МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЦИФРОВОЙ ЭКОНОМИКИ

Герман Андрей Александрович

РАСПОЗНАВАНИЕ БИОЛОГИЧЕСКИХ ОБЪЕКТОВ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки
09.04.01 Информатика и вычислительная техника,
магистерская программа
«Технологии виртуальной и дополненной реальности»

Владивосток
2020

Автор работы _____
(подпись)

«14» июля 2020 г.

Консультант(ы)

_____ _____
(подпись) (ФИО)

«_____»_____20___г.

Руководитель ВКР с.н.с., к.т.н.
(должность, уч. степень, уч. звание)

_____ М.Г. Александина
(подпись) (ФИО)

«24» июля 2020 г.

Назначен рецензент к.б.н., с.н.с.
(уч.степень, уч. звание)

ШБМ ДВФУ
Калинецкий Александр Марков
(фамилия, имя, отчество)

Защищена в ГЭК с оценкой

_____

Секретарь ГЭК

_____ _____
подпись И.О. Фамилия

«_____» _____ 20___г.

«Допустить к защите»

Заместитель директора ШЦЭ по УВР, к.э.н.

_____ Е.В. Сапрыкина
(подпись) (и. о.ф)

«24» 07 2020 г

УТВЕЖДАЮ
Директор Школы цифровой экономики

_____ И.Г. Мирин
(подпись) (и. о.ф)

«_____»_____20___г.

В материалах данной выпускной квалификационной работы не содержатся сведения, составляющие государственную тайну, и сведения, подлежащие экспортному контролю.

Сапрыкина Е.В.
_____
Ф.И.О.

зам. директора ШЦЭ по УВР
Подпись

Уполномоченный по экспортному контролю
«17» 07 2020 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)
ШКОЛА ЦИФРОВОЙ ЭКОНОМИКИ

ЗАДАНИЕ
на выпускную квалификационную работу

студенту **Герман Андрей Александрович** группы **М9118-09.04.01**

на тему **Распознавание биологических объектов**

утверждённую приказом от « 20 » декабря 2019г. № 124-01-07-08

Руководитель ВКР к.т.н, с.н.с ИАПУ ДВО РАН Алексанина Марина Георгиевна
Консультант: Кленин А.С., технический директор Центра НТИ по нейротехнологиям, виртуальной и дополненной реальности ДВФУ

1. Перечень вопросов, подлежащих к разработке (исследованию):
- создание датасета дальневосточных ларг необходимых для обучения нейронной сети;
- разработать программное обеспечение для обнаружения дальневосточных ларг на основе видеопотока;
- обучить нейронную сеть для распознавания ларг;
- проверить работу обученной модели нейронной сети на архивном видеофайле.

2. Основные источники информации, используемые для разработки темы (учебная и научная литература, нормативно-правовая литература, данные, полученные в ходе преддипломной практики)
Учебная и научная литература по тематике ВКР; информационные ресурсы сети Интернет; данные, полученные от Национального научного центра морской биологии ДВО РАН

Срок предоставления работы «14» 07 2020 г.

Дата выдачи задания «01» 02 2020 г.

Руководитель ВКР к.т.н, с.н.с ИАПУ ДВО РАН (подпись) Алексанина М.Г.

Задание получил (подпись) Герман А.А.

Аннотация

Оптическое зрение является важным компонентом в обнаружении, отслеживании и классификации биологических объектов. Точное обнаружение биологических объектов может помочь в распознавании и классификации объекта на видео, в трекинге его движения, детекцию внешних особенностей и даже понимание поведения отслеживаемого объекта. Однако обнаружение объектов на протяжении десятилетий было сложной задачей так как изображения объектов в реальной среде подвержены влиянию освещения, движения, масштаба, окклюзии.

Цель данной работы является разработка системы распознавания объектов морской биофауны (дальневосточных ларг) с использованием методов компьютерного зрения и глубокого обучения.

Обученную нейронную сеть мы применим для обнаружения объектов морской биофауны на изображении, видео и в режиме реального времени с помощью камеры.

Ключевые слова:

Дальневосточная Ларга; Обнаружение объекта на изображении; Обученная нейронная сеть Yolo; Детекторы объектов;

СОДЕРЖАНИЕ

Введение

Актуальность темы проекта. Дальневосточная нерпа или Ларга – вид тюленей, обитающих в Тихом океане вдоль дальневосточного побережья России. Является малоизученным морским млекопитающим. Последние годы ей активно стало интересоваться Национальный научный центр морской биологии ДВО РАН, так как выяснилось, что в заливе Петра Великого обитает уникальная береговая форма, которая размножается и выкармливает своё потомство исключительно на островах заповедника. Для изучения, в том числе и ларг, в 2017 году была создана лаборатория морских млекопитающих, исследования которой проходят в Приморском океанариуме на о. Русском.

Интерес к данной популяции связан с тревожно низкой численностью. Само ее существование возможно благодаря тому, что рождаются и выкармливаются детёныши на островах, которые охраняются Дальневосточным морским заповедником. За его пределами Ларга - объект зверобойного промысла. Исходя из этого, основная цель биологов – это добиться включения местной особи в Красную книгу. Детальное изучение животного должно помочь доказать его уникальность.

На текущий момент биологи имеют огромное количество данных, получаемых с камер, установленных над островами-лежбищами. Устройства ведут запись круглые сутки. Видео просматриваются людьми с целью поиска «интересных» моментов, подсчёта количества ларг на лежбище, поиском маркированных особей и составления карты посещений геозон которые могут помочь в изучении поведения Ларг (рисунок 1). Это очень медленная работа, которая должна подлежать оптимизации.

Проектная идея выпускной квалификационной работы заключается разработке системы распознавания объектов морской биофауны (дальневосточных ларг) с использованием методов компьютерного зрения и глубокого обучения.

Цель и задачи проектного исследования. Целью магистерской диссертации является разработка программных модулей для классификации и обнаружения морских животных на изображениях и видеофайлах.

Для достижения поставленной цели были поставлены следующие *задачи*:

- создать пользовательский набор данных Дальневосточных Ларг (Dataset) необходимых для машинного обучения;

- обучить нейронную сеть для распознавания Дальневосточных Ларг;

- разработать программу для обнаружения Ларг на изображениях, архивных видеофайлов, а также распознавание объекта в реальном времени.

Теоретическую и методологическую основу выпускной работы составляют представленные исследования в зарубежных и отечественных источниках. Список приведён в конце работы.

Структура этой работы. Выпускная квалификационная работа состоит из введения, трех глав, выводов и обсуждений, списка литературы и приложений. Структура работы отвечает поставленным цели и задачам проектного исследования. Глава 1 также содержит обзор современных систем обнаружения объектов. Глава 2 содержит описание методов, используемых для генерации набора данных и оценки детектора. Глава 3 представляет результат проделанной работы. Наконец, заключение и обсуждение проделанной работы и возможной будущей работы представлены в главе 4.

Глоссарий

AP Average Precision - средняя точность

mAP Mean Average Precision

CNN Convolutional Neural Network - сверточная нейронная сеть

FCNN Fully Connected Neural Network - полносвязная нейронная сеть

FN False Negative - ложноотрицательный

FP False Positive - ложноположительный

IOU Intersection over Union - пересечение между двумя ограничивающими рамками

R-CNN Regions with CNN - региональные сверточные сети

RELU Rectified Linear Unit - усечённое линейное преобразование

R-FCN Region based Fully Convolutional Networks – региональные полностью сверточные сети

RPN Region Proposal Network – региональное гипотеза

SSD Single Shot MultiBox Detector – детектор одиночного выстрела

TP True Positive – истинный позитив

YOLO You Only Look Once -ты только один раз посмотри

1 Распознавание биологических объектов: постановка задачи и обзор существующих решений

1.1 Описание предметной области

Дальневосточная нерпа или Ларга – вид тюленей, обитающих в Тихом океане вдоль дальневосточного побережья России. Является малоизученным морским млекопитающим. Последние годы ей активно стало интересоваться Национальный научный центр морской биологии ДВО РАН, так как выяснилось, что в заливе Петра Великого обитает уникальная береговая форма, которая размножается и выкармливает своё потомство исключительно на островах заповедника. Для изучения, в том числе и ларг, в 2017 году была создана лаборатория морских млекопитающих, исследования которой проходят в Приморском океанариуме на о. Русском.

Интерес к данной популяции связан с тревожно низкой численностью. Само ее существование возможно благодаря тому, что рождаются и выкармливаются детёныши на островах, которые охраняются Дальневосточным морским заповедником. За его пределами Ларга - объект зверобойного промысла. Исходя из этого, основная цель биологов – это добиться включения местной особи в Красную книгу. Детальное изучение животного должно помочь доказать его уникальность.

На текущий момент биологи имеют огромное количество данных, получаемых с камер, установленных над островами-лежбищами. Устройства ведут запись круглые сутки. Видео просматриваются людьми с целью поиска «интересных» моментов, которые могут помочь в изучении поведения Ларг (рисунок 1). Это очень медленная работа, которая подлежит оптимизации.

Рисунок 1.1 Дальневосточный морской заповедник. Лежбище «Кентавр» на островах архипелага Римского-Корсакова

Так же у биологов имеется большая база фотографий меченных Ларг. Метка представляет собой идентификационный номер на теле на боку (рисунок 2). Характеристики (пол, возраст, вес и т.д.) каждого меченного животного записаны (рисунок 3). Все это делалось с целью слежения за особью в течение всей ее жизни. Такой метод очень неудобен, так как не всегда удаётся распознать метку на фото или видео. Чаще всего учёные прибегают к сравнению набора фотографий, с целью выделения отдельной особи с помощью рисунка пятен на теле. Это очень долгий и неэффективный метод.

Рисунок 1.2 Меченная Ларга

Цель данной работы – является разработка программных модулей для классификации и обнаружения морских животных на изображениях и видеофайлах, которая облегчит и продвинет работу учёных по их изучению.

1.2 Обзор существующих методов решения

Как упоминалось во введении, обнаружение визуальных объектов - это проблема локализации и классификации объектов на изображениях.

1.2.1 Обзор литературы

В этом разделе представлен обзор вкладов в решение проблемы обнаружения объектов. Некоторые из ссылок, включённых в этот раздел, используют общие методы для повышения способности их архитектуры к обучению. Два примера таких методов - пакетная нормализация и выпадение. В центре внимания этого доклада представление различных архитектур, специфичные для обнаружения объектов, поэтому общие методы, такие как пакетная нормализация и выпадение, упоминаются лишь кратко. Метод исключения состоит в том, чтобы во время обучения сети удалить узлы с вероятностью. Во время тестирования используются все узлы, но веса от

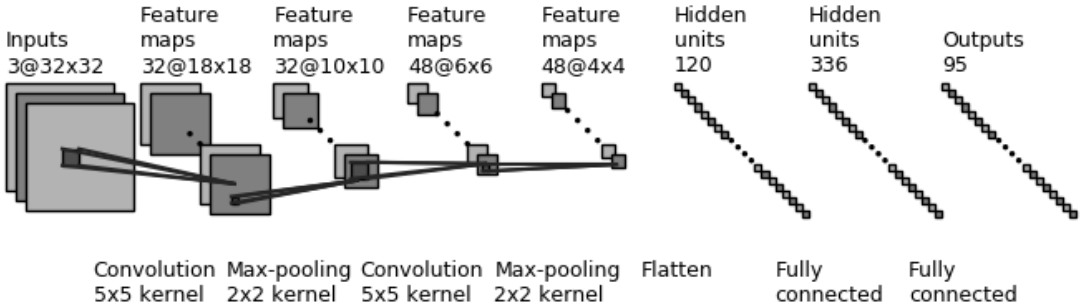

Рисунок 1.4 Визуализация простой архитектуры нейронной сети. Генерируется путём изменения кода из [46]

Метод исключения состоит в том, чтобы во время обучения сети удалить узлы с вероятностью. Во время тестирования используются все узлы, но весовые коэффициенты из затем умножаются на вероятность, использованную для отсева. Этот процесс приводит к тому, что веса не слишком сильно зависят друг от друга, что приводит к более устойчивому процессу обучения [26]. Краткое описание нормализации партии обсуждается позже, см. Раздел «Дополнительные работы для YOLO».

1.2.2 Классификация изображений с использованием CNN

Один из первых полных конвейеров, использующих CNN для классификации изображений, был представлен в [14], где авторы представляют архитектуру, способную классифицировать рукописные числа по банковским чекам.

После 1998 года был период, когда было сделано не так много улучшений из-за аппаратных ограничений, но в 2012 году Алексей Крижевский и другие представили свою архитектуру CNN для классификации изображений [13]. Эта работа (AlexNet) рассматривается многими как большой прорыв для CNN, одной из причин этого является его первое место в конкурсе визуального распознавания изображений в широком масштабе ImageNet (ILSVRC) в 2012 году [1]. После AlexNet там было предложено множество других архитектур для классификации изображений, ZFNet [28] и VGG16, представленных в [25], и многие другие, этот тип сетей позднее использовался в качестве строительных блоков в архитектурах, предназначенных для обнаружения объектов.

Рисунок 1.4 показывает простую иллюстрацию того, как строительные блоки из Раздела 2.1 объединяются, чтобы сформировать архитектуру для классификации изображений, LeNet 5 из [14]

1.2.3 CNN для обнаружения объектов

В контексте обнаружения объектов была проделана огромная работа. Много разных способов использования разных архитектур CNN были предложены. На рисунке 2.5 показана временная шкала недавно предложенных архитектур [2, 6, 7, 18, 20–23], где R-CNN и YOLO обозначают «Области в CNN» и «Вы смотрите только один раз» соответственно.

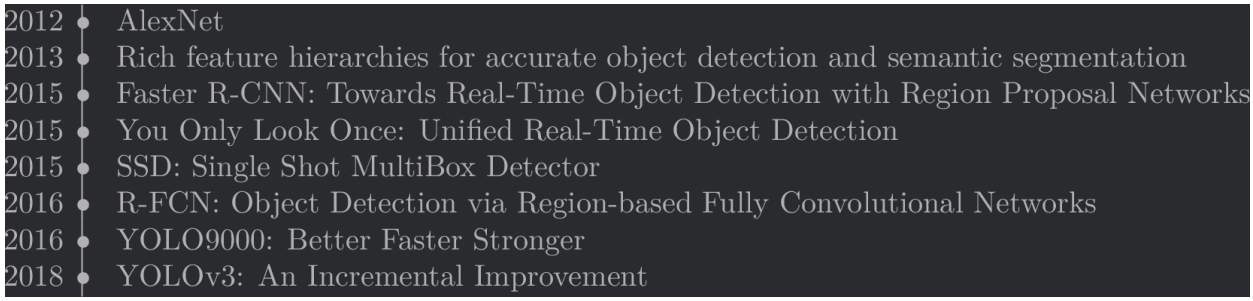

Рисунок 1.5: Временная шкала над работой, способствующая архитектуре обнаружения объектов и одному классификатору (AlexNet).

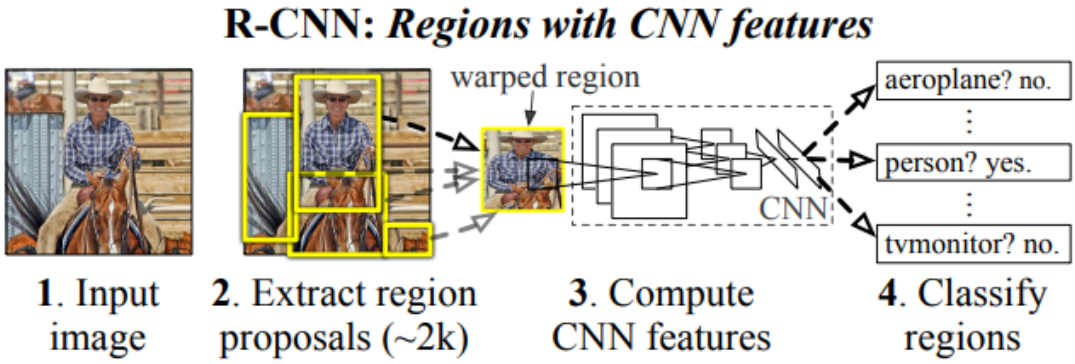

Рисунок1.6: Иллюстрация архитектуры R-CNN. Взято из [7]

R-CNN

В статье [7], написанной Tsuing-Yi и другими, представлена архитектура для обнаружения объектов путём объединения CNN для классификации с интересующими регионами. Эта архитектура проиллюстрирована на рисунке 1.6, взятом из оригинальной статьи [7]. Как показано на рисунке 1.6, несколько

фрагментов изображения пропускаются через CNN для определения классификации, это главный недостаток метода, в последующей работе представлены альтернативные методы для уменьшения вычислительных затрат.

Fast R-CNN

Чтобы создать более эффективную архитектуру, чем R-CNN, авторы [6] представили метод, в котором сеть VGG16 из [25] использовалась для извлечения одной карты характеристик для всего изображения. Из этой карты объектов векторы объектов фиксированного размера были извлечены из всех областей интересов с использованием слоёв максимального пула.

Из векторов объектов, извлечённых из каждого RoI, система определяет класс RoI, используя слой softmax, и уточняет местоположение объекта, используя регрессию. Эта архитектура иллюстрируется рисунком 1.7 из [6].

Быстрее R-CNN

Подобно работе, проведённой автором Fast R-CNN, создатель Faster R-CNN [23] больше использовал архитектуру CNN, чем предыдущая работа.

