|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и управление»

КАФЕДРА «Системы автоматического управления и электротехника» (ИУ3-КФ)

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Разработка модуля трекинга объектов видеопотока для мобильной автономной платформы***

Студент \_\_УТС.Б-81\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_** Лохмачев Н. В.**\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Корлякова М. О.**\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

**АННОТАЦИЯ**

Расчетно-пояснительная записка 67 с., 41 рис., 13 табл., 11 источников, 7 прил.

Объектом разработки является программное обеспечение, автоматизирующее процесс формирования набора данных, дообучения модели и детектирования.

Цель работы — исследование и реализация алгоритмов компьютерного зрения для формирования управления автономной мобильной платформой

Глава 1 содержит анализ характеристик автономной платформы и методов детектирования и выбор наилучшего из подходов для решения поставленной задачи.

В главе 2 представлены эксперименты, полученные в процессе выполнения задачи.

Глава 3 представляет собой анализ работы с точки зрения охраны труда и экономической целесообразности.

Приложения содержат программные коды для сравнения сред MatLab и Python, формирования набора данных и детектирования.

Поставленная цель достигается за счет применения современных подходов машинного обучения для решения задачи детектирования и обеспечения связи между микроконтроллерами мобильного робота и компьютером с мощной аппаратной составляющей с целью реализации данного процесса в реальном времени.

**Содержание**

[ВВЕДЕНИЕ 5](#_Toc42965685)

[1. Научно-исследовательская часть 6](#_Toc42965686)

[1.1. Постановка задачи 6](#_Toc42965687)

[1.2. Обзор методов детектирования 7](#_Toc42965688)

[1.3. Конфигурация модели – нейронная сеть 8](#_Toc42965689)

[1.4. Сверточные нейронные сети 12](#_Toc42965690)

[1.5. Принцип работы алгоритмов детектирования 14](#_Toc42965691)

[1.6. Концепция переноса обучения 15](#_Toc42965692)

[1.7. Возможность реализации модели на автономной платформе 16](#_Toc42965693)

[1.8. Сравнение моделей детектирования объектов 17](#_Toc42965694)

[1.9. Архитектура и принцип работы MobileNet SSD 18](#_Toc42965695)

[1.10. Выбор алгоритма трекинга 21](#_Toc42965696)

[1.11. Сравнение характеристик Raspberry Pi 3 и современного ноутбука 23](#_Toc42965697)

[1.12. Вывод 25](#_Toc42965698)

[2. Проектно-конструкторская часть 26](#_Toc42965699)

[2.1. Составляющие автономной мобильной платформы 26](#_Toc42965700)

[2.2. Сравнение программных сред MatLab и Python 27](#_Toc42965704)

[2.3. Выбор программной составляющей для решения задачи 34](#_Toc42965705)

[2.4. Тестирование Object Detection API 35](#_Toc42965706)

[2.5. Написание программы формирования набора данных для тонкой настройки модели MobileNet SSD 36](#_Toc42965707)

[2.6. Автоматизация процесса тонкой настройки модели детектирования 39](#_Toc42965708)

[2.7. Подбор гиперпараметров модели 45](#_Toc42965709)

[2.8. Разработка интерфейса 48](#_Toc42965710)

[2.9. Объединение IP-камеры и вычислительной системы общим сервером 49](#_Toc42965711)

[2.10. Тестирование программно-аппаратной системы 51](#_Toc42965712)

[3. Технологическая часть 53](#_Toc42965713)

[3.1. Экономическое обоснование дипломного проекта 53](#_Toc42965714)

[3.2. Организация и планирование работ 53](#_Toc42965715)

[3.3. Определение стоимости специального оборудования 55](#_Toc42965716)

[3.4. Расчет основной заработной платы 55](#_Toc42965717)

[3.5. Оценка экономической целесообразности разработки 57](#_Toc42965718)

[3.6. Характеристика параметров рабочего места 58](#_Toc42965719)

[3.7. Утилизация люминесцентных ламп 64](#_Toc42965720)

[ЗАКЛЮЧЕНИЕ 66](#_Toc42965721)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 67](#_Toc42965722)

[ПРИЛОЖЕНИЕ А Установка NVIDIA CUDA для ускорения работы модели детектирования 69](#_Toc42965723)

[ПРИЛОЖЕНИЕ Б Настройка Object Detection API 72](#_Toc42965724)

[ПРИЛОЖЕНИЕ В Код Python для формирования набора данных 73](#_Toc42965725)

