Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(государственный университет)»

Физтех-школа радиотехники и компьютерных технологий

Факультет радиотехники и кибернетики

Кафедра физико-технической информатики

Разработка алгоритмов быстрого обнаружения объектов на основе небольшого количества реальных размеченных данных

Выпускная квалификационная работа
(бакалаврская работа)

Направление подготовки: 03.03.01 Прикладные математика и физика

Выполнил:
студент 616а группы _____ Башаров Илья Валерьевич

Научный руководитель:
к.ф.н. _____ Рыков Владимир Васильевич

Москва 2020

Содержание

Аннотация

В работе рассматриваются различные методы детектирования в условиях малого количества реальных аннотированных данных. Для решения поставленных задач разработаны программные модули для распознавания объектов на изображениях. Приводится скорректированный для этой задачи алгоритм обучения модуля Nuisance Disentangled Feature Transform. Эксперементально обосновывается эффективность обучения, используя модуль Nuisance Disentangled Feature Transform. Исследуется минимально необходимое соотношение реальных данных в выборке для достижения максимально возможного качества.

Аббревиатуры

1. Domain Adaptation – преобразование домена данных.
2. Non-Maximum Suppression (NMS) – подавление немаксимумов.
3. Nuisance Disentangled Feature Transform (NDFT) – помехоустойчивый модуль преобразования признаков.
4. Frames per second (FPS) – количество кадров в секунду.
5. Region-based Convolutional Neural Network (R-CNN) – свёрточная нейронная сеть на основе предложенных регионов.
6. Region Proposal Network (RPN) – сеть предложений регионов.
7. Intersection over Union (IoU) – пересечение над объединением.
8. WordNet – лексическая база данных английского языка в виде электронного тезауруса.
9. БПЛА – беспилотный летательный аппарат.
10. Region of Interest (ROI) – регион интересов.
11. Feature Pyramid Network (FPN) – функциональная пирамидальная сеть.
12. Receiver operating characteristic (ROC) – рабочая характеристика приемника.
13. Average Presicion – средняя точность.
14. Backpropagation – обратное распространение ошибки.
15. Loss – функция ошибки.
16. Baseline – базовый уровень.
17. Software Development Kit (SDK) – комплект для разработки программного обеспечения.
18. Learning rate – скорость обучения.
19. Stochastic gradient descent (SGD) – стохастический градиентный спуск.

1 Введение

Проблема недостатка данных для обучения стоит с самого начала революции нейронных сетей. Известно, что чем больше данных принял алгоритм, тем выше его качество. К примеру, существуют стандартизированные аннотированные коллекции данных, состоящих из миллионов примеров для нейросетевых алгоритмов. К большому сожалению, все наборы данных были размечены людьми, это очень трудоемкая работа. Ошибки, выявленные в результате этого кропотливого коллекционирования, очень дорого могут обойтись – градиенты, передающиеся с помощью метода обратного распространения ошибки в нейросети, могут переполнится и алгоритм ничему не научится.

XXI век – век прорывных технологий. Иженеры и ученые стремятся к автоматизации процессов, чтобы достигнуть наивышего качества. Как сократить время и ошибки при работе над аннотацией данных?

С другой стороны, уже давно существуют программы автоматического генерирования синтетических данных. Но, согласно последним экспериментам ученых-исследователей машинного обучения, нейронные сети очень легко подстраиваются под искуственные данные, не показывая высокого качетсва работы на реальных размеченных данных.

Почему так происходит? Наш мир гораздо богаче и порой состоит из сложных структур, которых очень сложно описать с помощью детерминированной программы.

Данной проблемой занимается такая область машинного обучения, как Domain Adaptation. Она объясняет, как обучить модель на данных из домена-источника (source domain) так, чтобы она показывала сравнимое качество на целевом домене (target domain).

Научная новизна заключается в постановке задачи минимизации и разработке нового состязательного спобоба обучения модели для сохранения высокого качества детекции на реальных данных, которых зачастую получить труднее, чем искуственно-созданных.

Рассмотренная мной **проблема актуальна** и имеет **прикладной характер**. Результаты, полученные в этой работе, могут использоваться в реальной жизни при разработке нейросетевых алгоритмов.

2 Общее описание задачи детекции и входных данных

2.1 Описание задачи

В данной работе рассматривается построение алгоритма, обрабатывающего в реальном времени 2D-изображения в условиях малого количества реальных данных. На выходе мы ожидаем список ограничивающих рамок, внутри которых локализован объект, уверенность классификации, а также тип объекта. Другими словами, алгоритм решает задачу детекции на 2D изображении. Наглядный пример – на рисунке 1.

Важно понять, что такое ограничивающая рамка. В качестве определения возьмем, что это минимальный по площади прямоугольник для набора точек, которые должны все лежать внутри него. Поскольку входное изображение двумерное, то прямоугольник определяется по 4 числам.