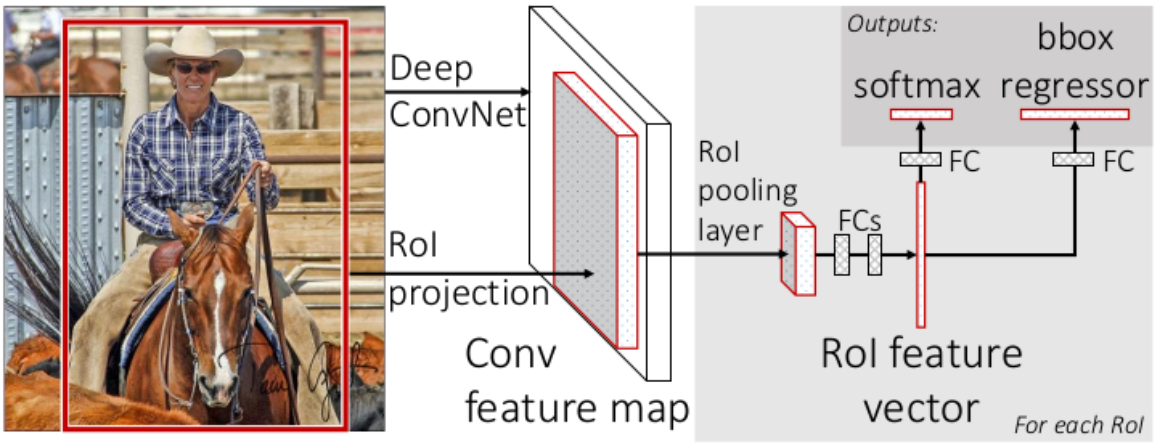

Рисунок 1.7: Иллюстрация архитектуры Fast R-CNN. Взято из [6]

В [23] этой работе авторы представляют архитектуру, которая использует VGG16 в качестве региональной сети предложений (RPN), та же карта характеристик также используется в части оценки объекта конвейера, другими словами, они использовали CNN для обеих границ объекта. и ответ на вопрос,

есть ли объект или нет. Эта архитектура иллюстрируется тем же изображением, которое использовали первоначальные авторы на рисунке 1.8.

1.2.4 Ты только один раз посмотри (YOLO)

В статье об архитектуре YOLO авторы представляют свою архитектуру в качестве первой системы детектора объектов с глубоким обучением, которая выполняет обнаружение объектов в режиме реального времени [22] со скоростью 45 кадров в секунду. Высокая частота кадров достигается за счет простоты архитектуры, в которой настраиваемый CNN предназначен для решения всей проблемы обнаружения объектов, просто отображая входное изображение в соответствующий тензор, описывающий объекты в изображении. В конечном тензоре есть параметры, описывающие местоположение объекта в разделе и вероятность того, что он является объектом в этом разделе, и вероятности для соответствующих классов. Архитектура сети YOLO здесь показана тем же рисунком, что и оригинальный документ на рисунке 1.9. Для случая, показанного на рисунке 1.9, конечный тензор представляет собой тензор 7x7x30, в котором 30 является результатом того, что сеть разработана для набора данных POSCAL VOC (20 различных объектов) и для каждого субрегиона (исходное изображение делится семь раз ширина и высота) изображения, которое выбирается сетью для создания одной гипотезы объекта и двух потенциальных ограничивающих рамок (представлены пятью значениями x, y, width, height и trust).

1.2.5 SSD: однократный детектор MultiBox

Автор статьи, описывающей архитектуру SSD [18], представляет работу, аналогичную работе YOLO [22], где системы похожи как по архитектуре, так и по производительности, но три основных различия заключаются в том, что архитектура SSD ищет объекты в нескольких масштабах карты объектов, извлечённые CNN, и этот SSD использует якорные блоки для обработки ограничительных рамок [18].

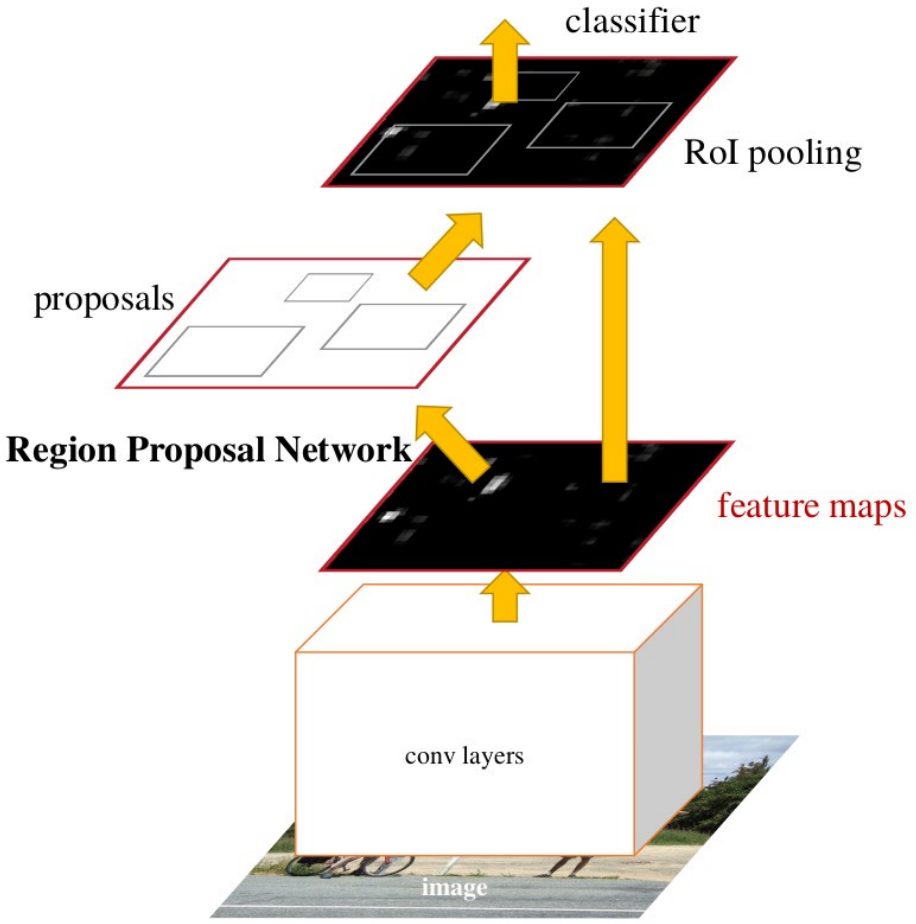

Рисунок 1.8: Иллюстрация архитектуры Faster R-CNN. Взято из [23]

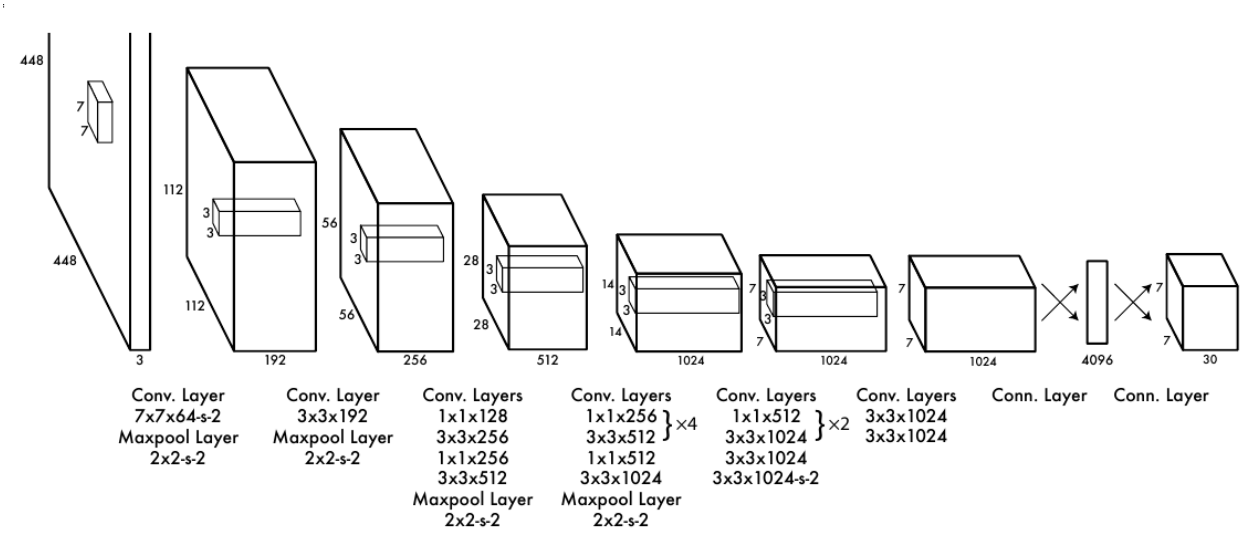

Рисунок 1.9: Иллюстрация архитектуры YOLO. Взято из [22]

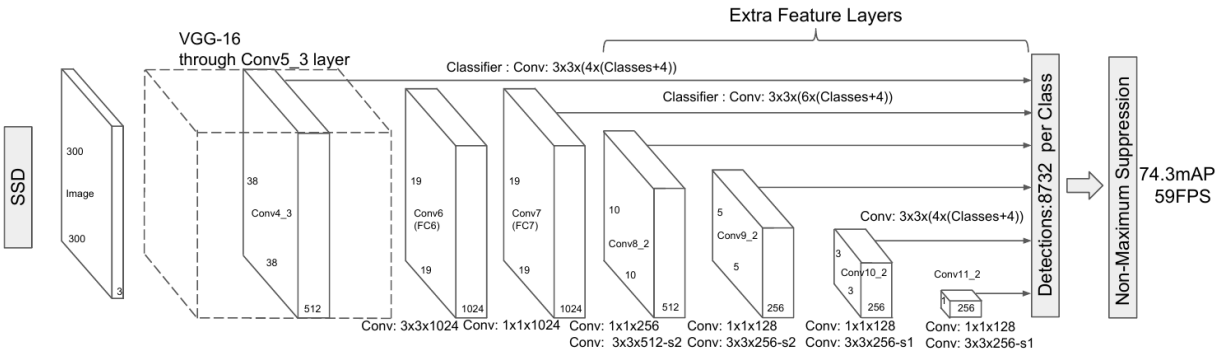

Рисунок 1.10: Иллюстрация архитектуры SSD. Взято из [18]

Детектор SSD также сильно зависит от увеличения данных и жёсткого негативного анализа, что означает, что во время обучения коллекция негативных патчей фильтруется так, чтобы соотношение между негативными и позитивными патчами составляло не более 3: 1. Шаги для увеличения данных включают случайное искажение цвета, случайное расширение и случайное кадрирование и шаг случайного горизонтального переключения. Архитектура проиллюстрирована на рисунке 1.10, первоначально представленном в [18].

R-FCN: обнаружение объекта через региональные полностью сверточные сети

В [2] Jefeng Deep и другие представляют архитектуру, которая похожа на Faster RCNN [23], используя CNN для извлечения карты объектов для определения местоположения и обнаружения, но с той разницей, что извлечённые карты объектов предназначены для позиционно-чувствительны. Это означает, что для нахождения одного объекта требуется несколько карт объектов. Эта архитектура показала выдающиеся результаты при 83: 6% mAP для набора данных PASCAL 2007 [2]. Архитектура R-FCN также была представлена как лучший детектор в обзорной статье [29], где было показано, что результаты PASCAL 2012 года составляют 85% mAP.

Дополнительная работа с изменениями для YOLO

Основной оригинальный автор статьи YOLO [22] Joseph Redmon представил две дополнительные работы в [20, 21]. Несколько изменений, внесённых в [20], заключались в реализации пакетной нормализации, использовании якорных ящиков и дополнительном добавлении сквозного уровня, позволяющего сети использовать более детализированные функции, подобные архитектуре SSD.

Пакетная нормализация - это метод нормализации выходных данных с одного уровня на другой, с интуитивным пониманием того, что интерес представляет именно распределение с предыдущего уровня. Пакетная нормализация применяется для ускорения конвергенции во время обучения [11], в случае YOLOv2, пакетная нормализация сделала ненужным использование отсева. В последней статье [21] авторы представляют обновлённую архитектуру CNN с 53 сверхточными слоями вместо предыдущих 19 в [20], и теперь YOLOv3 предсказывает коробки в трех разных масштабах. Это изменение увеличило значение COCO mAP с 21: 6% для YOLOv2 до 33: 0% для YOLOv3 в наборе данных COCO.

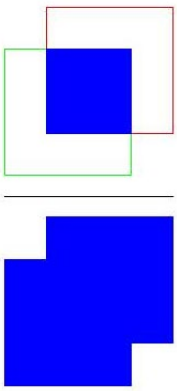

Рис. 1.11. Визуализация пересечения над объединением, поле истинности земли представлено зелёным прямоугольником, а обнаруженный объект - красным прямоугольником.

В рисунке 1.11 показано уравнение Intersection over Union

«Изучив это уравнение, мы видим, что пересечение над объединением - это просто отношение.

В числителе мы вычисляем площадь перекрытия между предсказанной ограничительной рамкой и ограничивающей рамкой.

Знаменатель - это область объединения, или, проще говоря, область, охватываемая как предсказанной ограничительной рамкой, так и ограничивающей рамкой наземной истины.

Разделив область перекрытия на область объединения, мы получаем наш окончательный результат - пересечение над Союзом.»

Также показано, что архитектура YOLOv3 выполняет обнаружение объектов со скоростью 78 кадров в секунду в [21] по сравнению с частотой кадров 45 для YOLOv1 [22] для того же оборудования.

1.2.6 Метрики для эффективности обнаружения объектов

Чтобы иметь возможность оценить производительность детектора, используются различные метрики оценки. Существует несколько различных конкурсов по обнаружению объектов, два из самых популярных это PASCAL VOC [5] и COCO [17]. Из-за различий в наборе данных и метрической системе для двух проблем, результаты между двумя наборы данных не сопоставимы. Чтобы определить, какой детектор дал лучшие результаты, тесты PASCAL и COCO определили свою собственную систему показателей или оценки. Обе метрики построены на двух метриках, называемых точностью и возвратом [8].

$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN} \quad (1.1)$$

Определение точности и напоминание о том, что TP, FP и FN - это число истинно положительных, ложноположительных и ложноотрицательных соответственно. Более конкретной проблемой обнаружения объекта является измерение пересечения по объединению (IoU), которое используется для

определения связи между предсказанными блоками обнаружения и наземными блоками истинности. Пересечение объединения проиллюстрировано на рисунке 1.11, что также может быть выражено с помощью обозначения для наборов как:

$$\mathrm{IoU} = \frac{B_{gt} \cap B_p}{B_{gt} \cup B_p} \quad (1.2)$$

где *Bgt ; Bp* относится к окну истинности и предсказанию, соответственно.

PASCAL VOC метрика

Задача PASCAL VOC использует показатель, который называется Средняя точность. Определение средней точности (AP), также называемой средней точностью (mAP), если усреднено по всем классам, имеет вид:

$$AP = \frac{1}{11} \sum_{r \in [0, 0.1, \ldots, 1]} P_{\text{interp}}(r) \quad (1.3)$$

where

$$P_{\text{interp}}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \quad (1.4)$$

и $p(\tilde{r})$ точность при вызове $\tilde{r}$.

Эти уравнения изначально найдены в статье, описывающая вызов POSCAL VOC [5]. Интуиция, лежащая в основе этого измерения, состоит в том, чтобы создать среднее значение на кривой точного возврата, см. Рисунок 1.12, где точность находится на вертикальной оси, а возврат - на горизонтальной оси. На практике увеличение числа обращений означает включение все большего числа обнаружений, предоставляемых проверенным детектором. В этом случае подразумевается, что проверенный детектор также обеспечивал достоверность наряду с обнаружением, тем самым предоставляя возможность добавлять обнаружения по одному в порядке убывания достоверности. AP, определенный

в уравнении 1.11, поэтому соответствует области под кривой Pinterp на рисунке 1.12. Для того чтобы обнаружение было классифицировано как истинный положительный результат, его IoU должно быть выше 0: 5, и с одним блоком Bp может быть связан только один Bgt box [5]. В 2010 году вызов PASCAL VOC сделал обновление для расчёта AP. Вместо усреднения по 11 одинаково разнесённым разным точкам мера изменилась на усреднённую по всем обнаружениям. Это новое определение описывается уравнением 1.5. Также необходимо указать, что это обновление не отражено в документе для PASCAL [5], это новое определение, например, можно найти в исходном коде для оценки 1.

$$AP = \sum_{r=0}^{1} (r_{n+1} - r_n) P_{\text{interp}}(r) \tag{1.5}$$

Чтобы дать пояснение за описанные метрики, приведён пример. Рисунок 1.13 содержит иллюстрацию того, как могут выглядеть обнаружения от детектора, эти обнаружения помечены в соответствии с вероятностью того, что они являются объектом, наиболее вероятно, помечены как [47]. На рисунке 1.13 поля истинности земли показаны с использованием зелёных прямоугольников, в то время как красные прямоугольники представляют обнаружения. Для того чтобы обнаружение считалось истинно положительным, его значение IoU должно составлять 0,5 или выше. Обнаружения, показанные на рисунке 1.13, дают график точного отзыва на рисунке 3.7, из которого можно рассчитать значение AP с помощью любого определения.