[ПРИЛОЖЕНИЕ Г Код Python для детектирования 76](#_Toc42965726)

[ПРИЛОЖЕНИЕ Д Код Python для подбора гиперпараметров при помощи генетического алгоритма 78](#_Toc42965727)

[ПРИЛОЖЕНИЕ Е Код Python для обучения нейронной сети на наборе данных Fashion MNIST и её тестирования 81](#_Toc42965728)

[ПРИЛОЖЕНИЕ Ж Код MatLab готовой обученной модели классификтора Fashion MNIST, сгенерированный MatLab 83](#_Toc42965729)

# ВВЕДЕНИЕ

Развитие компьютерных технологий открывает всё больше возможностей для создание интеллектуальных и эффективных систем компьютерного зрения. Они направлены на решение проблем, связанных с обработкой цифровых изображений и распознаванием объектов. В свою очередь, данные алгоритмы направлены на решение задач автопилотирования, распознавания рукописных символов, слежения за объектами, детектирования лиц, проверки на подлинность и других. Однако, универсальные детекторы, хотя и настроены на распознавание до 1000 классов одновременно, не всегда решают локальные задачи, которые стоят перед мобильными автономными системами.

Например, модули охранной системы должны опознавать определенные объекты конкретной среды (обитателей квартиры (людей и животных) или работников офиса и передвигаемые объекты (стулья, другие подвижные модули и т. п.)). С другой стороны универсальные объектные списки содержит возможность распознавать объекты, которые никогда не появятся в поле зрения этой системы. Таким образом, необходимо обеспечить настройку модуля распознавания на уникальные объекты конкретной среды в условиях дефицита ресурсов с точки зрения предоставляемых примеров, вычислительных сред и времени.

Задача является достаточно актуальной, поскольку модели детектирования используются как в системах автоматического пилотирования машин и дронов, так и в медицине для выявления различных заболеваний на снимках. Например, компания eSmart Systems использует дронов для детектирования поломок на линиях электропередач, а модель Inception V3 от Google помогает ученым искать раковые клетки в теле человека.

Основной целью курсовой работы является анализ существующих подходов к реализации алгоритма детектирования и его тонкая настройка под детектирование произвольных объектов. Объект исследования – методы детектирования в различных реализациях, а также подбор гиперпараметров.

# Научно-исследовательская часть

## Постановка задачи

Мобильная платформа вместе со средой, в которой она функционирует, накладывает множество ограничений, под которые необходимо либо настраивать саму систему, либо искать обходные пути борьбы с ними. Основными ограничениями являются:

* маломощные вычислительные ресурсы;
* задержки при посылке сигналов с микрокомпьютера Raspberry Pi на микроконтроллер AtMega;
* перегрев процессора Raspberry Pi;
* IP-камера низкого разрешения, имеющая шум;
* достаточно разнообразная среда;
* малая оперативная память Raspberry Pi.

Несмотря на данные ограничения, в рамках данного проекта полученной системе необходимо выполнять следующие задачи:

* детектирование в реальном времени цели, к которой необходимо двигаться роботу;
* автоматизация процесса тонкой настройки модели детектирования под произвольный объект;
* обеспечение удобства работы с системой пользователю.

В конечном итоге необходимо иметь разработанный алгоритм детектирования, который в дальнейшем можно будет использовать для управления мобильной автономной платформой, включающий в себя:

* функцию детектирования объекта;
* функцию формирования набора данных;
* функцию дообучения модели на произвольных объектах.

## Обзор методов детектирования

Задача детектирования заключается в том, что по входному изображению необходимо предсказать местоположение определенных классов объектов на изображении. Обычно местоположение представляет собой рамку, в которую заключается объект определенного класса.

Изначально задача детектирования решалась методом сопоставления шаблону. Алгоритм запоминает некоторый шаблон, который ему необходимо найти на изображении. К этой задаче есть несколько подходов:

1. Прямое сопоставление шаблону

Если изображение содержит множество идентичных деталей, то можно создать один или несколько шаблонов, которые будут сопоставляться областям изображения в разных разрешениях. Очевидно, данный подход, несмотря на свою простоту, является вычислительно сложным. Помимо этого, разные изображения могут иметь разные характеристики (яркость, контраст, зашумленность), что сделает данный метод практически неработоспособным.

2. Подход, основанный на движении объекта

В тех случаях, когда шаблон может не обеспечивать прямое соответствие, можно использовать шаблоны, которые детализируют объект в ряде различных условий, таких как изменяющиеся перспективы, подсветка, цветовые контрасты или нужные позиции изображения.