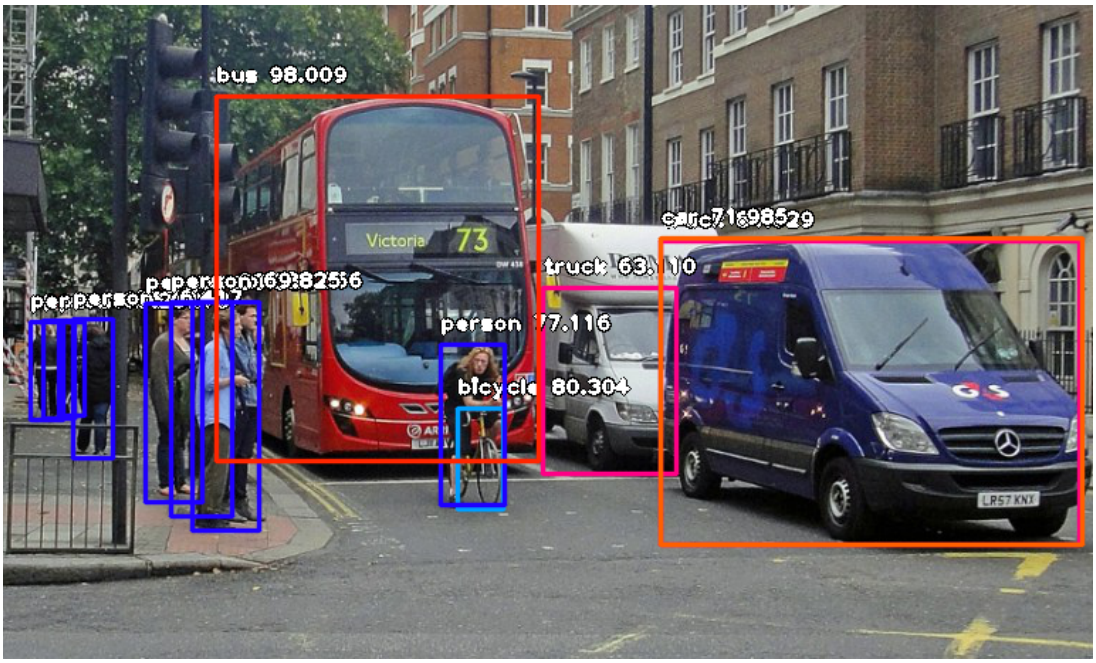


Рис. 1: Многоклассовая детекция.

2.2 Описание входных данных

Чтобы сделать представление данных более универсальным, были придуманы и стандартизированы форматы входных данных для задачи детекции. Наиболее часто применяемяемые форматы – MSCOCO [1], ImageNet [2], PascalVOC [3] и другие. Рассмотрим самые основные из них. Сравнительная характеристика представлена в таблице 1.

Название коллекции	Обучение	Валидация	Тест	Количество классов	Формат аннотации
MSCOCO 2017	118K	5K	41K	80	.json
PascalVOC 2012	6K	6K	–	20	.xml
ImageNet	1M	–	–	200	.xml

Таблица 1: Сравнение датасетов для детекции объектов.

MSCOCO

MicroSoft Common Objects in COntext (MSCOCO) – коллекция, имеющая иерархическую структуру. Имеет не только аннотацию для детекции, но и для сегментации, и других задач компьютерного зрения. Практически все исследования тестируются на этом наборе данных.

PascalVOC

Pattern Analysis, Statistical Modelling and Computational Learning Visual Object Classes (PascalVOC) – первая попытка создать соревнование для исследователей. Имеет разметку для секментации и детекции. Отсутсвие разделения на тестирующую выборку, а также малое количество аннотированных данных – главная причина, почему он в теряет свою популярность.

ImageNet

ImageNet – проект по созданию и сопровождению массивной базы данных аннотированных изображений, предназначенная для отработки и тестирования методов распознавания образов и машинного зрения. Был популярен в 2010-2015 годах, когда совернование по распознаванию образов по изображению [4] имело ценность для исследователей задачи классификации изображений. Создан в соответсвии с иерархией WordNet.

3 Обзор литературы

Концепция автоматического детективания появилась давно. Первым успешным примером алгоритма является детектор границ Канни [5]. Фильтр, основанный на градиентах и трассировке изображения, независимо от входных данных выделял объекты всех типов и размеров. Несколько лет назад появились методы обработки изображений с помощью нейронный сетей, которые превзошли по точности классические алгоритмы. На данный момент можно выделить 3 основных подхода к решению задачи детектировния объектов на основе неиросетевых алгоритмов:

1. Одноэтапные - алгоритмы детектирования You Only Look Once (YOLO) [6, 7, 8, 9], Single-shot Multibox Detector (SSD) [10] и др.

   Изображение бьется на ячейки равномерным образом. Извлекаются признаки из изображения. Далее с их помощью предсказываются ограничивающие рамки и карта принадлежности класса для каждой ячейки изображения. Исходя из взаимного расположения ячеек и рамок, которые их покрывают, приписываются классы. Отбрасываются ложные рамки с помощью алгоритма NMS.

   Плюсы – высокая скорость обработки изображения, в зависимости от модели и ее гиперпараметров, около 15-100 FPS.

   Минусы – по точности проигрывают другим подходам на известных коллекциях, например, [1].

2. Двуэтапные - алгоритмы детектирования R-CNN [11, 12, 13] и др.

   Извеченные признаки из изображения подаются в первичную регрессию и классификацию - RPN, далее отбрасываются ложные, близко-расположенные ограничивающие рамки. Те, что остались после работы алгоритма NMS, дополнительно уточняются другой нейросетью.

   Плюсы – хорошее качество на входных данных.

   Минусы – невысокая скорость обработки изображения, зачастую ограничивающая использование детектора в реальном времени.

3. Каскадные – каскадные R-CNN [14] и др.

   Более сложная каскадная архитектура, которая подстраивается под ошибки других уровней, помогает значительно увеличить качество, но сама модель содержит больше параметров, следовательно, становится тяжелее.

Плюсы – значительный прирост в качестве.

Минусы – не предназначена в реальном времени.

3.1 Быстрая R-CNN

Состоит из нескольких важных частей (см. Рис 2):

1. Нейронная сеть извлечения признаков.
2. Генератор анкерных областей – простая генерация областей, с различными соотношениями сторон ограничивающих рамок.
3. Сеть предсказания регионов – первичные кассификация и регрессия предложенных областей от генератора.
4. NMS – алгоритм, выбирающий лучшую предложенную ограничительную рамку из других рядом расположенных.
5. Пулинг ROI – слой, приводящий разные по ширине и высоте области к фиксированным для входа в нейронную сеть.