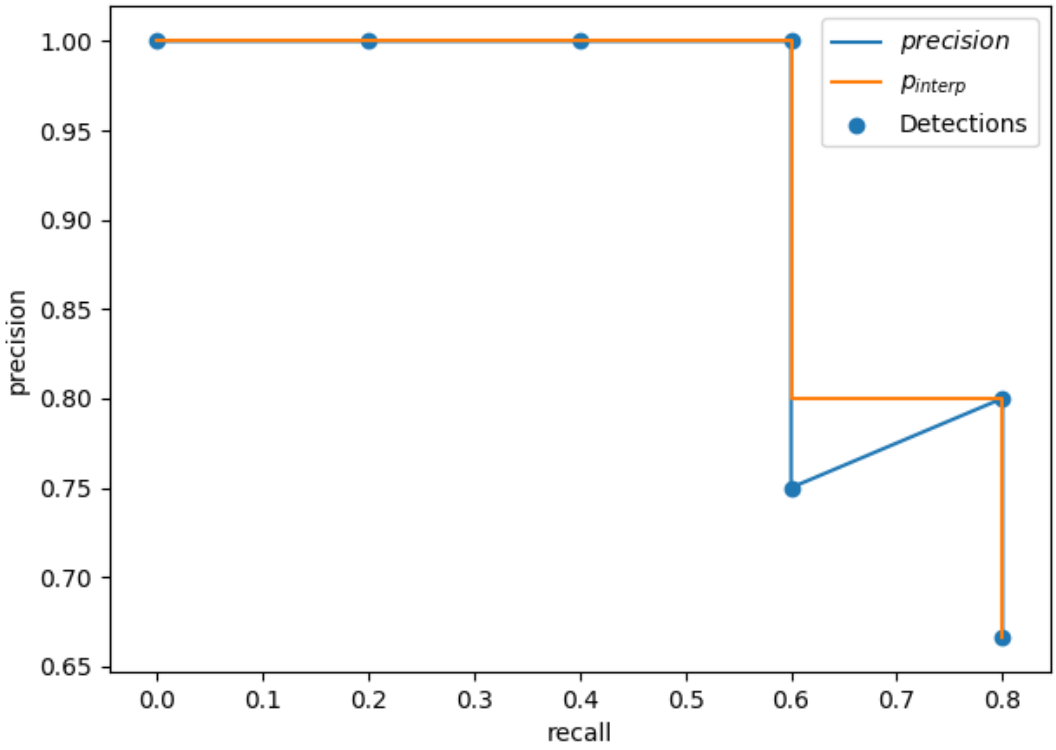

Рисунок 1.12: Визуализация точного отзыва с соответствующей кривой Pinterp. Этот рисунок был сгенерирован с помощью обнаружений и значений, представленных на рисунке 1.13.

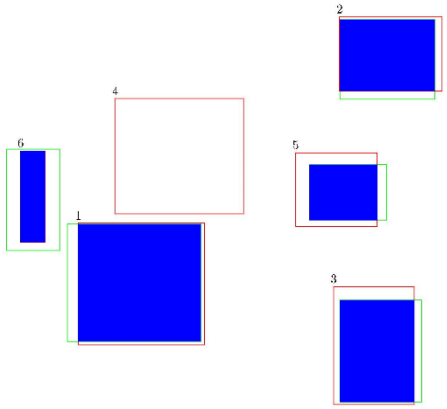

Рисунок 1.13: Иллюстрация обнаружений и соответствующих значений Precision и Recall.

Метрика для COCO

Последние исследовательские работы, как правило, дают результаты только для набора данных COCO. В COCO mAP для расчёта используется 101-точечное интерполированное определение AP. Для COCO AP - это среднее значение для нескольких IoU (минимальное IoU для рассмотрения положительного совпадения). AP @ [. 5: .95] соответствует среднему AP для IoU от 0,5 до 0,95 с размером шага 0,05. Для соревнования COCO AP - это среднее значение по 10 уровням IoU по 80 категориям (AP @ [. 50: .05: .95]: от 0,5 до 0,95 с размером шага 0,05).

```
Average Precision (AP):
   AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small            % AP for small objects: area < 32^2
   AP^medium           % AP for medium objects: 32^2 < area < 96^2
   AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
   AR^max=1            % AR given 1 detection per image
   AR^max=10           % AR given 10 detections per image
   AR^max=100          % AR given 100 detections per image
AR Across Scales:
   AR^small            % AR for small objects: area < 32^2
   AR^medium           % AR for medium objects: 32^2 < area < 96^2
   AR^large            % AR for large objects: area > 96^2
```

Рисунок 1.14: Приведены некоторые другие метрики, собранные для набора данных COCO.

Результат AP для детектора YOLOv3.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

COCO for YOLOv3

Рисунок 1.15: AP @ .75 означает AP с IoU = 0,75.

http://cocodataset.org/#detection-eval

mAP (средняя средней точности) - это среднее значение AP. В некотором контексте мы вычисляем AP для каждого класса и усредняем их. Но в каком-то контексте они означают одно и то же. Например, в контексте COCO нет разницы между AP и mAP. Вот прямая цитата из COCO:

«AP усредняется по всем категориям. Традиционно это называется «средняя точность» (mAP). Мы не делаем различий между AP и mAP и предполагаем, что разница ясна из контекста.»

В ImageNet используется метод AUC. Таким образом, даже все они следуют одному и тому же принципу в измерении AP, точный расчёт может варьироваться в зависимости от наборов данных.

Выводы: таким образом, на основании проведённого анализа существующих решений была выбрана в качестве основной нейронная сеть Yolo V3 максимально подходящего для нашего проекта.

2 Методология распознавания объектов методами машинного обучения

2.1 Нейронный сети и глубокое обучение

Визуальная идентификация объектов - чрезвычайно легкая задача для людей, но не для компьютера.

Распознавание изображения относится к процессу, с помощью которого компьютер обрабатывает изображение и идентифицирует объекты, которые существуют в этом изображении. Как правило, распознавание изображений представляет собой двухэтапный процесс.

Первым шагом является сегментация изображения [31,32,33], где изображение делится на несколько значимых областей. На втором этапе извлекаются признаки каждой области, и классификатор определяет, к какому типу объекта принадлежит эта область изображения.

Исследования по распознаванию изображений начались с распознавания текста [34, 35] в 1950-х годах. Объектами для идентификации были буквы, цифры и символы. Популярным приложением для распознавания текста является распознавание автомобильных номеров [36, 37]. Эта технология сейчас очень зрелая. Обработка цифровых изображений [38] началась в 1960-х годах. Цифровые изображения имеют большие преимущества, такие как простота хранения, удобная и сжимаемая передача, и их нелёгкое искажение. С тех пор распознавание изображений стало очень активной областью исследований. Многие различные методы были изобретены для различных целей. Но функции, которые извлекаются из изображений, как правило, разработаны человеком, что весьма специфично для приложения.

С развитием глубоких нейронных сетей [39, 40] исследования по распознаванию изображений продвигались очень быстро. Для набора данных POSCAL VOC [41] точность распознавания увеличилась с первоначальных 30% до текущих 90%.

В конкурсе 2012 ImageNet Large Scale Recognition Challenge (ILSVRC) [43] модель AlexNet достигла точности 85%, что на 10% больше по сравнению с

лучшей традиционной моделью. Преимущество использования нейронных сетей состоит в том, что сеть автоматически изучает соответствующие представления характеристик из обучающих изображений, что устраняет необходимость в разработке функций.

Это поворотный момент в истории распознавания изображений. Эти достижения смещают акцент с традиционных методов распознавания изображений на методы с использованием глубоких нейронных сетей.

В конкурсе ILSVRC 2013 все участники используют решения и алгоритмы, основанные на методах глубокого обучения. В конкурсе ILSVRC 2015 года на основе сверточной нейронной сети (CNN).

Алгоритмы достигли точности распознавания, превышающей 95 %, что выше, чем у людей.

В период с 2017 по 2018 год 29 из 38 участников конкурса предоставили решения, точность распознавания которых превышала 95 %, а самая высокая - 97,3 %. Это иллюстрирует большой потенциал глубокого обучения распознаванию изображений.

Хотя глубокое обучение до сих пор достигло больших успехов в области распознавания изображений, нам все еще предстоит решить множество проблем.

Первая проблема заключается в том, как улучшить возможности обобщения модели; как добиться того, чтобы существующая модель достигла хорошей производительности в тех типах изображений, на которых сеть не была обучена. Неподготовленные сцены могут создать много проблем для сети. Обычно простым решением этой проблемы является увеличение набора обучающих данных. Однако не всегда возможно иметь все сцены в наборе обучающих данных.

Вторая проблема заключается в том, как лучше использовать небольшие наборы данных. Идентификация объектов, которые были замечены только один раз, относительно легко для людей. Однако это очень сложно для компьютера. С существующей технологией, если используется большой набор данных,

точность распознавания может быть легко увеличена. Но с небольшими наборами данных результаты не так хороши, как хотелось бы.

Являясь важной отраслью компьютерного зрения, распознавание объектов имеет широкий спектр применений во многих областях повседневной жизни. С развитием интеллектуальных аппаратных устройств растёт количество изображений и видеоинформации, которые объединяют все более важную роль, которую играют технологии компьютерного зрения в человеческой жизни. Обнаружение и идентификация объектов также стали активными направлениями исследований. Технологии обнаружения и идентификации имеют следующие приложения в реальной жизни, такие как отслеживание объектов, видеонаблюдение, защита информации, автопилот, поиск изображений, анализ медицинских изображений, анализ сетевых данных, навигация дронов, анализ изображений с помощью дистанционного зондирования, системы защиты и т. д.

Обнаружение и распознавание объекта относится к нахождению объекта со сцены (изображения), включая два процесса обнаружения (где) и идентификации (что). Сложность задачи заключается в извлечении и распознавании обнаруживаемой области. Следовательно основная структура задачи состоит в том, чтобы сначала установить модель для выделения областей-кандидатов со сцены, а затем определить модель классификации области-кандидата. Наконец, выполняется точная настройка параметров модели классификации и местоположения эффективного кадра-кандидата.

Методы обнаружения и идентификации объектов в целом можно разделить на две категории:

1. Метод обнаружения и распознавания объектов на основе традиционной обработки изображений и алгоритмов машинного обучения.

2. Метод обнаружения и распознавания объектов на основе глубокого обучения.

В моей работе я буду опираться на 2 метод.

2.1.1 Нейронные сети (NN) и Сверточные нейронные сети (CNNs)

Целью нейронной сети, используемой в контролируемом обучении, является изучение отображения между входом x и выходом y . Другими словами, нейронная сеть может рассматриваться как аппроксиматор функции, y = f (x). Нейронные сети обычно реализуются с использованием нейронных слоёв. Нейронная сеть может быть реализована путём объединения нескольких функций вместе.

$$y = f^3(f^2(f^1(x))). \quad (2.1)$$

Простой пример нейронной сети с возможностью реализации функции XOR(исключающего ИЛИ) может быть реализована с использованием одного скрытого слоя с двумя скрытыми единицами; эта функция также может быть описана как:

$$y = f^2(f^1(x)) \quad (2.2)$$

Чтобы сеть реализовала функцию XOR, необходимо использовать хотя бы одну нелинейную функцию; это может быть достигнуто с помощью точного преобразования, за которым следует нелинейная функция активации. Поэтому функция f определяется как:

$$f^1(x) = g(W^T x + b^1) \quad (2.3)$$

где g представляет нелинейную функцию активации, а W, b представляют изучаемые веса и смещения, соответственно. Для функции f 2 достаточно использовать точное преобразование, приводящее к окончательному определению:

$$f^2(f^1(x)) = w^T g(W^T x + b^1) + b^2 \quad (2.4)$$

Когда все нейроны в слое связаны с каждым нейроном в следующем слое, сеть называется Полностью Сверточная Нейронная Сеть (FCNN). Это очень мощная архитектура, способная, как указано в теореме универсального

приближения [10], аппроксимировать любую измеримую по Борелю функцию, по крайней мере, с одним скрытым слоем, использующим функцию сжатия как функцию, отображающую все действительные числа в открытый интервал от нуля до единицы и достаточное количество скрытых блоков.

Сверточные нейронные сети (CNN) представляют собой особый тип нейронных сетей или, в данном случае, сетей прямой связи. CNN обычно используются, когда входом в сеть являются изображения. Более подробная информация о шаге свёртки этих сетей представлена в следующем разделе.

Если теперь сеть с одним скрытым слоем способна аппроксимировать какую-либо функцию, то в чем собственно проблема? Способность реализовывать любую функцию - это не то же самое, что способность изучать любую функцию. А в некоторых случаях количество необходимых скрытых единиц может оказаться невозможным в вычислительном отношении. По этим причинам проводится огромное количество исследований, чтобы найти различные функции активации (g в уравнении 2.3) и описываются различные архитектуры и структуры сети. Эти вопросы о том, как устроена сеть и какую функцию активации следует использовать для того, чтобы сеть выучилась наилучшим образом.

Кстати, до сих пор это большое поле для активных исследований. В следующих разделах представлены различные архитектуры CNN. Выход из этих слоёв, описанных в следующие разделы как и в любой сети, проходящей через различные типы функций активации.

2.1.2 Свёртка

Математически свёртка определяется как:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{2.5}$$

Эта операция описывает отображение того, как сигнал x (a) взвешивается с сигналом w (t). Обычно функцию w (t) можно рассматривать как фильтр, применяемый к сигналу x(t).

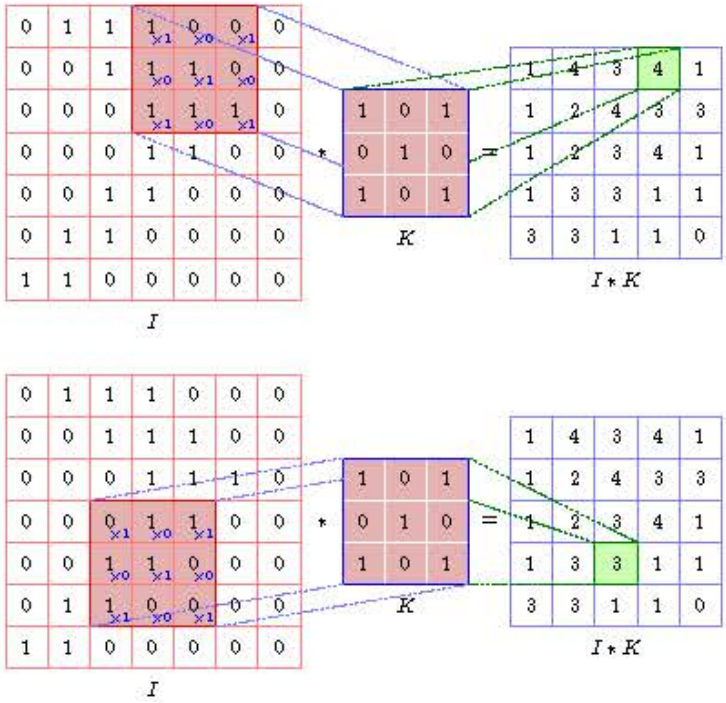

Рисунок 2.1: Простая иллюстрация того, как ядро K скользит по I, чтобы генерировать вывод.

В случае изображений входной сигнал является двухмерным и дискретизированным. Если I обозначает входное изображение и K представляет фильтр, свёртка затем определяется по уравнению 2.6.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.6}$$

Определения, данные в уравнениях 2.6, 2.7, являются обеими реализациями того, что математически называется свёрткой, но обычно, когда свёртка описывается в условиях глубокого обучения, функция, фактически обозначаемая свёрткой, является функцией взаимной корреляции. Кросс-

корреляция обладает свойствами, аналогичными свёртке, но одно отличие состоит в том, что функция кросс-корреляции не является коммутативной.

Функция взаимной корреляции определяется как:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.7}$$

функция, почти такая же, как функция, описанная в уравнении, но без зеркального отображения ядра. Подобно одномерному случаю, функция, определенная в уравнении 2.7, может быть описана как отображение из входного изображения в выходное изображение, где ядро было применено в нескольких местах изображения. Простым примером отображения, проводимого функцией взаимной корреляции, является приведены на рисунке 2.1. Размер получающегося изображения зависит не только от ширины и высоты ядра, но также от двух параметров, называемых шагом и отступом. Stride определяется как количество пикселей, которое ядро перемещает между производящими каждый элемент в выводе. Функция, включающая параметры шага, определяется как:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i \cdot s + m, j \cdot s + n) K(m,n) \tag{2.8}$$

Чтобы сохранить размер изображения, отступы используются для обработки случаев, когда ядро находится близко к границам I.

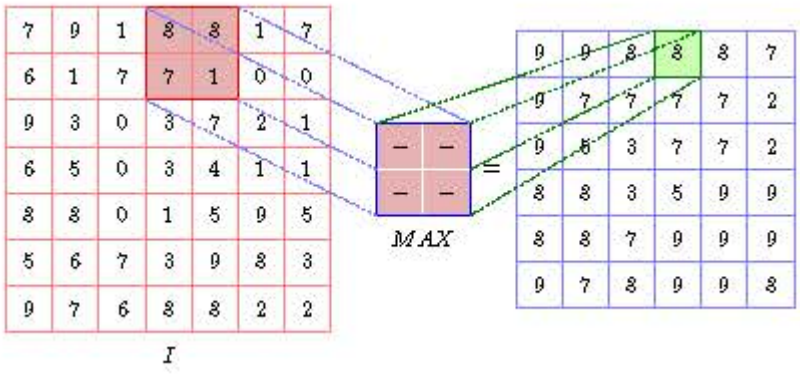

Рисунок 2.2: Визуализация того, как максимальный пул 2x2 отображает входное изображение на выходное изображение.

Есть несколько различных вариантов того, как сделать заполнение изображения, один из них - добавить нули на границе I, чтобы предотвратить уменьшение выходных значений по ширине и высоте. Для случая, когда использовались как отступ, так и шаг, не равный единице, ширина или высота на выходе рассчитывается как представление размера вывода, размер ввода и размер ядра:

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1 \quad (2.9)$$

o, i, k представляют размер вывода, размер ввода и размер ядра, p, s представляет количество нулей, добавленных к входу, и размер шагов, предпринятых ядром (то есть шаг). Более подробное описание того, как рассчитать выход с помощью сверточной операции, представлено в [4, 8].