3. Подход, основанный на признаках объекта

Данный подход заключается в том, что из изображения шаблона извлекаются некоторые признаки, такие как текстуры, формы, цвета и так далее. Ранее данная операция проделывалась вручную, что создавало множество проблем, поскольку какие-либо изменения в свойствах объекта (контрастность, яркость, зашумленность) приводило к тому, что признаки необходимо было извлекать иначе, то есть применяя другие фильтры и методы.

Однако, на сегодняшний день этим занимаются сверточные нейронные сети, которые способны автоматически извлекать необходимые признаки из изображения и создавать некий универсальный шаблон. Данный подход является современным и наиболее развитым, поскольку он обеспечивает независимость объекта от его поворота, масштаба и цвета.

## Конфигурация модели – нейронная сеть

Изначально нейронные сети были вдохновлены принципом работы мозга живого существа, состоящим в простейшей форме из нейронов и синапсов. Простейшие полносвязные нейронные сети, как правило, состоят из входного слоя, скрытого слоя (одного или нескольких) и выходного слоя, рисунок 1 [1].

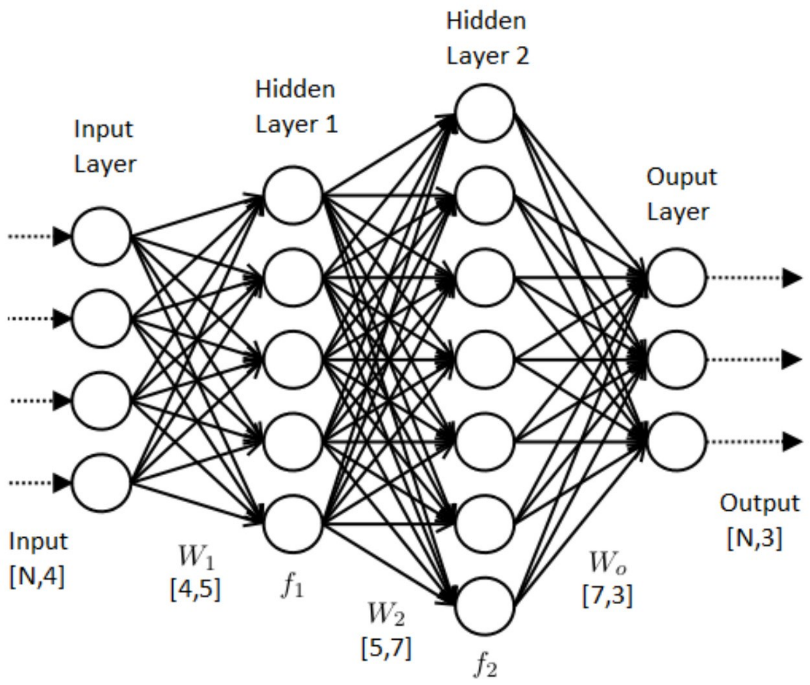


Рисунок 1 – Вид полносвязной нейронной сети

Каждый узел и каждая связь данной нейронной сети имеют свои значение и вес соответственно. Когда входной сигнал приходит в нейронную сеть, он «активирует» определенный набор нейронов в скрытом слое, который, в свою очередь, активирует нейроны следующего слоя и так далее. В конечном итоге, выдает выходной сигнал, который может быть чем угодно (число, набор чисел, целое изображение и так далее).

Для того, чтобы нейронная сеть выдавала верные значения, необходимо обучить её. Процесс обучения состоит в том, чтобы изменить веса нейронной сети так, чтобы в каждом её слое активировались нужные нейроны. Обучение нейронной сети может происходить с учителем и без учителя.

Если нейронная сеть обучается на основе эталонных выходных сигналов, то обучение происходит с учителем. Соотвественно, если выходные сигналы не даны, обучение происходит без учителя.

Обучение с учителем и без учителя – две разные задачи. В данной работе все алгоритмы основаны на парадигме обучения с учителем.

Обучение в нейронной сети происходит в два этапа:

1. Прямой проход

При прямом проходе для нейрона каждого слоя подсчитывается взвешенная сумма плюс смещение между значениями сигнала, приходящего на вход сети, и соответствующими им весами связей.

Этот процесс для каждого нейрона выражается формулой

,

,

где

 – индекс слоя,

– индексы нейронов  и  слоя,

 – количество нейронов в слое,

 – взвешенная сумма,

 – значение нейрона,

 – вес связи,

 – смещение.