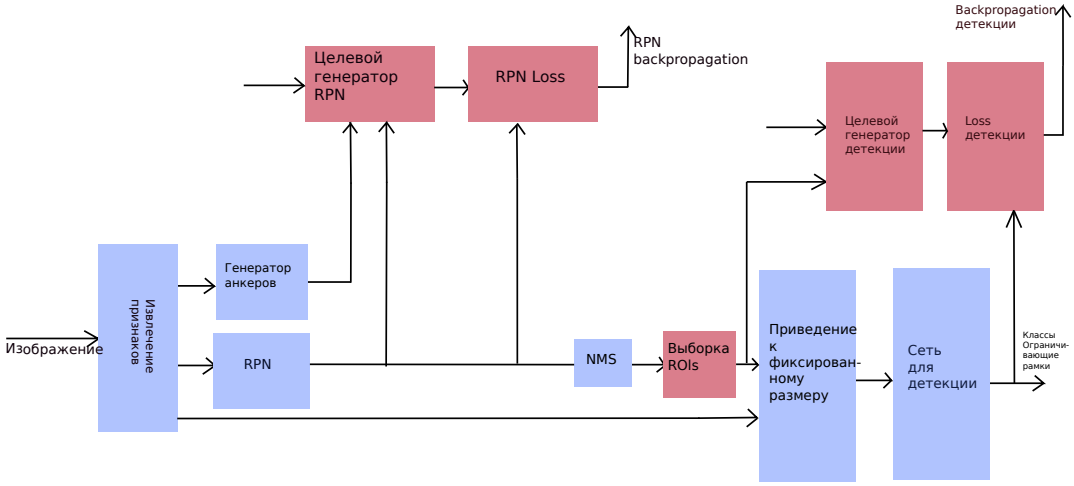


Рис. 2: Архитектура Faster-RCNN.

Нейронная сеть извлечения признаков

До 2015 года люди, создавая новые архитектуры сетей и развивая инфраструктуру высокопроизводительных систем, замечали, что чем глубже нейронная сеть, тем лучше она сможет извлекать признаки изображения. Существовали такие виды сетей, как AlexNet[22], ZF[23], VGG[24],

которые с каждым годом били рекоды на классическом ImageNet[2]. Но встала проблема – нельзя просто увеличивать количество конволюционных слоев и получать меньшие ошибки.

Это было связано в первую очередь с затуханием градиента в сети. Ошибка, полученная в конце сети, должна распространяться в начало, но так как в вклад в нее вносит каждый слой, первые слои глубокой сети практически не изменяются. Например, при исследователи GoogLeNet[25] решили, что лучше разные части сети обучать одинаково, подавая одновременно одни и те же данные, объединяя базисными (inception) слоями. Но ключевой в истории стала ResNet[26].

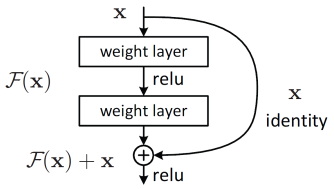


Рис. 3: Глубокая остаточная функция $\mathcal{F}$.

Оказывается, по своей природе, нейросети приходятся выучивать отображение $\mathcal{H}(x) = x$. Но проще получить остаточное возмущение $\mathcal{F}(x) = 0$ и добавить его ко входному тензору (см. Рис. 3). Авторы статьи [26] показывают, что благодаря такой структуре градиент затухает меньше на первых слоях, а также комбинируются признаки из разных репрезентативных слоев. С учетом сказанного, авторы смогли создать сеть с 152 слоями. Приведем наглядную стуктуру сети (см. Рис. 4).

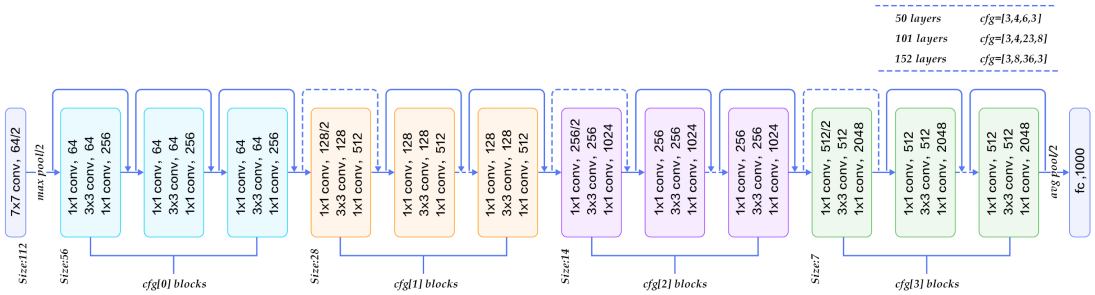


Рис. 4: Архитектура ResNet.

3.2 Модификация: FPN

Обнаружение объектов разных масштабов по фотографии является трудной задачей. Мы можем использовать признаки из сети для объектов одного масштаба (см. Рис. 5(b)), но создание разных сетей для разных размеров объектов дорогостояще по памяти и по времени.

Поэтому авторами FPN [17] было решено объединить признаки из разных слоев (см. Рис 5(d)). Подход состоит из двух путей: восходящего и нисходящего.

1. Путь снизу вверх – это сверточная сеть для извлечения признаков. По мере подъема пространственное разрешение уменьшается. Семантическое значение каждого слоя возрастает с увеличением глубины сети.

2. Путь сверху вниз необходим для построения слоев с более высоким разрешением из семантического богатого слоя.

Боковые связи между реконструированными слоями и соответствующими картами объектов необходимы для улучшения качества локализации объекта. Они также действует как пропуск соединений, чтобы облегчить обучение (подобно тому, что делает ResNet [26] (см. Рис. 4)).

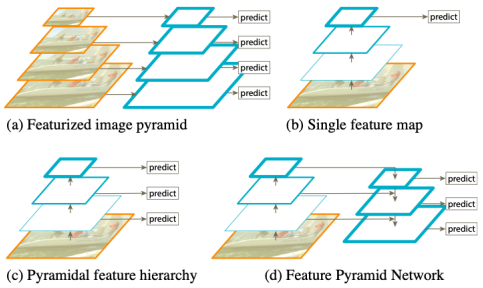


Рис. 5: Сравнение пирамид изображений.

Таким образом, FPN модуль надстраивается над сетью для извлечения признаков. В детекторе быстрая R-CNN (см. раздел **??**) карты признаков различных разрешений подаются в RPN для уточения локализации. Далее, исходя из заранее заданных констант, выбирается та карта с разшением, которая больше всего подходит под данную локализацию.

Безуслувно, часть проблем кроется в алгоритме NMS, поскольку основным фактором в нем является уверенность в предсказании, а не точность регрессирования ограничивающих рамок. Автоматический подбор констант в алгоритме подавления немаксимумов (нестрогий-NMS) [15], а также изобретение NMS – подобных алгоритмов, например, использующих предсказанное IoU [16], подняли качество.