2.1.3 Pooling (Объединение)

Обычно свёртка - не единственная операция, выполняемая в сверточной нейронной сети. Другая популярная операция называется пулом, которая предназначена для уменьшения размеров входных данных. Это может быть сделано с использованием максимального пула или среднего пула или некоторой другой математической функции для объединения значений. На рисунке 2.2 показано, как изображение объединяется с использованием фильтра пула 2x2 max. Основной мотивацией использования пула, кроме уменьшения размеров карты объектов, является уменьшение пространственной дисперсии в объектах. Этот факт интуитивно понятен, когда понимаешь, что подано несколько карт объектов в пул слой, который приведёт к тому же выходу. По аналогии со сверточной операцией выходной размер операции объединения можно рассчитать как:

$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1 \quad (2.10)$$

где еще раз i, k, s, представляет размер изображения, размер окна объединения и количество шагов в операции объединения.

2.1.4 Свёртка, используемая на практике

Одной из причин использования свёртки в CNN является уменьшение размерности входного изображения, что приводит к меньшим картам объектов, размер которых зависит от проблемы, решаемой CNN. Чтобы понять, как это достигается, важно понять, как сверточная операция отображает многомерное изображение, где третье измерение обычно является спектральным

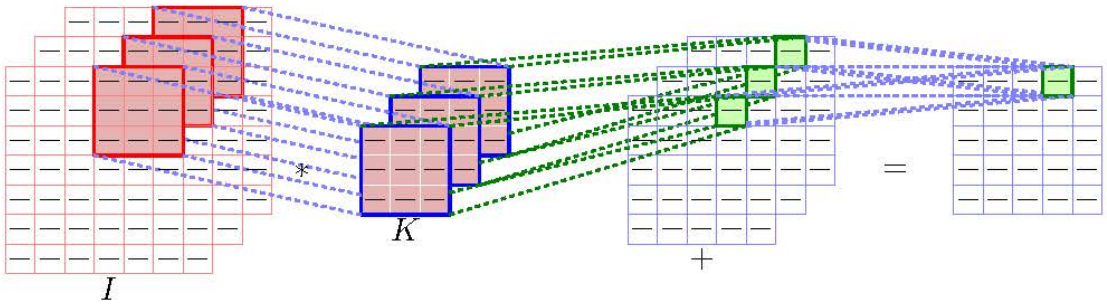

Рисунок 2.3: Визуализация того, как свёртка проводится в нескольких измерениях.

измерением, часто представляемое тремя цветами (RGB). Операция многомерной свёртки выполняется простым расширением суммирования по третьему измерению, которое, как мы предполагаем, имеет значения с для каждого пикселя. В качестве альтернативы, эта операция может быть описана как генерация числа карт нормальных объектов и объединение этих карт путём добавления каждого пикселя с соответствующими значениями пикселей. Следовательно, это снова приводит к двумерной карте объектов. Иллюстрация свёртки в нескольких измерениях представлена на рисунке 2.3. [8]

2.1.5 1 x 1 Свёртка

Частным случаем свёртки, который оказал большое влияние на область проектирования архитектуры нейронной сети, является свёртка один за другим, первоначально описанная и исследованная в [16]. Сначала может показаться странным выполнение свёртки с ядром 1 x 1, но на самом деле это очень мощная операция, которая позволяет увеличить количество слоёв и весов без уменьшения размеров элементов; это приводит к более мощной архитектуре. Одна архитектура, которая использует этот тип уровня, описана в разделе 1.3.1.

2.2 Особенности метода этого метода Thresholding with mask (порог)

Thresholding является одним из наиболее распространённых (и основных) методов сегментации в компьютерном зрении, и он позволяет нам отделить передний план (то есть объекты, которые нас интересуют) от фона изображения.

Thresholding приходит во многих формах. У нас есть простое пороговое значение, при котором мы вручную задаём параметры для сегментирования изображения - это очень хорошо работает в условиях контролируемого освещения, где мы можем обеспечить высокую контрастность между передним планом и фоном изображения.

Thresholding - это бинаризация изображения. В общем, мы стремимся преобразовать изображение в градациях серого в двоичное изображение, где пиксели равны 0 или 255.

Простым примером порогового значения будет выбор порогового значения T , а затем установка всех значений интенсивности пикселей меньше T на ноль, а всех значений пикселей больше T на 255. Таким образом, мы можем создать двоичное представление изображения.

Мы обнаружим объект на изображении с помощью цветового порога. В качестве входа у нас есть изображение и цвет Ларги.

Мы найдём необходимую маску для этого объекта удобным способом по трек-барам в OpenCV. Затем мы применим эту маску, чтобы найти координаты нужного объекта в режиме реального времени с помощью камеры.

Наконец, мы нарисуем ограничивающий прямоугольник с меткой вокруг обнаруженного объекта.

Алгоритм заключается в следующем:

Во-первых, мы читаем входное изображение в цветовом пространстве RGB.

Затем мы конвертируем его в изображение с цветовым пространством HSV.

Наконец, мы выбираем правильную маску для обнаружения цвета Ларги. В следствии, у нас будут номера для выбранной маски.

Импортируем библиотеку OpenCV и начинаем готовить трек бары это поможет нам выбрать номера для цветовой маски. Для этого мы используем функцию OpenCV createTrackbar (). В первом аргументе мы указываем имя самой трекбара.

Во втором аргументе мы указываем имя окна, в котором оно будет отображаться. Само окно мы определяем выше. Затем мы устанавливаем диапазон чисел от нуля до максимума 255.

Также мы должны передать в качестве аргумента функцию, которая будет делать что-то с выбранным числом. Вот почему мы определяем здесь пустую функцию, потому что нас интересуют только числа без какой-либо обработки.

Нам нужны шесть чисел для минимального и максимального диапазона цветов RGB: красный, зелёный и синий.

Также нам нужен цикл while, чтобы быть видеть изображение.

Теперь мы читаем входное изображение с помощью функции OpenCV, которая выдаёт нам изображение BGR в виде массива. Мы уменьшаем его с помощью функции resize (), чтобы использовать более удобное маленькое окно, и мы покажем оригинал изображение функцией imshow (), в которой мы передаём название окна и само изображение.

Когда мы создаём окно OpenCV, мы можем передать аргумент WINDOW NORMAL, который даст нам возможность изменить размер окна при его открытии.

Затем мы конвертируем входное изображение из BGR в цветовое пространство HSV. HSV - обозначает оттенок, насыщенность и значение. С этим цветовым пространством очень легко обнаружить объект по цвету. Изменяя значение свойства, мы сохраняем нужные цвета и пропускаем ненужные.

Теперь мы запускаем цикл while, где определяем переменные, чтобы получить шесть чисел для цветовой маски в соответствии с текущим положение

соответствующей трек-бара. Например, минимум синего цвета мы получаем из трек-бара с названием min blue на . Когда у нас есть все шесть чисел, мы реализуем операцию порогового значения с помощью функции OpenCV inRange (), где мы передаём наше изображение HSV и диапазон пикселей значения от минимального до максимального. Затем мы показываем приведённое двоичное изображение в окне и повторяем эту операцию в цикле, пока мы находим правильные шесть чисел для цветовой маски.

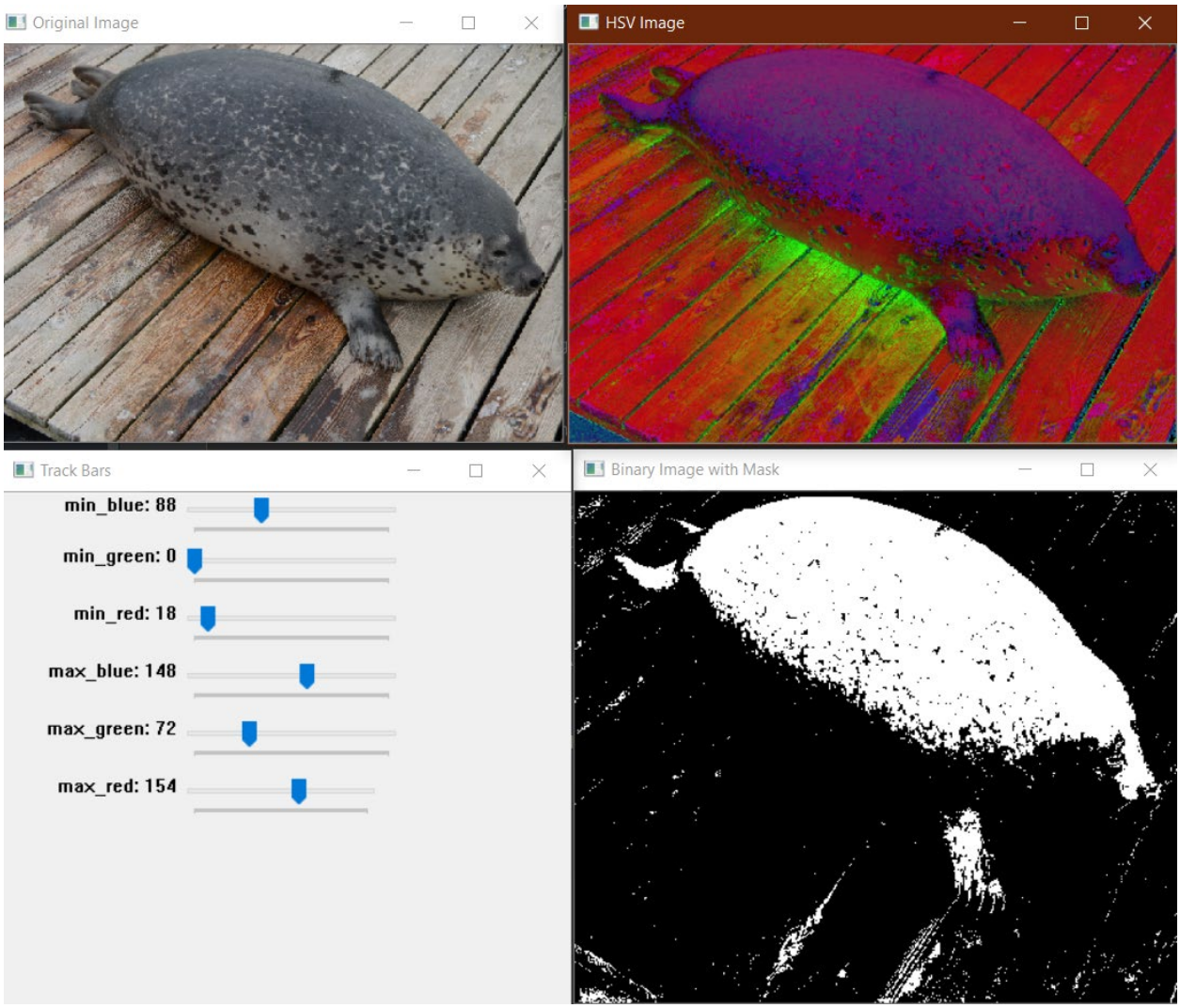

Рисунок 2.4: Операция в цикле, пока мы находим правильные шесть чисел для цветовой маски

Теперь мы остановим цикл и в терминале можно увидеть шесть печатных номеров.

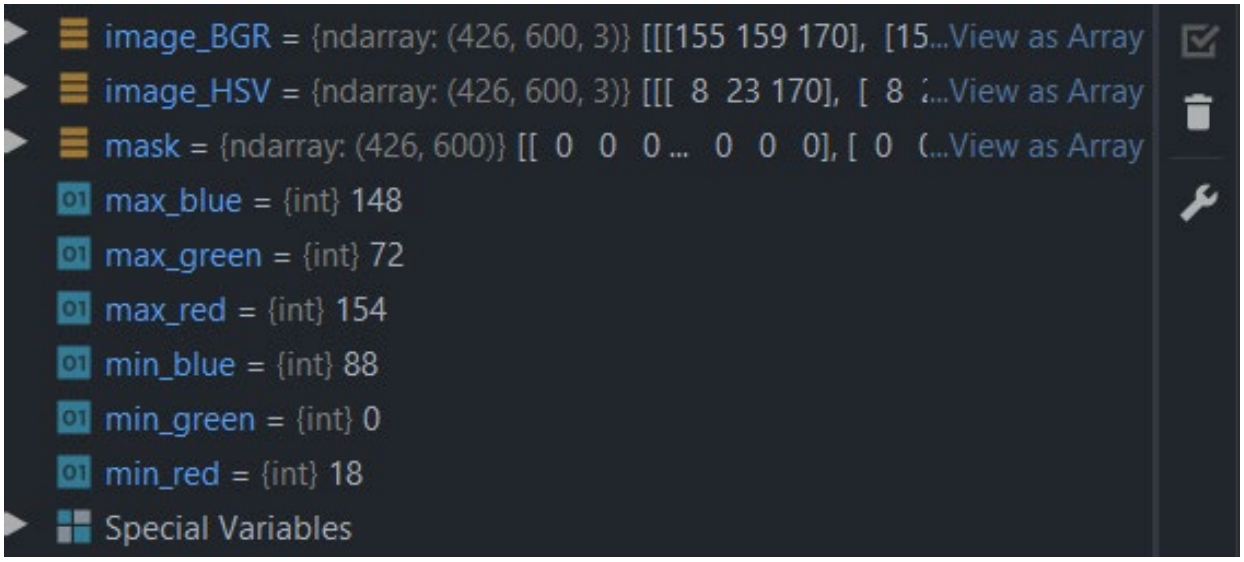

Рисунок 2.5: Нашли необходимую маску для детекции Ларги Object в режиме реального времени на веб камере.

Мы нашли необходимую маску для Ларги Object.

Обнаружение объекта в режиме реального времени и рисование границы коробки и этикетки стандартными функциями OpenCV мы сделаем в главе 3.

2.3 Создание набора данных

Как описано в главе 1, основной целью данной дипломной работы является дать нашим учёным работать с самыми передовыми системы распознавания объектов. А также разработать для них инструментарий чтобы повысить точность распознавания морских животных. Для реализации этой проблемы, необходимы изображения соответствующих характеристик. В следующих разделах представлены мотивы выбора изображений и описание того, как эти изображения были созданы.

Чтобы создать набор данных для глубокого изучения изображений, мы собираемся использовать API поиска **Bing Image от Microsoft**, который

является частью **Cognitive Services от Microsoft**, используемых для предоставления искусственного интеллекта зрению, речи, тексту и многим другим приложениям и программному обеспечению. Аналог поиска изображений **Google Images** для создания своего собственного набора данных - это утомительный, ручной процесс. Вместо этого я искал решение, которое позволило бы мне *программно* загружать изображения с помощью запроса.

Я не хотел открывать свой браузер или использовать расширения браузера для загрузки файлов изображений из моего поиска.

Много лет назад Google устарел в своём собственном API поиска картинок.

Для решения своей задачи я решил попробовать Microsoft Bing Image Search API.

Результаты были актуальны, и API было легко использовать.

Он также включает в себя бесплатную 30-дневную пробную версию, что меня вполне устраивало.

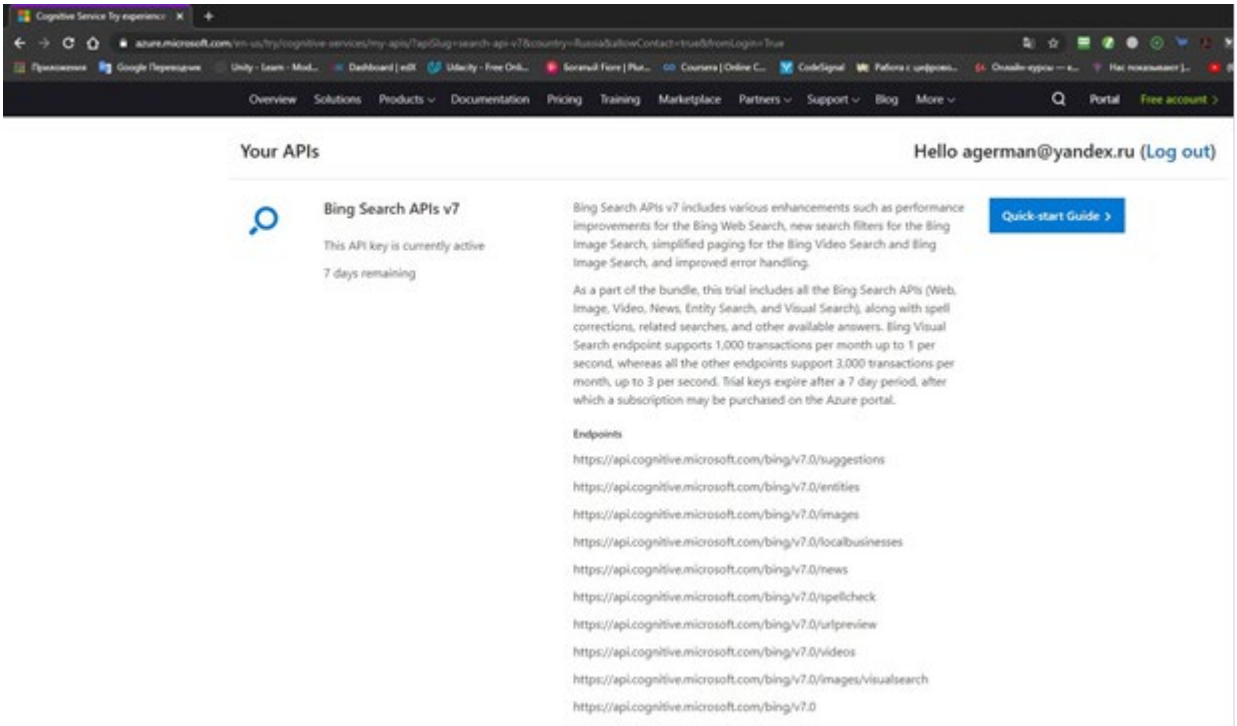

Рисунок 2.6: Получение API key для доступа к поиску изображений .