2. Обратный проход

Выходной сигнал данной сети сравнивается с эталонным и веса корректируются в соответствии с ошибкой между фактическим и эталонным выходными сигналами. Например, на основе градиентного спуска.

,

,

где

 – функция стоимости (потерь),

 – количество слоёв сети [2]

При обучении нейронной сети разработчик может столкнуться со множеством проблем.

В нейронных сетях происходит недообучение, если количество итераций оптимизации нейронной сети слишком мало, выбранная архитектура неудачна или слишком мал набор данных.

Помимо этого, нейронная сеть может переобучиться, что означает, что она «привыкла» к набору данных, на котором училась, из-за чего точность сильно падает. Для того, чтобы избежать этого, используют различные методы регуляризации нейронной сети. Методов регуляризации множество и для получения наилучшего результата необходимо экспериментировать.

Одним из методов регуляризации является добавление штрафа в функцию потерь:



где - настраиваемый параметр регуляризации,

L1 – L1-регуляризация,

L2 – L2-регуляризация.

Помимо этого, есть метод снижения сложности нейронной сети (Dropout), который с определенной вероятностью на определенной итерации отключает связи некоторых нейронов.

Также возможна аугментация данных (случайные повороты, масштабирования, изменения цветов и другие техники) для снижения переобучения. Это происходит, поскольку набор данных увеличивается.

Метод раннего останова нейронной сети – метод, при котором происходит остановка обучения пользователем. Для этого строится график ошибки предсказания на валидационном наборе данных и сеть останавливается, если ошибка начинает увеличиваться.

Ещё одним методом, который позволяет повысить качество нейронной сети, называется пакетная нормализация. При разбиении набора данных на пакеты (маленькие наборы примеров) можно на каждом их слое производить нормализацию следующим образом:



где  и  - настраиваемые моделью параметры.

В итоге, пакеты имеют фиксированные математическое ожидание и дисперсию.

Другой проблемой глубоких нейронных сетей является затухание или взрыв градиентов (градиент очень мал или очень велик). Это происходит из-за того, что веса со значением больше 1 после прохождения большого количества слоёв перемножаются и увеличиваются экспоненциально, в то время как веса со значениями от 0 до 1 экспоненциально уменьшаются.

Частичным решением этой проблемы является правильная инициализация весов перед обучением. Для каждого слоя l для матрицы весов справедлива формула:



Здесь, используя функцию активации ReLU, будет достигаться дисперсия, равная  для нейрона . Чем больше n, тем меньше .

## Сверточные нейронные сети

Сверточные нейронные сети показали куда большую эффективность при работе с изображениями, чем обычные полносвязные сети. Помимо этого, они более интуитивны, поскольку в любой сверточной нейронной сети можно увидеть, что конкретно выделяется в виде признаков.

Простая сверточная нейронная сеть представлена на рисунке 2.

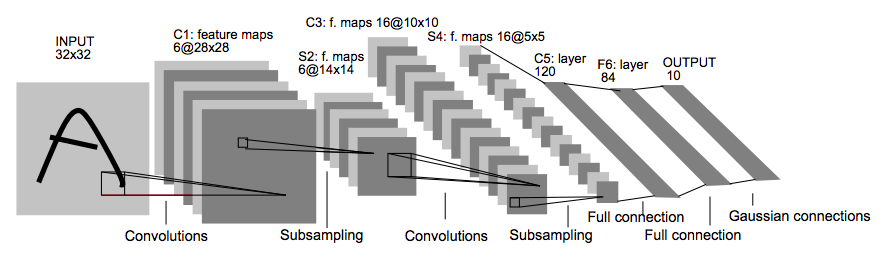


Рисунок 2 – Архитектура LeNet5

Каждая сверточная нейронная сеть обычно состоит из слоев свёртки, пулинга, регуляризаций и полносвязных слоёв. Рассмотрим каждый из них подробнее.

Операция свертки нужна для выделения каких-либо признаков изображения. По изображению (матрице) проезжает скользящее окно (фильтр) определенного размера, производя при этом обычное скалярное произведение элементов внутри окна. Полученные числа записываются в новую матрицу, рисунок 3.