Следущий этап в развитии компьютерного зрения, а конкретно задачи детекции, заключается в попытке настройки голов сети для классификации и регрессии ограничивающих рамок таким образом, чтобы они не коррелировали друг с другом. Предпосылкой этого послужила разная пространственная чувствительность этих модулей. Появились такие архитектуры, как двуголовая R-CNN [18], а также встраиваемая функция пространственного разделения задач классификации и регрессии ограничивающих рамок из статьи [19].

Но проблема основной проблемой алгоритмов является большое потребление входных данных для качественного решения задачи. Стало понятно, что такие коллекции данных, как [1, 2, 3], хотя и собраны, используя объекты из реальной жизни, но описывают не все её случаи. Простой пример: могут сменится погодные условия (туман / дождь /

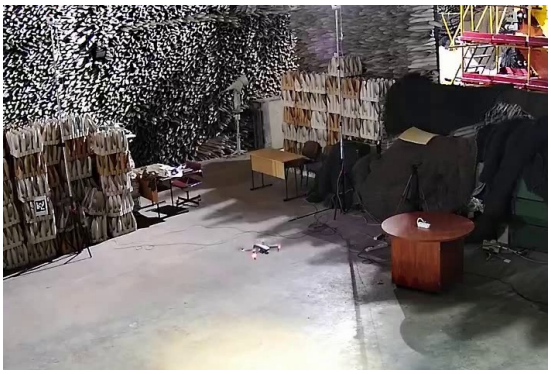
снег) – сеть будет ошибаться на таких примерах. С одной стороны, мы можем подобрать данные, покрывающие все виды ошибок. Но это потребует гораздо больше данных, а также ресурсов для исследований. Например, в статьях [20, 21] рассмотрены методы решения данной проблемы с помощью дополнительной классификации либо использования статистических методов машинного обучения, соответсвенно.

4 Постановка и анализ эксперимента

4.1 Данные

На данный момент для задачи детекции различных БПЛА нет готовых коллекций данных для обучения. С помощью камер видеонаблюдений было отснято и аннотировано вручную 1K и 2K изображений дронов Mavic-2 pro (см. Рис. 6b) и Phantom (см. Рис. 6a) от производителя DJI, соответсвенно. Поскольку полученных данных не хватало для качественного обучения (модель переобучалась под них), было решено использовать модуль для генерации синтетических данных с использованием 3D моделей от производителя (см. Рис. 6(c, d)).

(a) БПЛА Phantom.



(b) БПЛА Mavic-2 pro.

(c) Искусственный БПЛА Phantom.

(d) Искусственный БПЛА Mavic-2 pro.

Рис. 6: Входные данные для экспериментов.

Заметим, модуль для генерации синтетических изображений БПЛА учитывал такие недостатки съемки, как масштаб, уровень освещенности, перспетивное преобразование, поворот объекта, зашумление данных, смещение цветовых каналов.

Важным следствием решения задачи детектирования БПЛА Mavic-2 pro является возможность управления дроном, поскольку только для него от производителя DJI имеется официальный SDK.

Итог: бесконечно много искуственных и фиксированное количество реальных изображений.

План экспериментов

1. Проверить, какое будет качество на тестовой выборке данных, если для обучения использовать только реальные изображения, либо только синтетически-созданные – **baseline**;

   Использование идей Domain Adaptation:

   (a) Проверить качество на коллекции из синтетически-созданных и реальных данных, а также проделать пункт 2;

   (b) Проверить качество с использованием модуля NDFT и алгоритма его обучения 1, а также проделать пункт 2;

2. Оценить количество минимально необходимых реальных данных для максимально возможного качества;

3. Исследовать возможность выхода детектора в режим работы реального времени.

4.2 Выбор опорной модели

Некоторое время назад рассмотрены методы YOLO и SSD на тех же коллекциях реальных и искуственных данных для БПЛА Phantom (см. Рис. 6(a, c)). Их высокой FPS и хорошей точности хватило, чтобы решить задачу с БПЛА Phantom. Но на задаче детектирования БПЛА Mavic-2 pro указанные выше модели провалились.

Полькольку архитектура быстрой R-CNN [13] с соответствующими доработками (см. разделы 3.2, 4.2.1) могла сравниться по скорости обработки с решением, обрабатывающим изображения в реальном времени, она была выбрана в качестве опорной модели для детектирования БПЛА Mavic-2 pro.

4.2.1 Модификация: NDFT

В целом эта задача относится к большому разделу машинного обучения Domain Adaptation. Так, доменом-источником является синтетические данные (см. Рис. 6(c, d)) а доменом-целью – реальные данные. (см. Рис. 6(a, b)). Основная сложность – тренировка данной модели.

Идея – будем дополнительно классифицировать синтетические и реальные изображения с помощью состязательного модуля, описанного в [20].

Добавив дополнительную голову для класификации изображния (см. Рис 7), мы изменили функцию потерь.

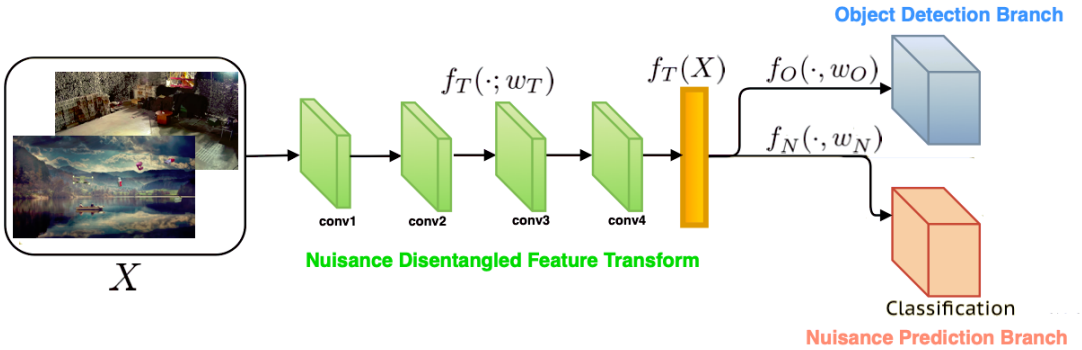


Рис. 7: Архитектура модуля NDFT.

Функция потерь представляет математическую формулировку сравнения предсказанных и истинных данных и значительно влияет на время обучения и достижимую точность решения поставленной задачи.