Как мы можем видеть из скриншота, пробная версия включает в себя все API-интерфейсы поиска Bing с общим числом транзакций 3000 в месяц - этого будет более чем достаточно, чтобы поиграть и создать наш первый набор данных глубокого обучения на основе изображений.

Настроим некоторые глобальные переменные:

```
построить набор изображений с глубоким обучением
# установите ключ API Microsoft Cognitive Services вместе с (1)
# максимальное количество результатов для данного поиска и (2) размер группы
# для результатов (максимум 50 на запрос)
API_KEY = "YOUR_API_KEY_GOES_HERE"
MAX_RESULTS = 250
GROUP_SIZE = 50
# установить URL API конечной точки
URL = "https://api.cognitive.microsoft.com/bing/v7.0/images/search"
```

Рисунок 2.7: Установка API key в код для работы скрипта .

Используя API, мы смогли программно загрузить изображения для обучения в глубокой нейронной сети, что стало огромным шагом вперёд по сравнению с необходимостью ручного сохранения изображений с помощью Google Images .

API поиска изображений Bing можно использовать в течение 30 дней, что идеально, если нужно быстро найти максимальное количество изображений для создания датасета.

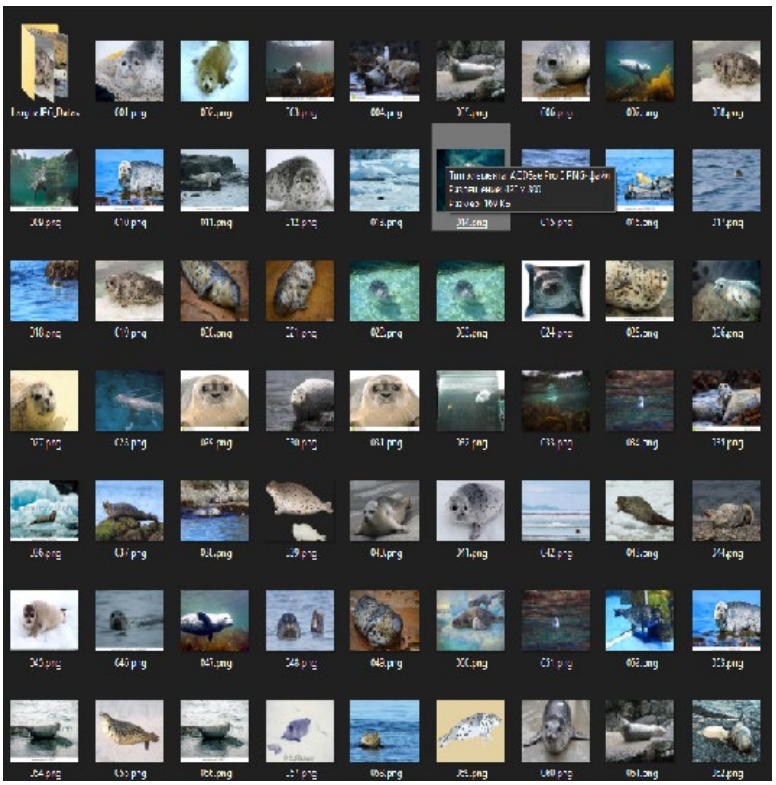

Рисунок 2.8: Получение изображений через API .

Однако как мы видим не каждое изображение, которое мы скачали, будет иметь отношение к нашему запросу – большинство будет, но не все из них.

К сожалению, нам вручную потребуется просмотреть все скаченные изображения на предмет их релевантности.

Так же получили от Национального научного центра морской биологии ДВО РАН, несколько архивных изображений Ларг за которыми они наблюдают что также дополнило мой датасет качественными изображениями.

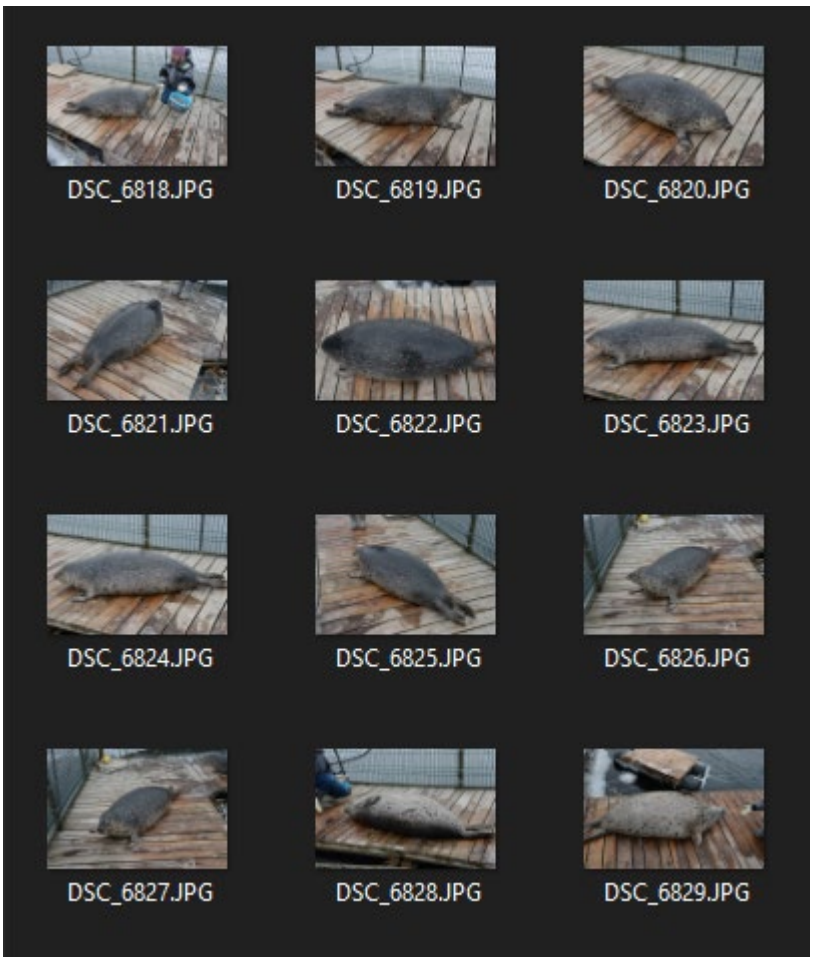

Рисунок 2.9: Получение изображений от ДВО РАН .

После сбора изображений приступили к аннатонированнию изображении для создания датасета.

2.3.1 Разметка изображений

На этом этапе нам нужно использовать метод аннотирования данных. Аннотация означает пометить данные как изображения и видео, которые у нас есть, на основе алгоритма, используемого для обнаружения.

Существуют различные типы методов аннотации: аннотация ограничительной рамки, семантическая аннотация, контурная аннотация и др. Для того, чтобы обучить версию YOLO V3, будем использовать аннотацию ограничительной рамки. Мы будем маркировать собственные файлы данных и структуры в формате YOLO. Мы помечаем изображения и видео и готовим собственный набор данных для обучения.

Существует ряд инструментов для разметки изображений, которые можно использовать как для обнаружения объектов, так и для сегментации изображений.

Photoshop and Gimp Первый набор инструментов, которые можно рассмотреть для разметки собственного набора данных изображений, является наиболее очевидным, включая полнофункциональные пакеты программного обеспечения для обработки изображений, такие как Photoshop и Gimp.

При использовании одного из этих инструментов ваш рабочий процесс будет выглядеть примерно так:

- Загрузите одно изображение из набора данных в программное обеспечение;
- используйте инструменты рисования, такие как прямоугольники, круги, эллипсы или инструмент многоугольника, чтобы нарисовать маску;
- заполните маску определенным цветом. Каждый ярлык будет иметь свой цвет;
- сохраните маску на диск в формате файла изображения без потерь (например, PNG).

Такой рабочий процесс полностью пригоден для аннотирования набора данных изображений. Проблема в том, что использование такого инструмента может стать чрезвычайно утомительным - Photoshop и Gimp не предназначены для аннотирования изображений.

Кроме того, такой инструмент составляет элемент человеческой ошибки. Слишком легко забыть, какой цвет соответствует какой маркировке класса.

Во-вторых, нет способа наложить метку класса на изображение / маску внутри программного обеспечения. Чтобы назначить метку класса, вам необходимо определить специальную структуру каталогов или поддерживать какой-либо текст поиска, файл JSON, YAML и т. д.

RectLabel Лучшим решением было бы использовать инструмент, такой как RectLabel который позволяет рисовать ограничивающие прямоугольники, многоугольники и кубические сплайны Безье.

Этот инструмент разработан с возможностью обнаружения и сегментации объектов, что позволяет вам экспортировать индексированные цветовые маски, что делает его довольно простым для интеграции в вашу функцию load_mask. RectLabel также включает в себя возможность одним нажатием кнопки наносить метки на области интереса.

Недостатком RectLabel является то, что это только macOS, поэтому, если у вас нет Mac, вам нужно попробовать другой инструмент.

Imglab Инструмент imglab является частью библиотеки dlib [44] и фактически используется в основном для обнаружения объектов. Используя imglab, вы можете нарисовать ограничивающий прямоугольник вокруг объекта, а затем предоставить метку.

Если все объекты, которые вы комментируете, имеют прямоугольную форму, то imglab - отличный вариант - вы можете аннотировать каждое отдельное изображение, и imglab создаст файл XML со всеми данными аннотаций. Используя эти данные аннотации, вы можете обновить свой подкласс Dataset для анализа XML-файла и использовать функцию OpenCV cv2.rectangle для рисования фактической маски.

Если ваши маски *не* являются строго прямоугольными, вам нужно использовать другой инструмент.

LabelMe LabelMe , стал своего рода стандартом для аннотаций изображений, включая обнаружение объектов и сегментацию экземпляров.

Инструмент полностью с открытым исходным кодом, написан на Python и предназначен для использования с графическим интерфейсом Qt.

Хотя инструмент полностью работоспособен и невероятно полезен, лично я обнаружил, что интерфейс немного медленный, «неуклюжий» и в целом менее интуитивно понятный, чем некоторые другие инструменты в этом

разделе. Конечно, попробуйте, но я думаю, что вы в конечном итоге выберете другой инструмент.

LabelImg

LabelImg (по сути является ответвлением от аннататора LabelMe.

LabelImg также опирается на Python и Qt, хотя пользователи Windows и Linux будут рады узнать, что двоичные файлы для приложения существуют - вам не нужно явно создавать их, как если бы вы использовали LabelMe. Однако читателям, использующим macOS, все равно нужно будет собрать проект из исходного кода.

Как и LabelMe, я нахожу LabelImg немного неуклюжим и сложным в использовании.

Но он оказался самым подходящим для проекта. Так как, сразу сохраняет аннотации в формате YOLO.

LabelBox В отличие от других проектов с открытым исходным кодом, Labelbox представляет собой приложение SaaS (программное обеспечение как услуга).

Существуют как бесплатные, платные, так и размещённые планы для продукта, но основная суть в том, что загружать и устанавливать нечего - вы просто указываете свой браузер на свои данные, либо путём загрузки на сервер LabelBox, либо запуска локально, и затем комментируйте *прямо* в своих браузерах.

Поскольку современные браузеры ведут себя скорее как «операционные системы», а не как «веб-браузеры», я вижу очень мало причин, почему мы должны ограничиваться загружаемыми частями программного обеспечения для аннотации изображений.

VGG Image Annotator (VIA) [45] от известной группы визуальной геометрии в Оксфордском университете.VIA также является проектом с открытым исходным кодом, но основное преимущество заключается в том, что (1) он работает в вашем браузере и (2) *полностью* автономен - все, что вам нужно сделать, это загрузить файл HTML и открыть его веб-браузере.

Оттуда пользовательский интерфейс поможет вам загрузить ваш набор данных изображений. Затем вы можете аннотировать ваши изображения как для обнаружения объектов, так и для сегментации экземпляров.

После того, как вы закончили комментировать свои изображения, мы можем скачать файл JSON или CSV, содержащий сегментацию. Можно и сохранить сам проект, убедившись, что если закроется браузер, мы сможем перезагрузить проект с того места, где остановились, и продолжить комментировать.

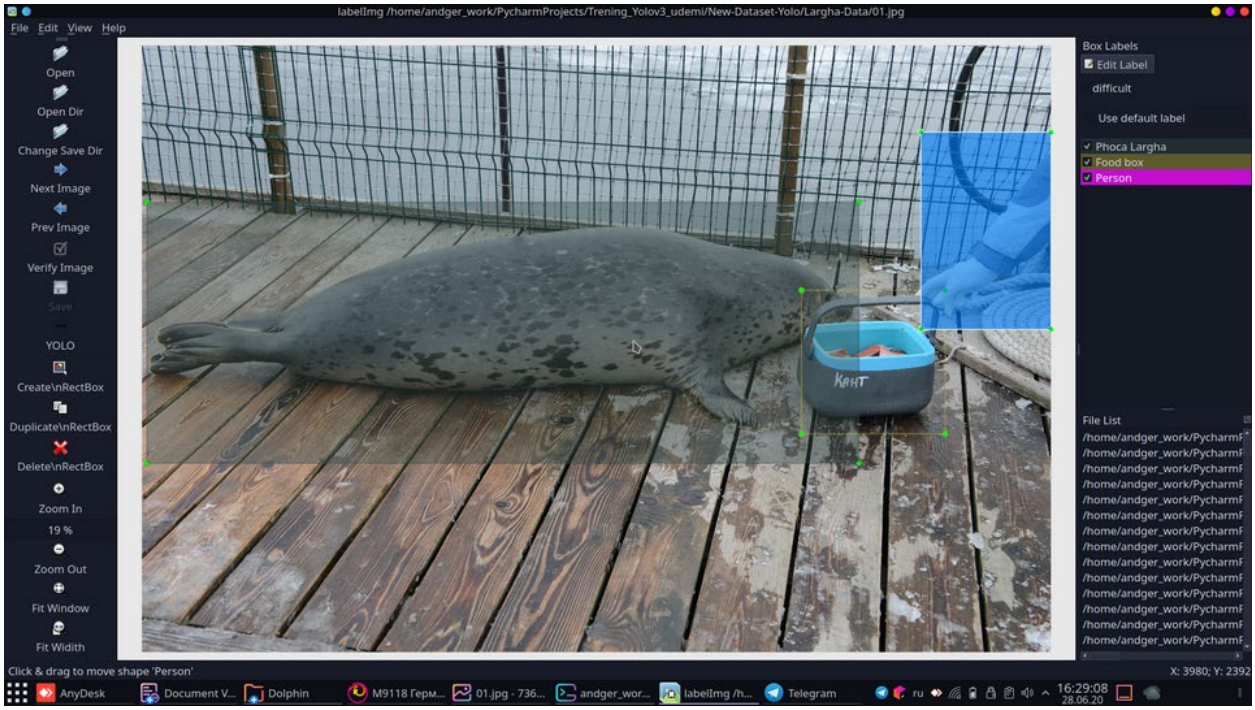

Рисунок 2.10: Разметка в формате Yolo в ***LabelImg.***

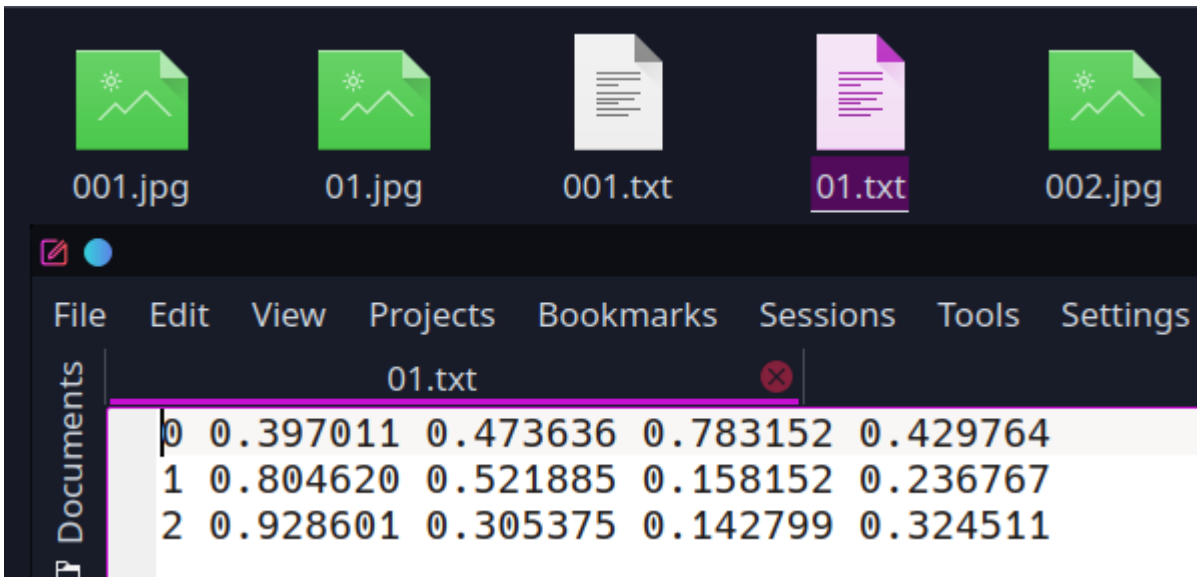

Рисунок 2.11: Разметка изображения в формате YOLO.