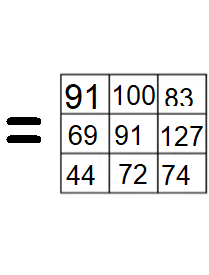
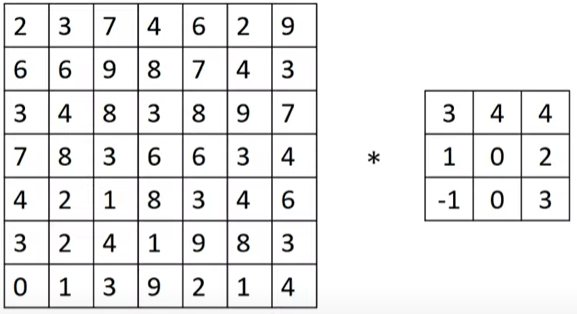


Рисунок 3 – Операция свертки с шагом 2

Размер новой матрицы зависит от размера фильтра, добавления рамки вокруг исходной матрицы и шага. Он высчитывается по следующей формуле:



где

 – размер исходного изображения,

– размер добавленной рамки,

– размер фильтра,

– шаг фильтра.

При обучении любой сверточной сети происходит обучение именно фильтров. Помимо этого, фильтры могут быть многомерными. Многомерный фильтр можно рассматривать, как несколько одномерных фильтров, параллельных друг другу. Многомерные фильтры на выходе дают многомерные признаки.

Чем глубже сверточная нейронная сеть, тем выше уровень извлечения признаков. Например, на первом слое извлекаются определенные шаблоны пикселей, далее на следующем слое они комбинируются в более сложные формы, рисунок 4.

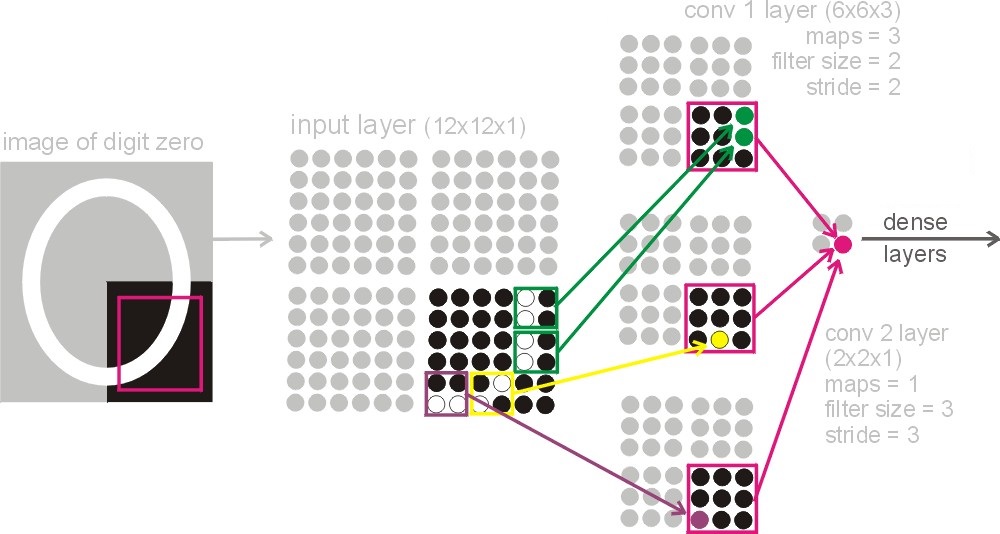


Рисунок 4 – Пример работы сверточной нейронной сети

После сверточных слоёв, как правило, идут слои пулинга, которые по сути находят максимальное (или среднее) число в рамке и таким образом выделяют наиболее влиятельный признак. В конце нейронной сети обычно находится два полносвязных слоя, которые уже на основе извлеченных признаков производят классификацию.

Кроме того, после слоёв свертки могут стоять слои регуляризации.

Одним из современных методов регуляризации является добавление «скипов» - пропусков нескольких слоёв. Такие сети называется ResNet-подобными. Их достаточно легко оптимизировать. Простые сети при увеличении глубины показывают большую ошибку обучения и теряют в точности. ResNet исправляют данную проблему и позволяют увеличивать точность с увеличением глубины.

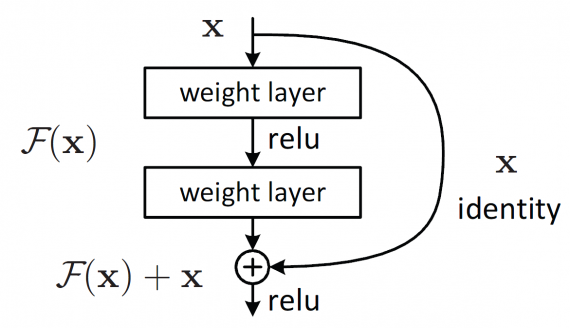


Рисунок 5 – Слой со «скипом» ResNet

## Принцип работы алгоритмов детектирования

Каждый алгоритм детектирования имеет свою специфику и свою архитектуру. Несмотря на это, у всех них есть общие черты.