Задача минимизации функции потерь ставится следущим образом:

$$\min_{f_O, f_T} L_O(f_O(f_T(Y_T)), Y_O) - \gamma L_N(f_N(f_T(Y_T)), Y_N),$$
$$\min_{f_N} L_N(f_N(f_T(X)), Y_N),$$

где O - задача детектирования объекта, N – задача классификации изображения (является ли синтетической), T – задача извлечения признаков из изображения, $f_{O,N,T}$ – модули, выполняющие соответствующую задачу, $Y_{O,N}$ – обучающая выборка для соответствующей задачи, $L_{O,N}$ – функции потерь, γ – весовой коэффициент.

$$l_{1,smooth}(x) = \begin{cases} |x|, & |\mathrm{x}| > \mathrm{a} \\ \frac{1}{|a|}x^2 & |\mathrm{x}| < \mathrm{a} \end{cases}$$

Что берется в качестве функции потерь? Для задачи детектирования берется сглаженный $l_{1,smooth}$ (см. формулу 4.2.1). Зачастую в задачах классификации L_N импользуют softmax, но по словам авторов статьи [20] при состязательном обучении это вызывает взрывы градиентов. Заменой служит L_{ne} – отрицательная энтропия. Это способствует модели делать «неопределенные» прогнозы относительно домена изображения, тем самым, забывая его.

Таким образом,

$$\min_{f_O, f_T} L_O(f_O(f_T(Y_T)), Y_O) + \gamma L_{ne}(f_N(f_T(Y_T))),$$
$$\min_{f_N} L_N(f_N(f_T(X)), Y_N),$$

Представим обновленный алгоритм 1 обучения предложенного модуля, который позволит решить проблему данных разного домена для быстрого детектора R-CNN с модификацией NDFT.

Алгоритм 1 Обучение модуля NDFT с использованием состязательного обучения для задачи детекции.

INPUT: предобученные модули f_T, f_O, f_N
k - количество эпох для сброса весов у f_N

procedure LEARNING NDFT(f_T, f_O, f_N, k)
  for epoch in range(epochs) **do**
    Sample a mini-batch of n examples $[X_1, \cdots, X_n]$
    Update $f_T(w_T)$ and $f_O(w_O)$ with gradients:
    $\nabla_{w_T, w_O} \frac{1}{n} \sum_{i=0}^{n} L_O(f_O(f_T(Y_T^i)), Y_O) + \gamma L_{ne}(f_N(f_T(Y_T^i)))$
    while predictions from f_N have training accuracy ≤ 0.9 **do**
      Update $f_N(w_N)$ with gradients:
      $\nabla_{w_N} \frac{1}{n} L_{ne}(f_N(f_T(Y_T^i)))$
    end while
    Restart f_N for every k iterations and repeat the Procedure again.
  end for
end procedure

4.3 Метрики качества для задачи детекции

«Говорят, что компьтерная программа обучается решению задачи T на основе опыта E, если качество решения по метрике P растет по мере накопления опыта E» [27]. Таким образом, задача T - предсказание ограничивающих рамок для объектов на изображении. Опыт E заключается в поиске закономерностей в представленных данных. А как же оценить наш накопленный опыт?

То, что невозможно измерить — очень сложно улучшить. Метрики качества показывают насколько хорошо данный алгоритм решает поставленную задачу. В простейшем случае с классификацией, самой просто

метрикой может быть подсчет правильно классифицированных изображений в процентом соотношении. Поскольку в общем виде задача детекции включает себя как и локализацию (предсказание рамок), так и классификацию (предсказание коэффициента уверенности и класса объекта), то необходимо определить функции качества для этих подзадач. Я разберу самые известные метрики для задачи детекции: Precision score [4.3.2], Recall score [4.3.2], IoU [4.3.1], Dice score[4.3.1], AP@[.5:.95] [4.3.3] и различные его модификации.

Важно понимать, что выбор метрики напрямую влияет на решение задачи. Если метрика будет выбрана не корректно, то задача может быть плохо или вовсе не решена.

4.3.1 Локализация

Задача ставится следущим образом: есть предсказанная, а также истинная ограничивающие рамки. Как нам понять, насколько хорошо наше предсказание описывает реальные данные?

IoU

IoU или индекс Жаккара [28] - первый известный коэффициент сходства (см. Рис. 8a). Нормированный. К приложению задач компьютерного зрения определяется как

$$IoU = \frac{|A \cap B|}{|A \cup B|},$$

где A и B измеримые множества.

Подставив предсказанную и истинную рамки вместо множеств, мы получим оценку сходства областей. Когда мы хотим оценить качество локализации на большом количестве примеров, необходимо найти соответсвие между предсказанием и истиной.

Плюсы: простая для понимания.

Минусы: по ней нельзя сравнивать алгоритмы, поскольку учитывается только количество тех наложений, у которых порог выше заданного. Следует отметить, что сам порог, определяемый исследователями для конкретных задач, вносит свою неопределенность, а также учет только хороших примеров предсказаний ограничивает в понимании качества работы алгоритма.

Dice score

Dice score (Sørensen–Dice coefficient) – аналог уже рассмотренной метрики 4.3.1 (см. Рис. 8b). Также нормирован, но используется реже. Можно выразить через IoU. Записывается как

$$Dice\ score = \frac{2|A \cap B|}{|A| + |B|},$$

где A и B измеримые множества.

К **Минусам** можно добавить, что метрика не интерпретируема.

(a) Метрика IoU. (b) Метрика Dice.

Рис. 8: Сравнение метрик локализации. Площадь, отмеченную красным цветом, необходимо измерить.

4.3.2 Классификация

Попробуем с помощью метрик классификации избавится от проблемы учета только «хороших» примеров. Будем классифицировать ограничивающие рамки на 3 подвида: **True Positives** (TP), **False Negatives** (FN), **False Positives** (FP). Ознакомиться с определениями можно, используя Рис. 9.

Рис. 9: Классификация органичивающих рамок.

Разберем основные метрики, которые используются в задаче детекции. Необходимо заметить, что их описание ограничивается одним классом.