Рисунок 2.4 Показывает номер класса Phoca_Largha, который равен 0, потому что это первый класс в списке, затем 1 FoodBox и 2 класс Person.Затем центральная точка объекта по x и y, ширина и высота объекта. Все четыре числа нормализуются по реальной ширине изображения и реальной высоте изображения.

Маркировка видео в формате YOLO происходила схожим образом.

Рисунок 2.12: Инструмент командной строки ffmpeg.

Мы будем использовать инструмент командной строки ffmpeg. Выбираем видео файл с которого хотим получить изображения. Если мы возьмём только одно изображение в секунду, это не будет достаточно для обучения. Если мы берём 50 изображений в секунду это будет слишком много, а разница между соседними изображениями будет небольшой. Мы возьмём 4 изображения (*fps=4*). После применения команды мы получаем планируемое количество

изображения для увеличения нашего набора для датасета что значительно увеличит точность классификации объектов при детекции.

2.3.2 Подготовка файлов для обучения

Когда дело доходит до обучения YOLO версии 3 в среде Darknet, нам потребуется несколько файлов:

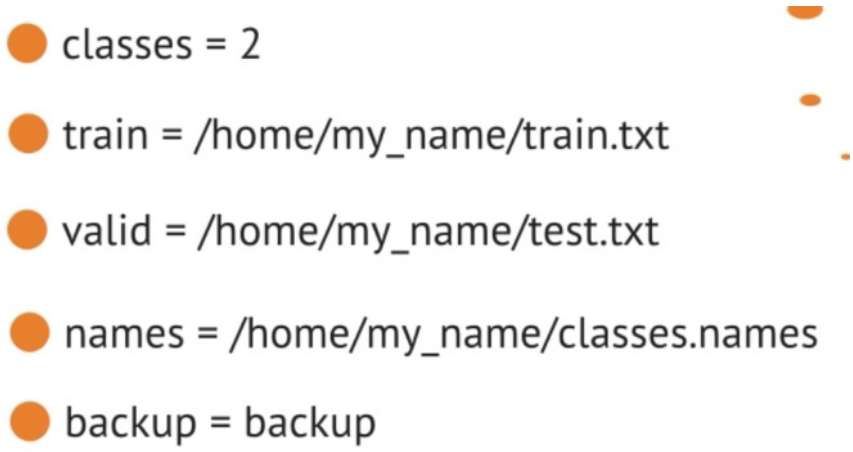

Рисунок 2.13: Файлы для обучения YOLO версии 3 в среде Darknet.

После того, как все изображения были помечены, пришло время подготовить другие файлы, необходимые для обучения в среде Darknet.

Эти файлы:

- labelled.data;
- classes.names;
- train.txt;
- test.txt.

Пять строк внутри labelled.data :

- классы = 3
- train = / home / my_name / train.txt
- valid = / home / my_name / test.txt
- имена = / дом / my_name / classes.names
- резервная копия = резервная копия

Первая строка указывает количество классов, а именно количество помеченных объектов, на которых будет обучаться YOLO v3 , и которые будут использоваться для обнаружения после обучения.

Во второй строке указывается полный путь к файлу train.txt, который, в свою очередь, содержит полные пути к изображениям для обучения. То же самое относится и к **третьей строке** с той разницей, что изображения используются для проверки во время обучения.

Четвертая строка указывает полный путь к файлу classes.names, в которой есть имена помеченных объектов.

Пятая строка указывает папку, в которой будут сохранены обученные веса.

Файлы train.txt и test.txt выглядят следующим образом (каждый путь находится в новой строке):

- /home/my_name/labelled-images/image001.jpg
- /home/my_name/labelled-images/image002.jpg
- /home/my_name/labelled-images/image003.jpg
- ...
- /home/my_name/labelled-images/image799.jpg
- /home/my_name/labelled-images/image800.jpg

Файл classes.names выглядит следующим образом (имена классов и их количество могут быть разными):

- Phoca Largha.
- FoodBox.
- Person.

Создаём папку с именем Labeled-Data, чтобы все было организовано.

- Labeled -Data /
- full-path.py
- train.py
- creating-names.py

Прежде чем создавать необходимые файлы для обучения в среде Darknet , необходимо найти абсолютные или полные пути к каталогу с аннотированными изображениями.

Когда будет найден полный путь, пора создавать файлы train.txt и test.txt.

Затем настало время для создания файлов labelled_data.data и classes.names.

Подводя итоги того, что мы сделали и чему научились.

Мы получили новые навыки. Мы можем помечать объекты на одном изображении, ограничивая рамки и сохраняя результаты в YOLO формат. И даже больше, мы можем пометить объекты на видео. Для этого мы узнали, как разбить видео на множество изображений. Мы обсуждали что такое формат YOLO и как он выглядит. Мы узнали, что для обучения нам нужно иметь в нашем набор данных TXT-файлов с аннотациями. И эти текстовые файлы должны иметь те же имена, что и изображения.

В следующем разделе мы собираемся создать собственный набор данных в формате YOLO из существующего.

2.4 Добавление необходимого класса или несколько классов для обучения из существующего общедоступного набора данных

В прошлой главе мы узнали, как пометить собственный набор данных.

Мы пометили изображения и видео ограничивающими рамками. Но что, если мы хотим добавить в обучение нашего детектора YOLOv3 другие популярные объекты? Такими популярными объектами могут быть люди, собаки, кошки и многие другие.

Можно ли извлечь только необходимые данные для обучения из уже помеченного огромного существующего набора данных?

Мы можем получить только необходимый класс или несколько классов для обучения из существующего набора данных. Для этого мы будем использовать открытый набор данных изображений, который общедоступен в Интернете.

На нем помечены изображения из 600 классов. Для обучения нашего детектора объектов YOLOv3, мы извлечём несколько классов из этого набора данных и структурируем файлы в формате YOLO. Загрузка изображений из огромных открытых изображений Dataset:

Рисунок 2.14: Open Images Dataset – открытый большой датасет.

Используя уже установленный инструментарий OIDv4, загружаем нужный класс изображений для обучения.

Для загрузки нужных классов используем аргументы:

- ***--classes «Car Bicycle_wheel Bus»***

имена классов (обратите внимание, здесь у нас есть имя класса, состоящее из двух слов, и нам нужно использовать символ нижней черты для их соединения вместо использования пробела)

- ***--type_csv train***

указание типа набора данных (поезд, проверка и тестирование; или все)

- ***-- multiclasses 1***

указав, что все классы будут загружаться вместе в одной папке

- ***--limit 800***

указание количества изображений, которые будут загружены для каждого класса.

Чтобы проверить загруженные аннотации, просто запускаем **визуализатор**, который будет отображать изображения и ограничивающие рамки, выполнив следующую команду в Терминале.

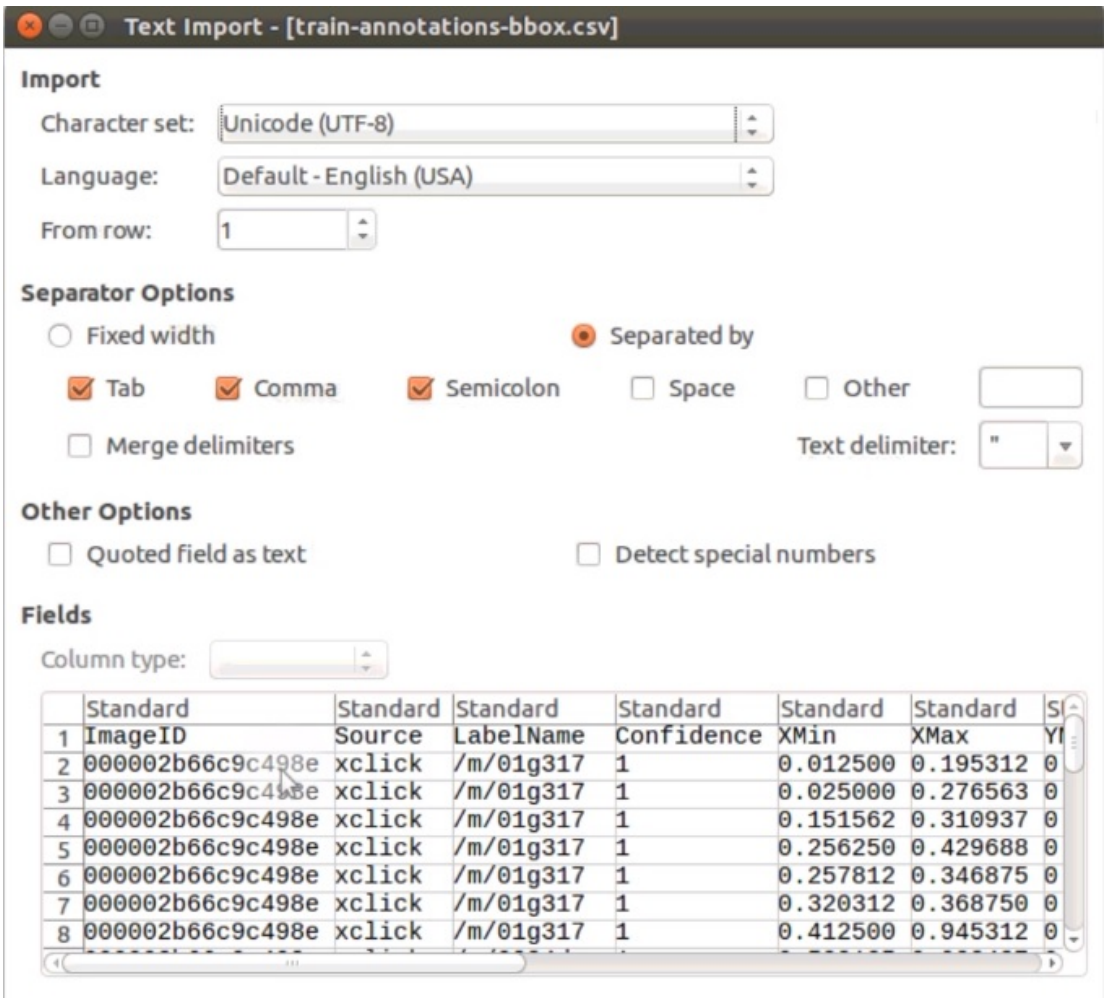

Рисунок 2.15: CSV-файл и папка с набором данных.

Теперь, когда мы загрузили изображения и аннотации в одном файле CSV, который включает в себя ограничивающие рамки, нам нужно конвертировать аннотации в формат YOLO. Как мы помним, в формате YOLO

рядом с файлом изображения у нас должен быть текстовый файл с тем же именем, что и у файла изображения.

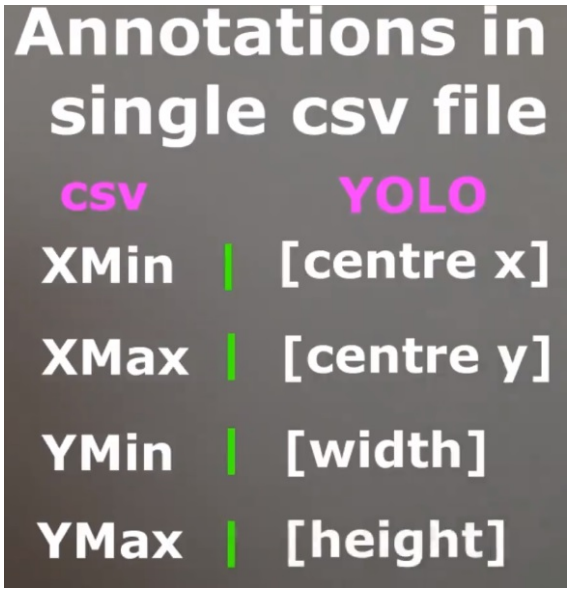

Рисунок 2.17: CSV-файл и папка с набором данных.

У нас есть две задачи для реализации:

- нам нужно создать текстовые файлы рядом с каждым недавно загруженным изображением с такими же именами как файлы изображений имеют;

- нам также нужно преобразовать аннотации ограничивающих рамок, которые находятся внутри одного CSV файл для всех изображений и для всех объектов в формате YOLO.

И, конечно же, нам нужно писать внутри созданных текстовых файлов преобразованные аннотации ограничительных рамок.

Еще одно важное замечание, которое стоит упомянуть. Теперь, когда мы загрузили изображения с помощью инструментария и проверили оригинальные CSV аннотации визуализатором нам нужно изменить имя папки. В этом случае мы исключаем будущие возможные ошибки. Итак, просто переименовываем его.

Блок 1

Мы устанавливаем полные пути к CSV-файлам и изображениям. Мы уже сделали это.

В следующем блоке мы готовим список классов, для которых мы хотим получать аннотации.

Мы скачали класс для Человека.

После того, как список необходимых классов готов мы читаем первый CSV-файл, чтобы получить из него зашифрованные строки для имён классов. Мы используем функцию чтения CSV и указываем полный путь к файлу и его имени.

Блок 2

Мы реализуем все необходимые вычисления для преобразования данных разметок в формат YOLO.

Во-первых, мы создаём пустые новые столбцы для формата YOLO, это номер класса, центр объекта в x, центр объекта в y, ширина объекта и высота объекта.

Затем мы перебираем все зашифрованные строки в sub-dataFrame и присвоить новому столбцу соответствующий номер класса в соответствии с положением зашифрованной строки в списке. Мы делаем это методом loc, который мы применяем к sub-dataFrame, и в первом аргументе мы находим все строки из sub-dataFrame с текущей зашифрованной строкой.

Используя второй аргумент, мы указываем, с каким столбцом мы хотим работать, а именно с номером класса. И применяя символ присваивания, мы присваиваем всем расположенным строкам в столбце номер класса, позицию зашифрованного строка в списке, например 0, 1, 2, как у нас есть три класса.

Затем мы рассчитываем центр объекта в x и центр объекта в y для всех строк одновременно. И мы сохранить результаты в новых столбцах центр x и центр y. Для расчёта центральной точки нам просто нужно сложить данный максимум x с минимальным x и делим результат на 2. То же самое мы делаем для оси y.

После этого мы рассчитываем ширину и высоту для всех строк одновременно, а также сохраняем результаты в новые столбцы ширина и высота. Вычислять, нам просто нужно вычесть максимум из минимума для осей x и y соответственно.

Теперь, когда мы вычислили все необходимые числа в формате YOLO, мы извлекаем sub-dataFrame просто необходимые столбцы и еще один столбец с ImageID. Мы снова используем здесь метод loc, который применяется к sub-dataFrame, говоря, что нам нужны все строки, но только эти столбцы.

Блок 3

Мы записываем рассчитанные аннотации в формате YOLO в текстовые файлы и сохраняем их рядом с каждым изображением.

Мы будем использовать метод ходьбы, который перебирает все подкаталоги и все файлы, начиная с текущего каталога. Поэтому, и, прежде всего, нам нужно изменить текущий каталог на тот, где у нас есть наши изображения. Для этого мы используем «method chdir» и указываем необходимый каталог, который мы нашли с самого начала.

Теперь мы готовы извлечь окончательный полученный под-фрейм данных, в котором есть только необходимые столбцы для формата YOLO.

Мы снова делаем это методом loc, говоря, что нам нужны все строки, но только эти столбцы. Мы готовим путь, где сохранить TXT-файл с аннотациями, который является таким же, где изображение само по себе и имя TXT-файла, который совпадает с именем файла изображения. Наконец, мы сохраняем аннотации из извлечённого окончательного sub-dataFrame в текстовый файл с помощью метода CSV, указав путь, сказав, что нам не нужна строка заголовка и нам не нужен индексный столбец, а также указав разделитель, который является пробелом, который используется в формате YOLO.

После запуска скрипта мы видим, что рядом с каждым изображением есть TXT-файл с тем же именем. Мы можем видеть аннотации ограничительных рамок в формате YOLO.

Отлично, мы успешно преобразовали скаченные с другого датасета аннотации в формат YOLO и создали текстовые файлы рядом с каждым изображением.

Запустим инструмент LabelImg для маркировки изображений в формате YOLO и проверим наши преобразованные файлы.

В следующей главе мы будем мы объединять два подготовленных набора данных для обучения в среде Darknet.

2.4.1 Объединение созданного вручную набора данных со скаченным набором аннатонированных данных в свободном доступе

В предыдущей главе мы подготовили файлы, необходимые для обучения в среде Darknet.

Это было сделано для нашего пользовательского набора данных, который мы скачали. В предыдущем разделе мы также подготовили файлы, необходимые для обучения. Но мы сделали это для уже аннотированного набора данных.

Мы уже знаем, что для обучения YOLO версии 3 в среде Darknet нам нужно подготовить определенные файлы.

Первый файл, который мы собираемся создать, это файл данных. Имеет количество объединённых классов, пути к изображениям из и пользовательские наборы данных, которые будут использоваться для обучения и проверки во время обучения, путь к файлу, который присоединился к именам классов и папке, в которой можно сохранить тренировочные веса.

Имена точек классов файлов имеют имена объединённых объектов из маркированных и пользовательских наборов данных. Имена должны быть уникальными и не повторять друг друга из двух наборов данных.

Вот почему нам также необходимо обновлять аннотации в каждом текстовом файле. Нам нужно изменить номер класса согласно новому объединённому списку объектов из двух наборов данных.

Мы скопировали изображения из загруженного пользовательского набора данных и изображения из уже аннатонированного набора данных все вместе в одном папки. Все эти изображения имеют текстовые файлы аннотации в своих оригинальных папках.

Файл с именем train.py, и мы будем использовать его для создания файлов с полными путями к изображениям для обучения и проверки во время обучения для объединённого набора данных

Алгоритм заключается в следующем:

Мы устанавливаем полный путь к объединённым изображениям.