Простейшим алгоритмом детектирования является детектирование скользящим окном. Его суть заключается в том, что есть несколько рамок разного размера, которые проходят по всему изображению и проводят классификацию для каждой из области, выделенной рамкой. Однако, это не самый эффективный способ детектирования, поскольку он очень вычислительно ёмкий и требует проводить большое количество классификаций. Чем больше изображение, тем дольше будет производиться детекция. Однако, есть возможность сильно его упростить, если рассчитывать значения для каждого положения рамки при помощи сверточных нейронных сетей.

Например, необходимо распознать один из 4-х классов на изображении 28х28 пикселей. Вместо того, чтобы циклически проходить рамкой по изображению, можно произвести классификацию для каждого положения рамки за один проход, основываясь на принципе скользящего окна в сверточной нейронной сети, рисунок 6 [3].

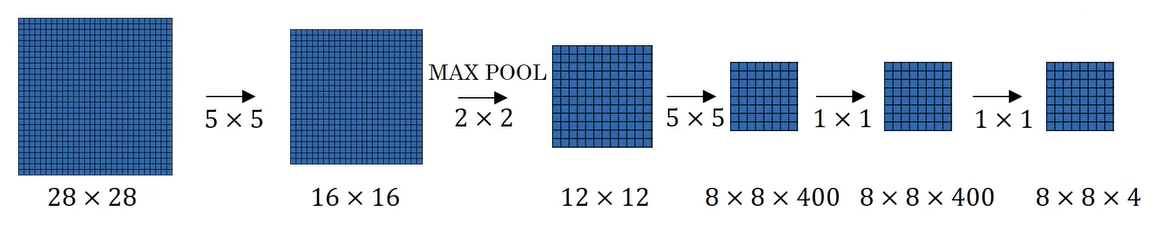


Рисунок 6 – Сверточная нейронная сеть, предсказывающая вероятности для каждого из 64 положений рамки

Выходом сети будет являться матрица 8х8х4, где каждый элемент 1х1х4 представляет собой вектор вероятностей принадлоежности области в одной из рамок классу.

Алгоритмы детектирования в реальном времени, однако, делятся на задачи регрессии и классификации. Один из таких алгоритмов (MobileNet SSD) будет рассмотрен далее.

## Концепция переноса обучения

Очевидно, что обучать свою нейронную сеть с нуля практически невозможно при данных ограничениях. Даже имея компьютер с высокой вычислительной мощностью это может занимать несколько дней. Следовательно, необходимо воспользоваться уже имеющимися технологиями и обученными моделями с целью применения их в своих условиях и ограничениях. Это и есть парадигма переноса обучения.

Перенос обучения (transfer learning) – это процесс адаптации заранее обученной модели под другую задачу.

Перенос обучения может иметь множество форм и подходов. Например, можно «отсоединить» последний слой обученного классификатора. Тогда вместо меток он будет выдавать извлеченные признаки входного сигнала, которые в дальнейшем можно использовать в других целях, рисунок 7 [4].