Точность

Точность (precision) – это доля правильных предсказанных примеров ко всем предсказанным. С учетом обозначений,

$$Presicion = \frac{TP}{TP + FP}$$

Таким образом, точность варьируется от 0 до 1, высокая точность подразумевает, что большинство предсказанных ограничивающих рамок соответствуют истине. Если необходимо уменьшать количество ложных срабатываний, то следует следить за изменением precision во время обучения модели.

Минусы данной метрики очевидны: она не учитывает ситуацию, когда модель не детектирует объекты.

Чувствительность

Решением минуса 4.3.2 является учет False Negatives примеров. Чувствительность (recall) – доля от правильно-предсказанных примеров ко всем истинным. С учетом обозначений,

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

Также, как и рассмотренная метрика 4.3.2, чувствительность варьируется от 0 до 1, высокая чувствительность показывает, что большинство предсказанных ограничивающих рамок покрывают все истинные рамки. Повышение recall полезно, когда необходимо детектирование всех объектов.

Но **минусом** данной метрики можно назвать неучет ложных False Positives срабатываний.

Выводы:

1. Высокая точность, но низкая чувствительность говорит о том, что большая часть истинных рамок была сдетектирована, но при этом было много ложных срабатываний;

2. Высокая чувствительность, но низкая точность указывает нам на то, что предсказания корректны, но часть истинных рамок была пропущена;

3. Увеличить производительность можно, введя порог по коэффициенту уверенности, чтобы не рассматривать неуверенные предсказания.

Соответсвенно, метрики 4.3.2 и 4.3.2 дополняют друг друга. Поэтому, основываясь показаниях двух метрик, можно ввести нечто среднее.

4.3.3 Average Presicion

Average Presicion является композицией уже рассмотренных метрик 4.3.2 и 4.3.2. Было выяснено, ROC кривая менее информативна, чем Precision-Recall кривая для несбалансированных классов [29]. Поэтому Average Presicion состоит из 2 основных стадий: построение Precision-Recall кривой и аппроксимация площади под графиком.

Precision-Recall кривая

Precision-Recall кривая – график в осях y [Точность] – x [Чувствительность] для различных порогов. Является промежуточным этапом для расчета метрики для классификации.

Как только точки кривой получены, следущим этапом идет расчет площади под графиком. Поскольку в алгоритме построения Precision-Recall кривой шла сортировка по возрастанию коэффициента уверенности по всем детекциям, то важно понимать, что примеры бывают «сложные» и уверенность модели может падать на предсказаниях. Таким образом, мы можем получить разные оценки качества при одних и тех же детекциях, но разной уверенности.

Решением этой проблемы служит не точное нахождение площади под графиком, а интерполяция ее, поскольку зачастую Precision-Recall кривая имеет зигзагообразный вид, изображенный на рисунке 10.

Существуют большое колиство видов интерполяций, но самый распространенный в задачах детекции – по 10 точкам.

AP@[.5:.95]

Чтобы абстрагироваться от локализации и классификации, используется обобщенная метрика mAP – среднее Average Presicion при разных порогах IoU: от 0.5 до 0.95 с шагом 0.05. Также существуют важные

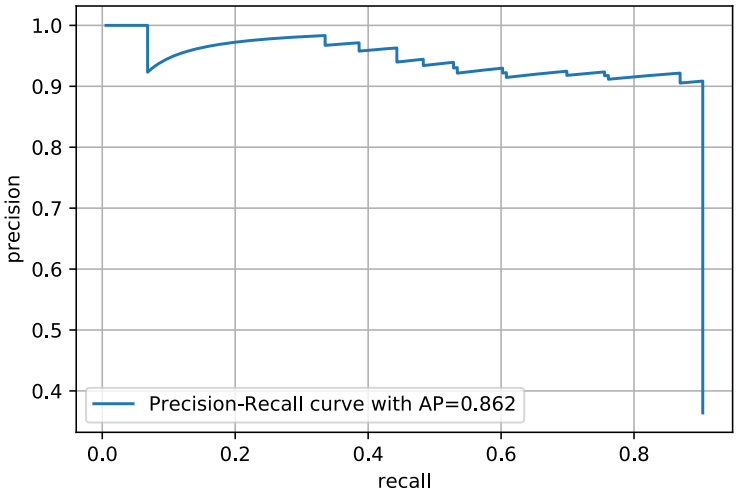


Рис. 10: Типичный вид Precision-Recall кривой.

погоги IoU – 0.5 и 0.75, которые также употребляются в сравнении нейросетевых алгоритмов.

Плюсы: одна цифра для оценки качества задачи детекции. Используется для сравнения всех алгоритмов детекции.

Минусы: сложна для интерпретирования.

5 Эксперименты

Согласно поставленному плану экспериментов 4.1, мною было проведено большое количество исследований с аннотированными данными и выбранными алгоритмами.

5.1 Увеличение выборки реальных данных

Самая первая идея, которая приходит в голову, – увеличение выборки данных путем добавления различных аугментаций:

1. Шум со случайными параметрами (устойчивость модели ко внешним изменениям);
2. Зеркальные отражения;
3. Обучение на синтетической выборке в черно-белом цвете (выделение и локализация объекта).

Но эти простые действия не принесли качественного результата. Ознакомиться с ними можно, используя таблицу 2.

Аугментация	Mavic-2 pro	Phantom	Precision	Recall
Ч/б синтетика	✓		36.26	14.63
Синтетика с шумом	✓		100	18.2
Ч/б реальные		✓	79.8	5.7

Таблица 2: Сравнение аугментаций на различных коллекциях данных.

5.2 Смесь данных без использования модуля NDFT

Далее пришла идея применить смесь данных в качестве базовой идеи Domain Adaptation. В первую очередь, хочется отметить результаты на крайних случаях: обучение либо только на синтетически-созданных (см. Рис. 6(c, d)), либо только на реальных данных (см. Рис. 6(a, b)).

В таблицах 3 и 4 приведены результаты эксперименотов для смеси данных. Наглядное представление метрики mean Average Precision представлено на рисунке 11.

Таким образом, для детекции БПЛА Mavic-2 pro требуется 50 % реальных размеченных данных для получения качественного результата.