Далее мы подготовим список, в котором есть все полные пути к изображениям. После этого, мы делим список на две части, экономия 85% на обучении и 15% на валидацию во время обучения. Затем мы создаём текстовые файлы и написать пути каждый в новой строке.

В первом блоке мы устанавливаем полный путь к каталогу, в котором находятся изображения.

В следующем блоке мы собираем все пути в список. Но, во-первых, мы меняем текущий каталог на тот, к которому мы присоединились изображений с помощью метода chdir и указываем необходимый каталог.

После этого мы просматриваем все найденные файлы изображений, проверяя два типа расширений, потому что теперь у нас они разные. Затем подготавливаем полный путь к этому изображению и добавляем е его в список.

Когда все пути к изображениям были найдены,

Мы делим список на две части. 15% для проверки во время обучения и 85% на обучение.

В конце, мы создаём текстовые файлы и пишем строки с полными путями.

Мы не указываем путь для сохранения файла, просто имя, потому что мы уже изменили текущую работу каталог с помощью метода chdir.

Мы собрали файлы из двух наборов данных в одну папку и успешно создали файлы train.txt и test.txt для этого объединённого набора данных.

Теперь мы собираемся перейти к коду и создать файлы join.data, файл classes.names и создавать новые обновлённые аннотации в текстовых файлах рядом с каждым изображением, имеющим соответствующий номер класса согласно новому **объединённому** списку классов.

Это файл с именем join-name.py, и мы будем использовать его для создания необходимых файлов для обучения в рамках **Darknet.**

Алгоритм заключается в следующем:

Мы создали полные пути, которые мы сделали только сейчас.

После этого читаем уже существующие файлы classes.name помеченного набора данных и загруженного пользовательского набора данных. Затем мы создаём файл classes.names для объединённого набора данных, который имеет только уникальные и неповторяющиеся имена классов из двух наборов данных.

Далее мы создаём файл join.data это имеет пять строк с необходимой информацией для обучения в среде Darknet.

Наконец, мы читаем текстовые файлы аннотаций из двух исходных папок для помеченного набора данных и загружаем пользовательский набор данных, затем обновляем номера классов в соответствии с новым файлом classes.names для объединённого набора данных и создаём новые текстовые файлы аннотаций рядом с каждым изображением в объединённой папке набора данных.

После объединения убедимся, что мы все сделали правильно.

Мы снова будем использовать инструмент LabelImg.

Это открытая папка с изображениями и текстовыми файлами аннотаций, которые мы только что создали для объединённого набора данных.

Чтобы проверить, обновили ли мы аннотации, а именно номера классов и в общем объединили два набора данных правильно с помощью инструмента LabelImg необходимо создать txt файл с именем classes.txt

Мы можем создать его с помощью любого редактора, например, блокнота или любого другого. Создайте текстовый файл с назовите classes.txt и копируем все строки из файла classes.names

Мы видим, что обновление текстовых файлов аннотаций для объединённого набора данных выполнено правильно.

Мы присоединились два набора данных и подготовили необходимые файлы для обучения в среде Darknet.

2.5 Используем фреймворк Darknet для обучения Object Detection на основе YOLOv3

Мы успешно создали пользовательские наборы данных и готовы начать обучение.

Мы пометили наш собственный набор данных. Мы создали собственный набор данных из датасета скаченного из интернета. И мы преобразовали эти наборы данных в формат YOLO. В этой главе мы установим и будем использовать фреймворк Darknet для обучения Object Detector на основе YOLOv3. **Darknet** – это инфраструктура нейронной сети с открытым исходным кодом, которая поддерживает вычисления CPU и GPU.

Мы подготовим все необходимые файлы для обучения. Настроим файл конфигурации. Запустим тренировочный процесс, и результаты теста

обнаружения объектов на изображении и видео опубликуем в главе 3. Для тренировки мы будем использовать два пользовательских набора данных, по одному. Первый, который наш собственный датасет, где Phoca Largha. Второй, мы объединились с датасетом скаченным из интернета.

Мы создали эти пользовательские наборы данных в предыдущих разделах. Давайте начнем и обучим детектор объектов YOLO используя наши данных.

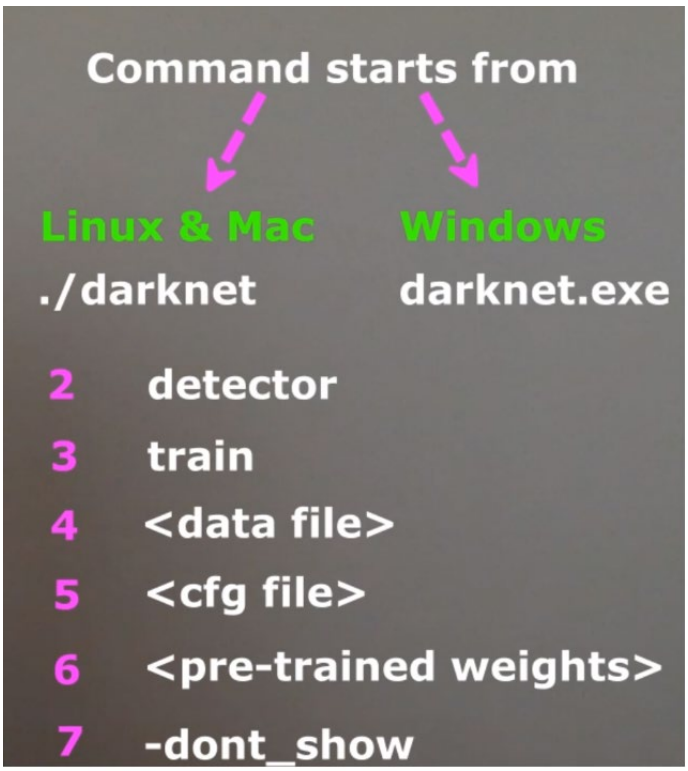

Рисунок 2.4: Команда для начала обучения в Darknet.

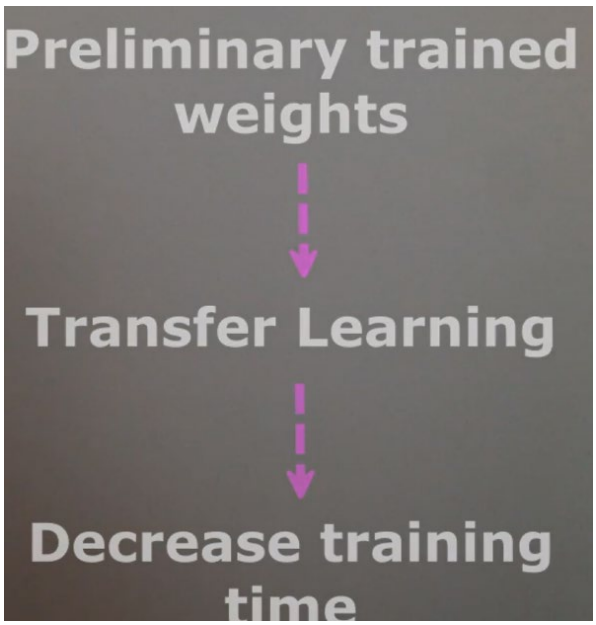

Рисунок 2.5: Использование предобученной модели для уменьшения времени обучения с нашим набором данных.

Мы полностью готовы начать обучение в рамках Darknet.

Подведя итоги того, что мы сделали в этой главе, и какие новые навыки мы получили.

Мы успешно загрузили уже помеченные изображения из огромного существующего набора данных.

Мы узнали, как использовать специальный инструментарий, который делает загрузку максимально простой.

Мы получили новый навык для загрузки необходимого фрагмента данных.

Мы также преобразовали аннотации, написанные в CSV-файл, в формат YOLO.

Мы получили еще один навык для преобразования разметки изображений. Мы объединили два подготовленных набора данных, помечены из предыдущих глав, и загружены пользовательские для обучения все вместе в среде Darknet.

3 Результаты выполненной работы

3.1 Проверка гипотезы использования стандартных инструментов OpenCV

На первом этапе выпускной квалификационной работы мы решили провести эксперимент с проблемой обнаружения объектов на видео штатными инструментами OpenCV (по цветовой маски объекта).

Алгоритм поиска цветовой маски:

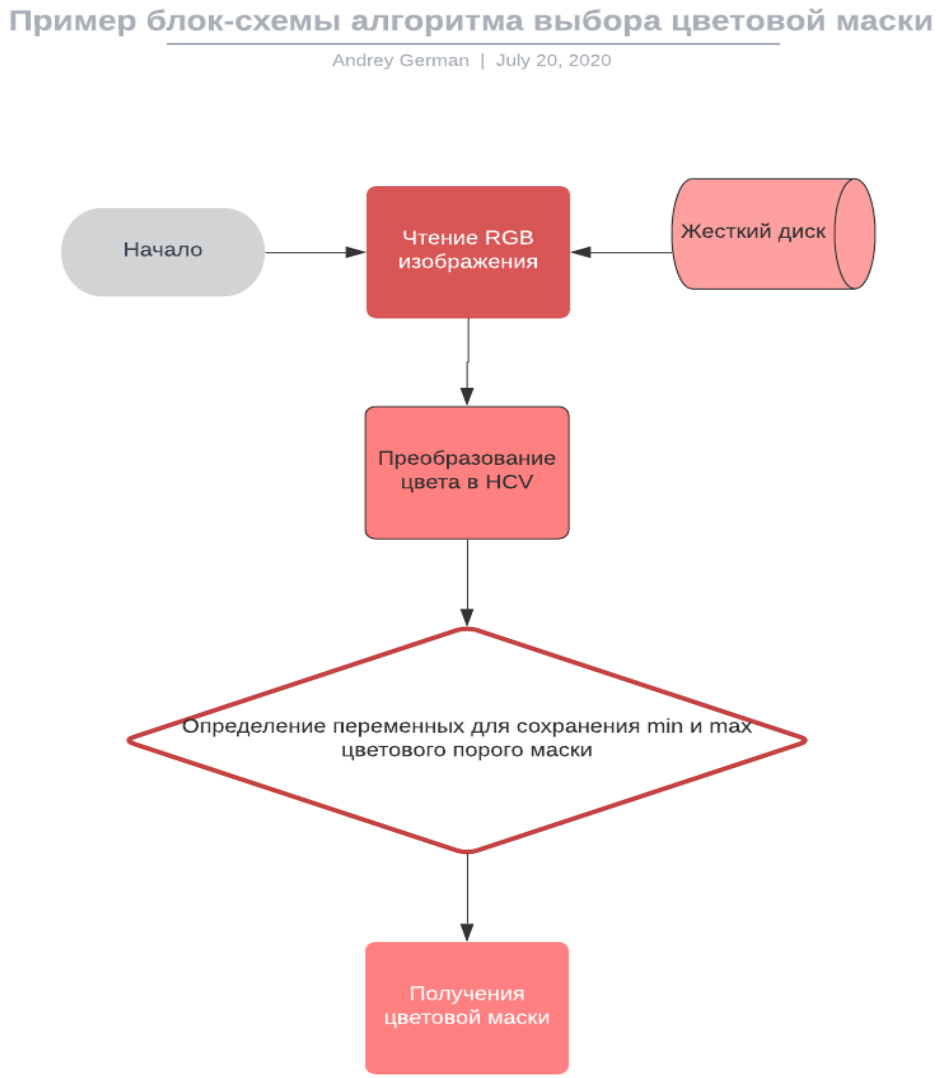

Рисунок 3.1: Алгоритм получения цветовой маски объекта

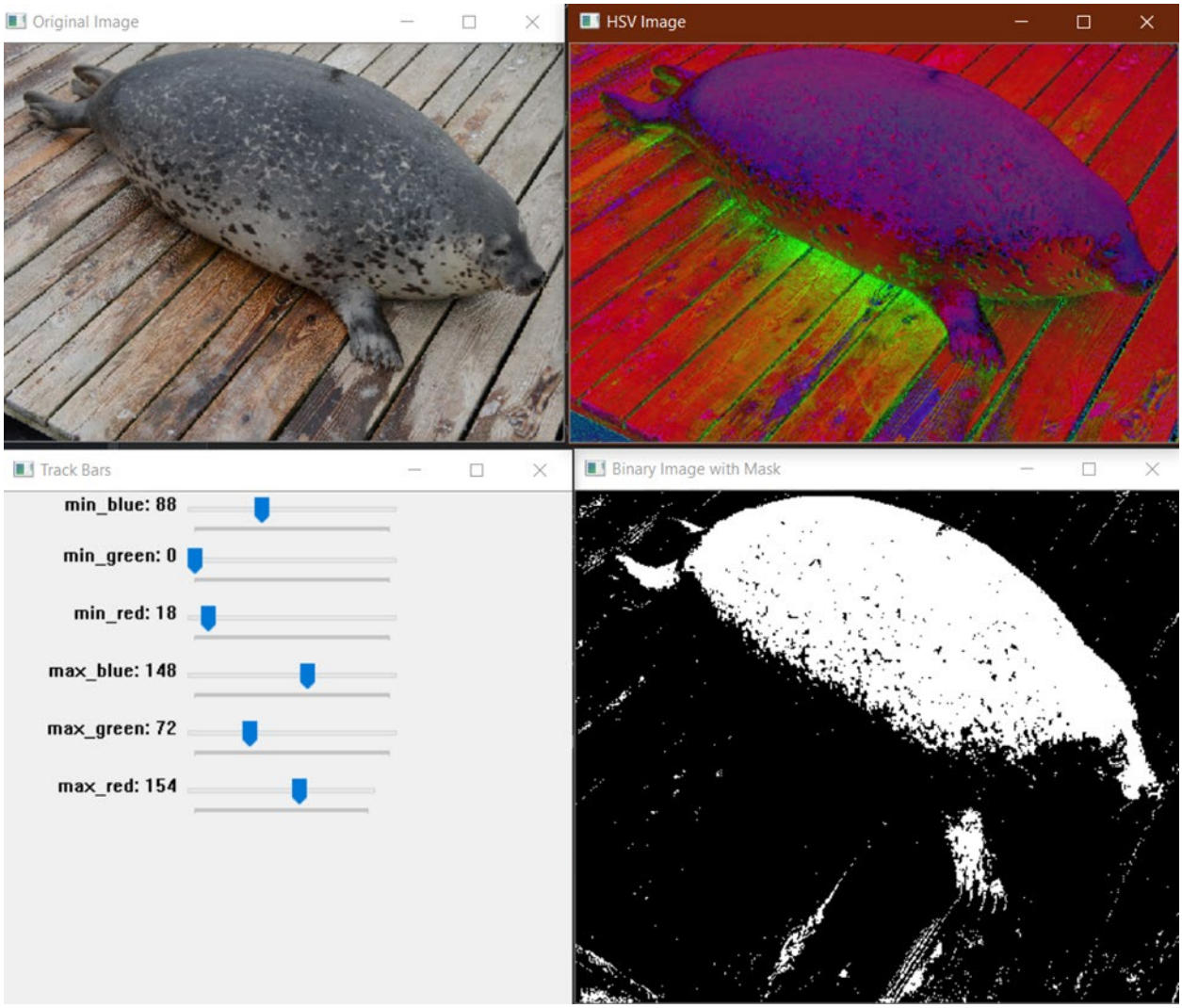

Рисунок 3.2: с помощью трекбола мы находим правильные шесть чисел для цветовой маски

В результате получаем цветовую маску дальневосточной ларги

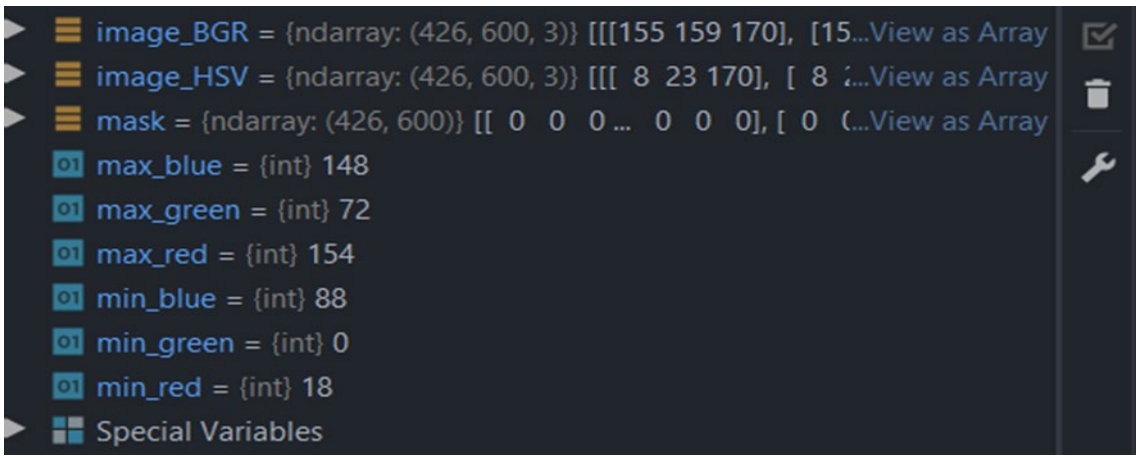

Рисунок 3.3: Нашли необходимую маску для детекции ларги в режиме реального времени на веб камере.