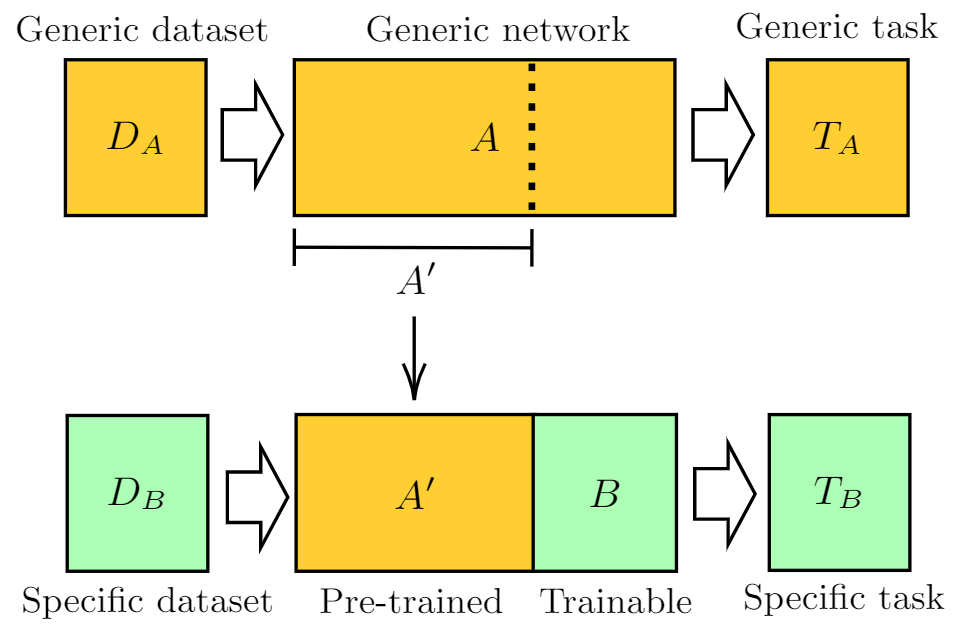


Рисунок 7 – Демонстрация концепции переноса обучения

Тонкая настройка (Fine tuning) – это один из подходов переноса обучения, в котором некоторым слоям запрещается изменяться. Он имеет смысл, поскольку, например, сверточные нейронные сети, как правило, на первых слоях имеют практически одинаковые веса, поскольку данные слои представляют собой экстракторы признаков низкого уровня (края объектов, простые формы). Следовательно, нет необходимости обучать нейронную сеть с нуля. Достаточно взять готовую модель, «заморозить» несколько слоёв (то есть не давать им изменять свои веса) и обучать оставшиеся последние слои.

## Возможность реализации модели на автономной платформе

Было поставлен эксперимент реализации более легкой модели детектирования на Raspberry Pi 3 автономной платформы. Однако, скорость обработки изображения при таком подходе – 1.5 кадра в секунду. При этом, задачи построения карты глубины и обучения с подкреплением для объезда препятствий выполнить при таких мощностях не предоставляется возможным.

К тому же, температура микрокомпьютера начинает сильно повышаться, что требует дополнительных затрат на кулер или на миниатюрный ускоритель вычислений Movidius Neural Compute Stick [5], что не является оптимальным решением с точки зрения затрат.

Как было описано выше, все вычисления и обработка изображения будут ложиться на мощный компьютер, что позволит решать задачи построения карты глубины, распознавания и детектирования, а также обучения с подкреплением для дальнейшего развития проекта.

С этой целью будет написана программа для трансляции изображения с камеры Raspberry на ноутбук MSI в реальном времени. Следовательно, можно гарантировать, что при данном подходе все вышеперечисленные задачи выполнимы.

## Сравнение моделей детектирования объектов

Одной из наиболее важных составляющих данной работы является выбор оптимальной модели детектирования. Для этого необходимо сравнить существующие алгоритмы по скорости и качеству детектирования. В процессе выполнения данной работы сравнивались следующие модели:

1. Алгоритм You Only Look Once на основе библиотеки Keras [6]
2. MobileNet Single Shot Detector на основе библиотеки Keras
3. MobileNet Single Shot Detector на основе библиотеки OpenCV dnn
4. MobileNet Single Shot Detector на основе Object Detection API [7]

Скорость моделей оценивалась по количеству кадров в секунду, а качество – по оценке mAP. Помимо этого, проверялась возможность дообучения модели, таблица 1.

Таблица 1 – Сравнение реализаций моделей по качеству и скорости

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Количество кадров в секунду** | **Оценка качества mAP** | **Возможность тонкой настройки** |
| **YOLO (Keras)** | 11 | 48 | Да |
| **MobileNetSSD (Keras)** | 11 | 22 | Да |
| **MobileNetSSD (OpenCV dnn)** | 25 | 22 | Нет |
| **MobileNetSSD (Object Detection API)** | 17 | 22 | Да |

Поскольку алгоритм YOLO показал достаточно низкую производительность (10-12 кадров в секунду) даже при вычислении на GPU, было принято решение отказаться от его использования. Выбор был сделан в пользу MobileNet SSD, так как он демонстрирует хорошую производительность как на GPU, так и на CPU (около 20 кадров в секунду). После тестирования данного алгоритма в разных реализациях, лучшего всего с точки зрения простоты дообучения и скорости он показал себя на основе Object Detection API от TensorFlow.

## Архитектура и принцип работы MobileNet SSD

Рассмотрим принцип работы модели Single Shot Detector. На рисунке 4 представлена её архитектура.