Количество данных	Отношение реальных ко всей коллекции, %	mAP	Precision	Recall	AP@.5	AP@.75
687	100	61.73	23.42	98.91	94.9	74.97
987	69.6	62.51	39.69	96.2	94.81	72.65
1087	63.2	62.12	87.13	95.65	94.49	73.18
1187	57.9	63.13	33.65	97.28	93.71	72.93
1374	50	63.21	69.69	96.20	95.38	74.18
1487	46.2	62.75	90.1	94.02	94.31	75.84
1687	40.7	62.12	79.91	92.93	92.31	72.46
2687	25.6	61.59	49.43	94.57	91.17	73.93
5094	13.5	58.89	51.17	95.11	92.52	67.90
4407	0	0.27	100	0.54	0.54	0

Таблица 3: Сравнение метрик на БПЛА Mavic-2 pro.

Количество данных	Отношение реальных ко всей коллекции, %	mAP	Precision	Recall	AP@.5	AP@.75
1453	100	72.99	50	100	99.72	92.55
1753	82.8	74.41	50	100	99.77	92.29
1853	78.4	73.57	50	100	98.75	92.19
1953	74.4	74.34	49.86	99.43	99.12	93.07
2253	64.5	73.48	47.83	100	98.90	92.79
2453	59.2	74.02	47.70	100	99.65	94.16
2953	49.2	73.07	49.58	99.43	99.39	92.41
3453	42.1	71.08	48.75	100	98.06	92.23
4453	32.6	71.02	50.72	100	100	92.89
3304	0	40.92	92.62	78.41	77.05	39.99

Таблица 4: Сравнение метрик на БПЛА Phantom.

5.3 Смесь данных с использованием модуля NDFT

Можно ли добиться результата выше, чем 63 mean Average Precision? Оказывается, возможно. Мною были проведены 2 эксперимента, результаты которых можно найти в таблице 5.

Таким образом, с помощью модуля NDFT увеличилось качество mean Average Precision на 17 у.е. С одной стороны, с помощью синтетически-созданный выборки мы улучшаем качество регрессии ограничивающих рамок, а с помощью ручной разметки реальных – увеличиваем устойчивость модели. С изменениями метрик во времени обучения модуля NDFT

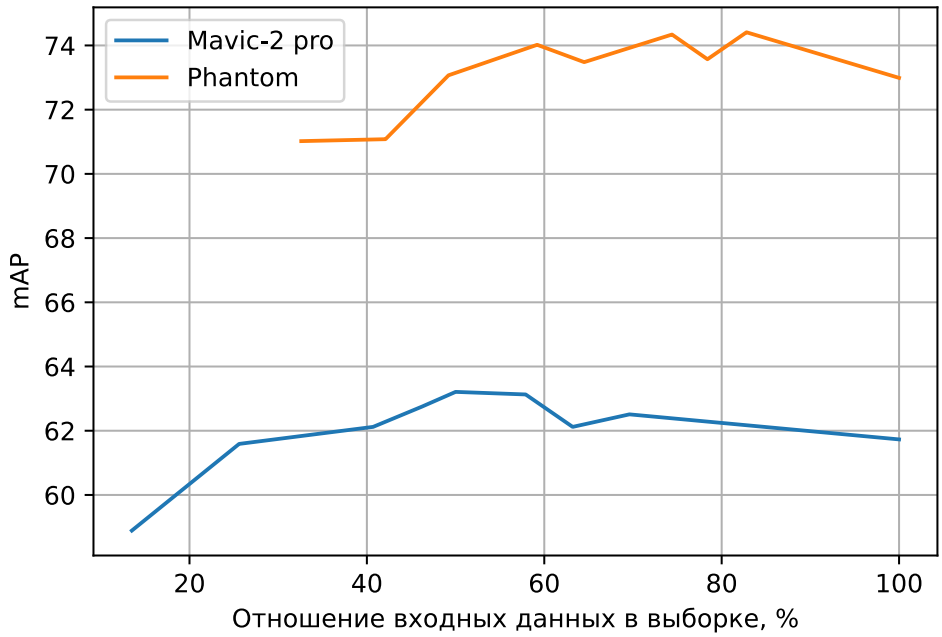


Рис. 11: Сравнение качеств детекции на разных БПЛА.

можно ознакомится на Рис. 12. Состязательный параметр из 4.2.1 был выставлен в $\gamma = 0.101$.

Приведем основные характеристики модели с модулем NDFT:

1. Нейронная сеть извлечения признаков – ResNet 101;
2. Вес модели – 557 Мегабайт;
3. Скорость обработки изображения – от 10 до 14 FPS;
4. Код написан на языке Python с использованием библиотеки PyTorch v0.4.0;
5. Обучение модели составило 26 часов при сбросе весов у классификатора через каждые 250 батчей;
6. Использовалась видеокарта GeForce RTX 2080 Ti.
7. Начальный learning rate был выставлен в 0.005, далее через каждые 4 эпохи делился пополам;
8. В качестве оптимизатора был выбран SGD.
9. Размер батча, в силу ограничения памяти видеокарты, был равен 4.

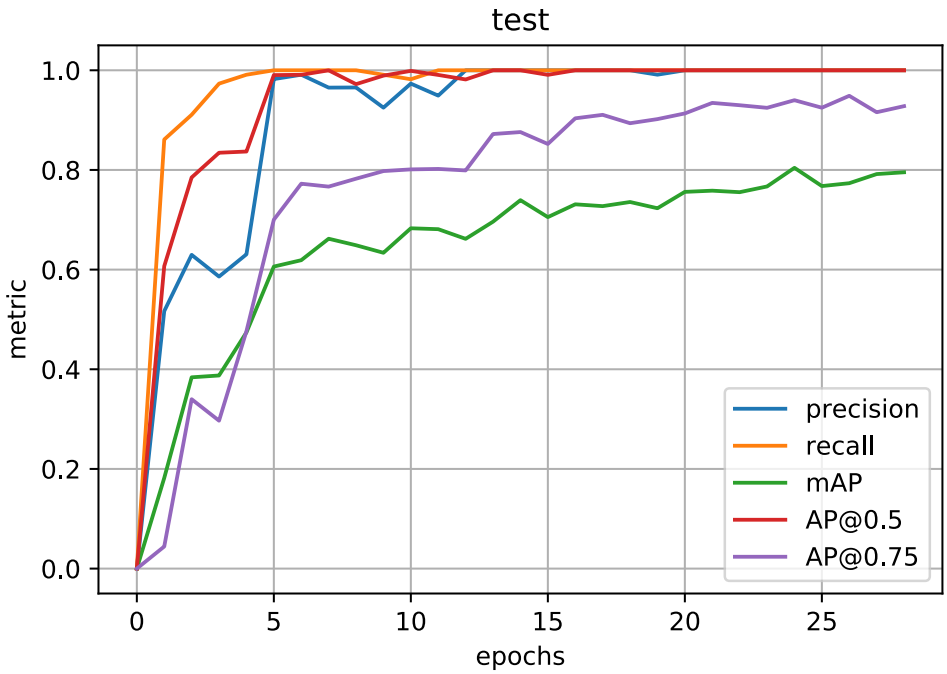


Рис. 12: Изменение метрик на различных эпохах при обучении модуля NDFT.