После получения цветовой маски было решено протестировать точность определения ларги на архивном видеофайле предоставленным Национальным научным центром морской биологии ДВО РАН

Алгоритм обнаружения при помощи порога цветовой маски:

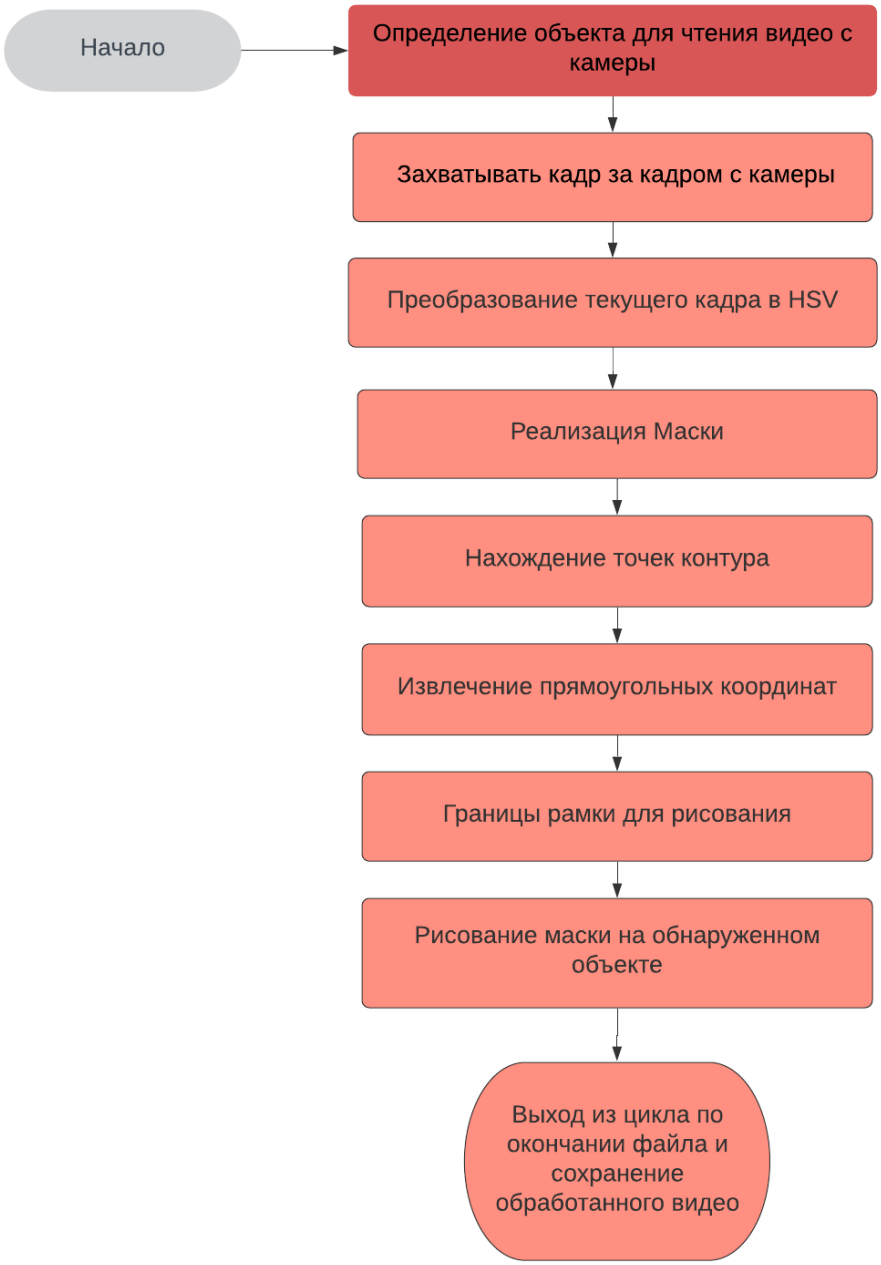

Рисунок 3.4: Алгоритм обнаружения при помощи порога цветовой маски

Тестирование проходило на:

ОС: Windows 10

– GPU: NVIDIA RTX 2070.

– CMake 3.14 - расширяемая система с открытым исходным кодом для управления процессом сборки приложений в операционной система независимо от компилятора.

– CUDA Toolkit 10.0 - среда разработки для создания высокопроизводительных приложений с методами ускорения GPU.

– OpenCV 3.4.6 - open-source библиотека для компьютерного зрения и машинного обучения, насчитывающая более 2500 оптимизированных алгоритмов.

– cuDNN 7.5.0 для CUDA Toolkit - библиотека, обеспечивающая ускоренную работу GPU в нейронных сетях.

Вывод:

На основании тестирования средняя точность обнаружения объекта для всего видеофайла оказалась не более 10% .

Рисунок 3.5: Обнаружение при помощи порога цветовой маски

Результат был признан неудовлетворительным. На основании различных источников было принято решение продолжить разработку программы по обнаружению дальневосточных ларг на основании нейронной мети YoloV3.

3.2 Результаты применения нейронной сети YOLOv3

YoloV3 по различным опубликованным тестам не уступает по точности таким системам обнаружения как RetinaNet [48], SSD [18], при этом по времени непрерывной работы обгоняет многие известные сети в несколько раз (результаты приведены в работе [49]). Более того, в YoloV3 есть возможность выбирать оптимальное соотношение скорости выполнения и точности, просто меняя размер модели без переобучения.

3.2.1 Описание датасета

В качестве обучающих данных был создан датасет дальневосточных ларг собранных из свободных источников и предоставленными изображениями Национального научного центра морской биологии ДВО РАН, содержащий 984 изображений (шт.). После аугментации с применением методов (**horizontal flip, random crop**) набор выборки увеличился до 5000 изображений для начала обучения нейронной сети.

Всего в наборе данных представлено 3 класса (Phoca Largha, Food box, Person). Каждому из 5000 изображений датасета (4250 из которых - обучающие данные, 750 - тестирующие), присвоен идентификатор объекта и координаты прямоугольника, в котором он содержится, для каждого объекта новая строка:

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

где <object-class> - целое число от 0 до (кол-во классов - 1);

<x_center> <y_center> <width> <height> - координаты центра прямоугольника, в котором находится объект, его ширина и высота, значения в промежутке от 0.0 до 1.0.

3.2.2.Рабочий процесс обучения нейронной:

Рисунок 3.6: Начало рабочего процесса обучения нейронной сети YoloV3

Рисунок 3.6: Проверка обученной модели нейронной сети YoloV3 на тестовую выборку.

В результате обучения нейронной сети YoloV3 точность обнаружения объекта на тестовой выборке составила 97.2%.

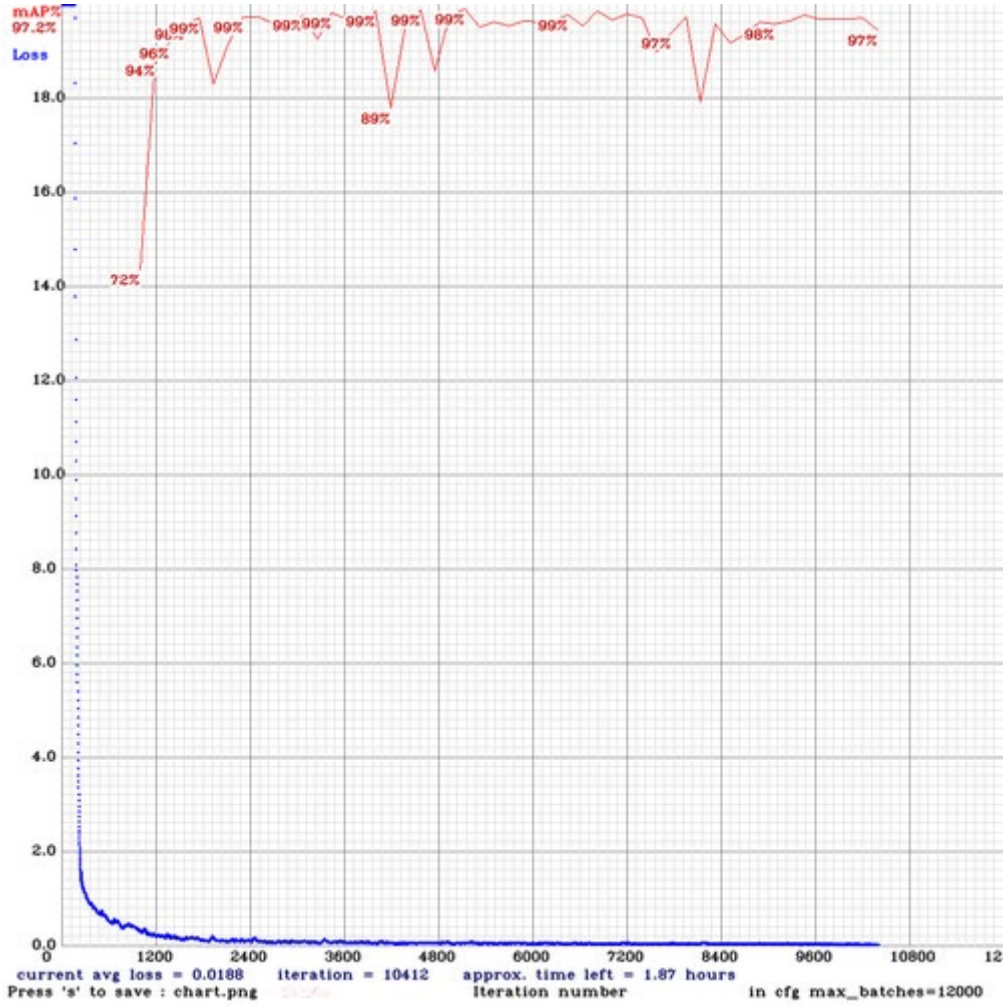

Рисунок 3.7 График обучения с mAP(97.2%) на тестовой выборке.

Как описано в разделе 1.2.6, оптимум mAP может быть найден путём нескольких итераций и оценки производительности модели тестовому набору.

На рисунке 3.7 Оптимум был найден при 9600 итераций с соответствующими mAP 97.2% на тестовой выборке.

3.2.3 Результаты работы детектора

Представлены результаты работы детектора, чтобы продемонстрировать решённые задачи, поставленные в разделе 1.1.

Рисунок 3.8 Обнаружение YoloV3 на изображении.

Рисунок 3.8 Обнаружение YoloV3 на видео.

Рисунок 3.8 Обнаружение YoloV3 классов Phoca Largha и Person.

Сравнение обработки видеофайла:

Обработка на CPU	Обработка на GPU
Время обработки файла составила 29,9 минут FPS : 1.9	Время обработки файла составила 5,9 минут FPS : 9.67

Тестовый файл был предоставлен ДВО РАН и составляет 3445 кадров, время 4,47 минут, объём (367Mb).

Как мы видим при использовании GPU удалось достичь приемлемого FPS, что позволяет обрабатывать видео в реальном времени.

Все программные модули были реализованы на языке программирования Python 3.6 с использованием различных библиотек глубокого обучения.

Исходный код обнаружения объекта на видео выложен на GitHub.

https://github.com/Andger1975/Largha-YoloV3.git

Заключение

Таким образом, в ходе выполнения данной выпускной квалификационной работы были достигнуты следующие результаты:

– Рассмотрены различные модели обнаружения объектов на изображении;

– создан набор данных дальневосточных ларг для работы с YOLO;

– проведён обзор фреймворка Darknet и системы обнаружения YOLO;

– обучена Нейронная сеть YoloV3;

– получены данные для оценки возможности применение обученной модели для обнаружения объекта в реально времени.

Полученные данные будут использованы для дальнейших исследований в рамках совместного проекта по распознаванию дальневосточных ларг Школы цифровой экономики и Национального научного центра морской биологии ДВО РАН.

Список использованных источников

1. Md. Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Mahmudul Hasan, Brian C. Van Esesn, Abdul A. S. Awwal, and Vijayan K. Asari. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches . CoRR, abs/1803.01164, 2018. URL http://arxiv.org/abs/1803.01164. Cited on page 10.

2. Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection viaregion-based fully convolutional networks. In Advances in neural information processing systems, pages 379–387, 2016. Cited on pages 10 and 14.

3. Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA : An Open Urban Driving Simulator. In Proceedings of the 1st Annual Conference on Robot Learning, pages 1–16, 2017. Cited on pages 1 and 21.

4. Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. 2016. Cited on page 8.

5. Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge . International Journal of Computer Vision, 88(2):303–338, June 2010. Cited on pages 15, 16, and 21.

6. Ross Girshick. Fast R-CNN. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1440–1448, Dec 2015. doi: 10.1109/ICCV.2015. 169. Cited on pages 10, 11, and 12.

7. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014. Cited on pages 10 and 11.

8. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org. Cited on pages 5, 6, 8, 9, 15, and 39.

9. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and

Pattern Recognition (CVPR), pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90. Cited on page 26.

10. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989. Cited on page 6.

11. Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning – Volume 37, ICML'15, pages 448–456. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045167. Cited on page 14.

12. Sezer Karaoglu, Yang Liu, and Theo Gevers. Detect2rank: Combining object detectors using learning to rank. IEEE Transactions on Image Processing, 25(1):233 248, Jan 2016. ISSN 1057-7149. doi: 10.1109/TIP.2015.2499702. Cited on page 28.

13. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. Cited on pages 10 and 25.

14. Yann Le Cun, Leon Bottou, and Yoshua Bengio. Reading checks with multilayer graph transformer networks. In Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on, volume 1, pages 151–154. IEEE, 1997. Cited on page 10.

15. Peilun Li, Xiaodan Liang, Daoyuan Jia, and Eric P. Xing. Semanticaware Grad-GAN for Virtual-to-Real Urban Scene Adaption. CoRR, abs/1801.01726, 2018. URL http://arxiv.org/abs/1801.01726. Cited on page 39.

16. Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. 2013. Cited on page 9.

17. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common

Objects in Context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 740– 755, Cham, 2014. Springer International Publishing. ISBN 978-3-319- 10602-1. Cited on pages 15, 18, and 37.

18. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision – ECCV 2016, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0. Cited on pages 1, 10, 12, 14, 25, 31, and 38.

19. Yusuke Niitani, Toru Ogawa, Shunta Saito, and Masaki Saito. ChainerCV: a Library for Deep Learning in Computer Vision. 10 2017. doi: 10.1145/ 3123266.3129395. Cited on pages 25 and 26.

20. Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6517 6525, 2017. Cited on pages 10, 14, and 37.

21. Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. CoRR, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804. 02767. Cited on pages 1, 14, 15, 29, and 37.

22. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779– 788, 2016. Cited on pages 12, 13, 14, and 15.

23. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015. Cited on pages 10, 11, 12, 13, and 14.

24. Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for Benchmarks. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 2232–2241, 2017. doi: 10.1109/ ICCV.2017.243. URL https://doi.org/10.1109/ICCV.2017.243. Cited on page 2.

25. Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556. Cited on pages 10 and 11.

26. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929– 1958, 2014. Cited on page 10.

27. Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. Cited on page 1.

28. Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In European conference on computer vision, pages 818–833. Springer, 2014. Cited on page 10.

29. Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review . CoRR, abs/1807.05511, 2018. URL http://arxiv.org/abs/1807.05511. Cited on page 14.

30. Muhlmann, K., Maier, D., Hesser, J., & Manner, R., "Calculating dense disparity maps from color stereo images, an efficient implementation." International Journal of Computer Vision, Vol. 47, 2002, pp.79-88.

31. Shi, J., & Malik, J., "Normalized cuts and image segmentation." Departmental Papers (CIS), 2000, pp.107.

32. Felzenszwalb, P. F., & Huttenlocher, D. P., "Efficient graph-based image segmentation." International journal of computer vision, Vol.59, No.2, 2004, pp.167 181.

33. Pal, N. R., & Pal, S. K., "A review on image segmentation techniques." Pattern recognition, Vol. 26, No.9, 1993, pp.1277-1294.

34. Hull, J. J., "A database for handwritten text recognition research." IEEE Transactions on pattern analysis and machine intelligence, Vol. 16, No. 5, 1994, pp.550-554.

35. Kim, G., Govindaraju, V., & Srihari, S. N., "An architecture for handwritten text recognition systems." International Journal on Document Analysis and Recognition, Vol.2, No.1, 1999, pp.37-44.

36. Chang, S. L., Chen, L. S., Chung, Y. C., & Chen, S. W., "Automatic license plate recognition." IEEE transactions on intelligent transportation systems, Vol.5, No.1,2004, pp.42-53.

37. Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., & Kayafas, E., "Alicense plate-recognition algorithm for intelligent transportation system applications." IEEE Transactions on Intelligent transportation systems, Vol. 7, No.3, 2006, pp.377-392.

38. Jensen, J. R., & Lulla, K., "Introductory digital image processing: a remote sensing perspective.",1987.

39. LeCun, Y., Bengio, Y., & Hinton, G., "Deep learning." nature, Vol.521, No.7553, 2015, pp.436.

40. Schmidhuber, J., "Deep learning in neural networks: An overview." Neural networks, Vol.61, 2015, pp.85-117.

41. Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A., "The pascal visual object classes challenge: A retrospective." International journal of computer vision, Vol. 111, No.1, 2015, pp.98-136.

42. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L., "Imagenet: A largescale hierarchical image database." IEEE conference on computer vision and pattern recognition, 2009, pp. 248-255.

43. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., & Fei-Fei, L., "ILSVRC-2012.", 2012.

44. Davis E. King. "Dlib-ml: A Machine Learning Toolkit". In: J. Mach. Learn. Res. 10 (Dec. 2009), pages 1755–1758. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm? id=1577069.1755843 (cited on page 61).

45. A. Dutta, A. Gupta, and A. Zissermann. VGG Image Annotator (VIA). http://www.robots. ox.ac.uk/~vgg/software/via/. 2016 (cited on page 62).

46. https://github.com/gwding/draw_convnet/

47. https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/

48. Focal Loss for Dense Object Detection / Tsung-Yi Lin, Priya Goyal, Ross Girshick et al. — arXiv, 2017.

49. Redmon Joseph, Farhadi Ali. YOLOv3: An Incremental Improvement. — arXiv, 2018.