direct text.

3. COCO dataset. – URL: http://cocodataset.org/#detection-eval (дата обращения 12.05.2020). – Текст: электронный.

4. Going Deeper with Convolutions. – URL: https://arxiv.org/abs/1409.4842 (дата обращения 14.05.2020). – Текст: электронный

5. Deep Residual Learning for Image Recognition – URL: https://arxiv.org/abs/1512.03385 (дата обращения 12.05.2020). – Текст: электронный.

6. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi / You only look once: Unified, real-time object detection// – 2015. – P. 1-10. – direct text.

7. J. Redmon. Darknet: Open source neural networks in C – URL: http://pjreddie.com/darknet/, 2013–2016. (дата обращения 12.05.2020). – Текст: электронный.

8. S. Ren, K. He, R. Girshick, and J. Sun. / Faster r-cnn: Towards real-time object detection with region proposal networks // – Jan. 2016. – P. 1-14. – direct text.

9. R. B. Girshick/ Fast R-CNN // ICCV. – Sep. 2015. – P. 1-9. – direct text.

10. P. F. Felzenszwalb, R. B. Girshick, and D. McAllester / Cascade object detection with deformable part models// In CVPR. – 2010. – P. 1-10. – direct text.

11. Воронов С. В., Мухометзянов Р. Н., Воронов И. В. Обнаружение и распознавание дорожных знаков в реальном времени на мобильных устройствах [Текст] // Автоматизация процессов управления. – Ульяновск: ФНПЦ АО «НПО «Марс», 2018, №2. – С. 105-111.

12. J. Redmon and A. Farhadi/ Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition // (CVPR) 2017 IEEE Conference, – 2017. – P. 1-9. – direct text.

13. Redmon and A. Farhadi / Yolov3: An incremental improvement // – 2018. – P. 1-9. – direct text.

14. Liu W., Anguelov D., Szegedy C. SSD: Single Shot MultiBox Detector [Text] // ECCV 2016. – 2016, Vol.1. – Pp. 21-37. Liu, Wei et al / SSD: Single Shot MultiBox Detector // Lecture Notes in Computer Science (2016) // – 2016. – P. 1-37.

15. T.Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie / Feature pyramid networks for object detection // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pages – 2017. – P. 1-9. – direct text.

16. T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar / Focal loss for dense object detection // – Feb. 2018. – P. 1-10. – direct text.

17. K. He, X. Zhang, S. Ren, J. Sun / Deep Residual Learning for Image Recognition // – Dec. 2015. – P. 1-10. – direct text.

18. A.Krizhevsky I.Sutskever G.E. Hinton / ImageNet Classification with Deep Convolutional Neural Networks // – P. 1-9. – direct text.

19. Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-fei, L. / ImageNet: A Large-Scale Hierarchical Image Database. In IEEE Conference on Computer Vision and Pattern Recognition // – P. 1-10. – direct text.

20. K. Simonyan, A.Zisserman / Very Deep Convolutional Networks for Large-Scale Image Recognition// – Apr. 2015. – P. 1-10. – direct text.

21. Pan, S. J., & Yang, Q / Automated Classification of Lung Diseases in Computed Tomography Images Using a Wavelet Based Convolutional Neural Network // IEEE Transactions on Knowledge and Data Engineering, – 2010. – P. 1-11. – direct text.

22. Flower recognition database – URL: https://www.kaggle.com/alxmamaev/flowers-recognition#102841525_bd6628ae3c.jpg (дата обращения 13.05.2020). – Текст: электронный.

23. labelImg – URL: https://github.com/tzutalin/labelImg дата обращения 14.05.2020). – Текст: электронный.

24. Imageai – URL: https://imageai.readthedocs.io/en/latest/ (дата обращения 14.05.2020 дата обращения 13.05.2020). – Текст: электронный.

25. Keras-retinanet – URL: https://github.com/fizyr/keras-retinanet (дата обращения 14.05.2020 дата обращения 14.05.2020). – Текст: электронный.

26. PyKinectv2 – URL: https://github.com/Kinect/PyKinect2 (дата обращения 14.05.2020 дата обращения 14.05.2020). – Текст: электронный.

27. K. He, G. Gkioxari, P. Dollar, and R. Girshick / Mask r-cnn // – ICCV– Apr. 2017. – P. 1-5. – direct text.