Количество данных	Количество реальных данных, %	mAP	Precision	Recall	AP@.5	AP@.75	NDFT
2200	3.36	24.34	84.13	94.64	66.28	11.7	✓
1374	50	63.21	69.69	96.20	95.38	74.18	
3000	30	**80.42**	**100**	**100**	**100**	**93.99**	✓

Таблица 5: Сравнение метрик с использованием модуля NDFT и без на БПЛА Mavic-2 pro.

Важно отметить, что для достижения подобных результатов синтетическая коллекция данных должна быть визуальна похожа на реальную.

6 Выводы

Прежде всего стоит отметить, что работа направлена на прикладную применимость исследованного модуля NDFT. По результатам экспериментов можно уверенно сказать, что 50 % выборки реальных данных можно заменить синтетической. В данной работе не проводились эксперименты над варицией реальных данных для модуля NDFT, поскольку классификатор с Cross-entropy loss эффективно распознает данные при сбалансированной выборке. Хочется отметить, что эта проблема также решаема с помощью идей, взятых, например, из статьи [30], но это уже тема для отдельного исследования.

Данную модель можно вывести в режим работы реального времени, если заменить блок извлечения признаков на более легкий. Это необходимо для автоматического принятия решений БПЛА.

Поставленная задача успешно выполнена. Думаю, если развивать идеи области машинного обучения Domain Adaptation, то в ближайшем будущем можно тратить гораздо меньше сил на ручную разметку, а также даже по нескольким образом успешно детектировать объекты даже в самых сложных ситуациях.

7 Список используемой литературы

[1] Lin Tsung-Yi, Maire Michael, Belongie Serge, Bourdev Lubomir, Girshick Ross, Hays James, Perona Pietro, Ramanan Deva, Zitnick Lawrence, and Dolla'r Piotr. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2015.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[4] *ImageNet Large Scale Visual Recognition Challenge*. http://www.image-net.org/challenges/LSVRC/.

[5] John Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE, 1986.

[6] Redmon Joseph, Divvala Santosh, Girshick Ross, and Farhadi Ali. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[7] Redmon Joseph and Farhadi Ali. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[8] Redmon Joseph and Farhadi Ali. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[9] Bochkovskiy Alexey, Wang Chien-Yao, and Liao Hong-Yuan Mark. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[10] Liu Wei, Anguelov Dragomir, Erhan Dumitru, Szegedy Christian, Reed Scott, Fu Cheng-Yang, and C. Berg Alexander. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325v5*, 2016.

[11] Girshick Ross, Donahue Jeff, Darrell Trevor, and Malik Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2014.

[12] Girshick Ross. Fast-rcnn. *arXiv preprint arXiv:1504.08083*, 2015.

[13] Ren Shaoqing, He Kaiming, Girshick Ross, and Sun Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2016.

[14] Cai Zhaowei and Vasconcelos Nuno. Cascade r-cnn: Delving into high quality object detection. *arXiv preprint arXiv:1712.00726*, 2017.

[15] Bodla Navaneeth, Singh Bharat, Chellappa Rama, and S. Davis Larry. Improving object detection with one line of code. *arXiv preprint arXiv:1704.04503v2*, 2017.

[16] Jiang Borui, Luo Ruixuan, Mao Jiayuan, Xiao Tete, and Jiang Yuning. Acquisition of localization confidence for accurate object detection. *arXiv preprint arXiv:1807.11590v1*, 2018.

[17] Lin Tsung-Yi, Dollar Piotr, Girshick Ross, He Kaiming, Hariharan Bharath, and Belongie Serge. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144v2*, 2017.

[18] Wu Yue, Chen Yinpeng, Yuan Lu, Liu Zicheng, Wang Lijuan, Li Hongzhi, and Fu Yun. Rethinking classification and localization for object detection. *arXiv preprint arXiv:1904.06493*, 2020.

[19] Song Guanglu, Liu Yu, and Wang Xiaogang. Revisiting the sibling head in object detector. *arXiv preprint arXiv:2003.07540v1*, 2020.

[20] Zhenyu Wu, Karthik Suresh, Priya Narayanan, Hongyu Xu, Heesung Kwon, and Zhangyang Wang. Delving into robust object detection from unmanned aerial vehicles: A deep nuisance disentanglement approach. *arXiv preprint arXiv:1908.03856*, 2019.

[21] Chen Yuhua, Li Wen, Sakaridis Christos, Dai Dengxin, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. *arXiv preprint arXiv:1803.03243*, 2018.

[22] Krizhevsky Alex, Sutskever Ilya, and E. Hinton Geoffrey. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[23] D Zeiler Matthew and Fergus Rob. Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*, 2013.

[24] Simonyan Karen and Zisserman Andrew. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[25] Szegedy Christian, Liu Wei, Jia Yangqing, Sermanet Pierre, Reed Scott, Anguelov Dragomir, Erhan Dumitru, Vanhoucke Vincent, and Rabinovich Andrew. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[28] Jaccard P. *Distribution de la flore alpine dans le Bassin des Dranses et dans quelques regions voisines*, volume 37. Bull. Soc. Vaudoise sci. Natur, 1901.

[29] Saito Takaya and Rehmsmeier Marc. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 2015.

[30] Lin Tsung-Yi, Goyal Priya, Girshick Ross, He Kaiming, and Dollár Piotr. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.