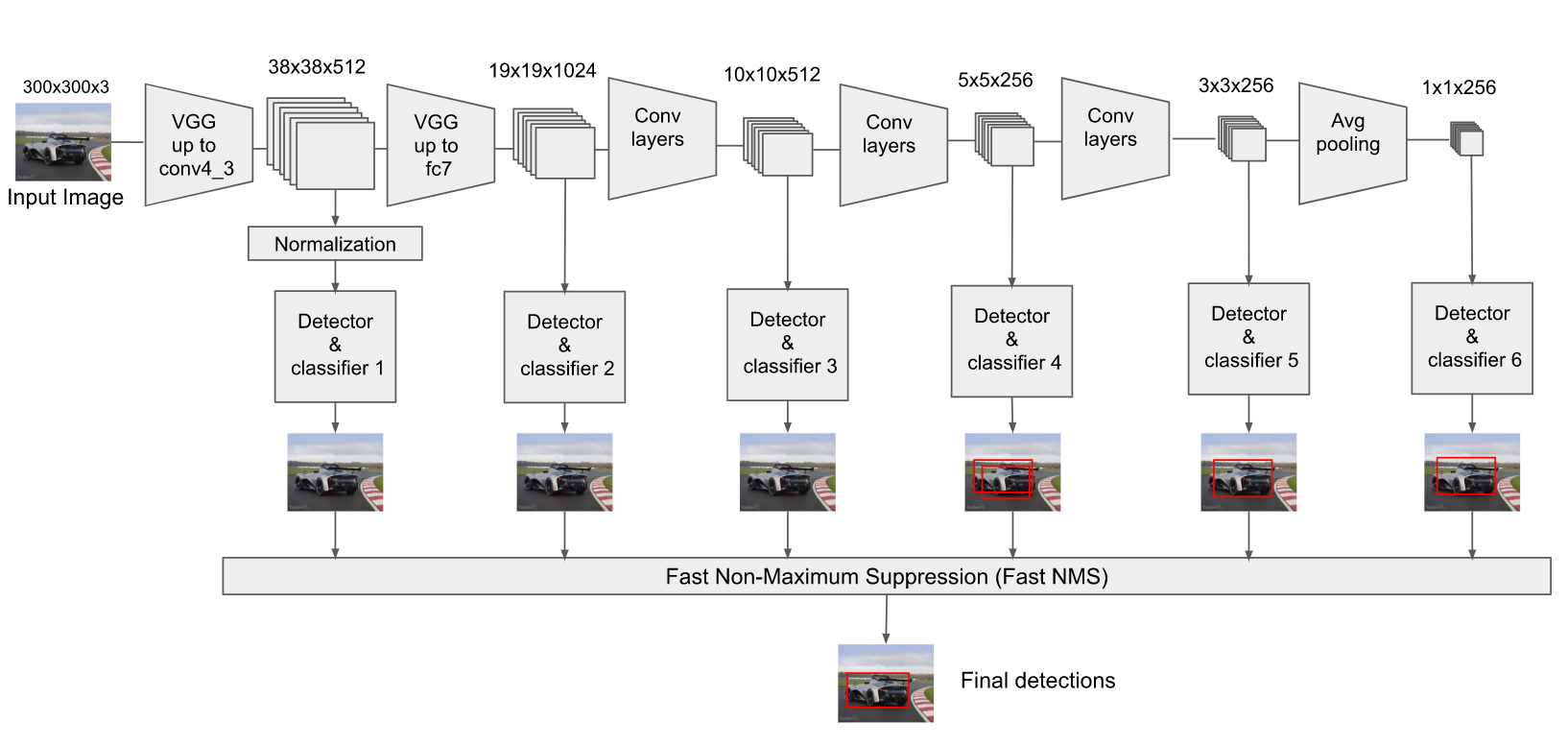


Рисунок 8 – Архитектура SSD

Как можно видеть на рисунке 8, на вход модели подается изображение, проходящее через несколько сверточных слоёв. Причем часть этих слоёв взята из модели VGG16. Каждому сверточному слою ставятся в соответствие блоки из детектора и классификатора, на выходе каждого из которых получается рамка. Это необходимо, что детектировать объекты разного размера. Чем меньше пространственная размерность (ближе к выходу), тем больше вероятность детектирования больших объектов. Далее применяется метод быстрого подавления рамок, вероятность которых является наименьшей.

Теперь рассмотрим блок детектор-классификатор 4 для понимания принципа его работы.

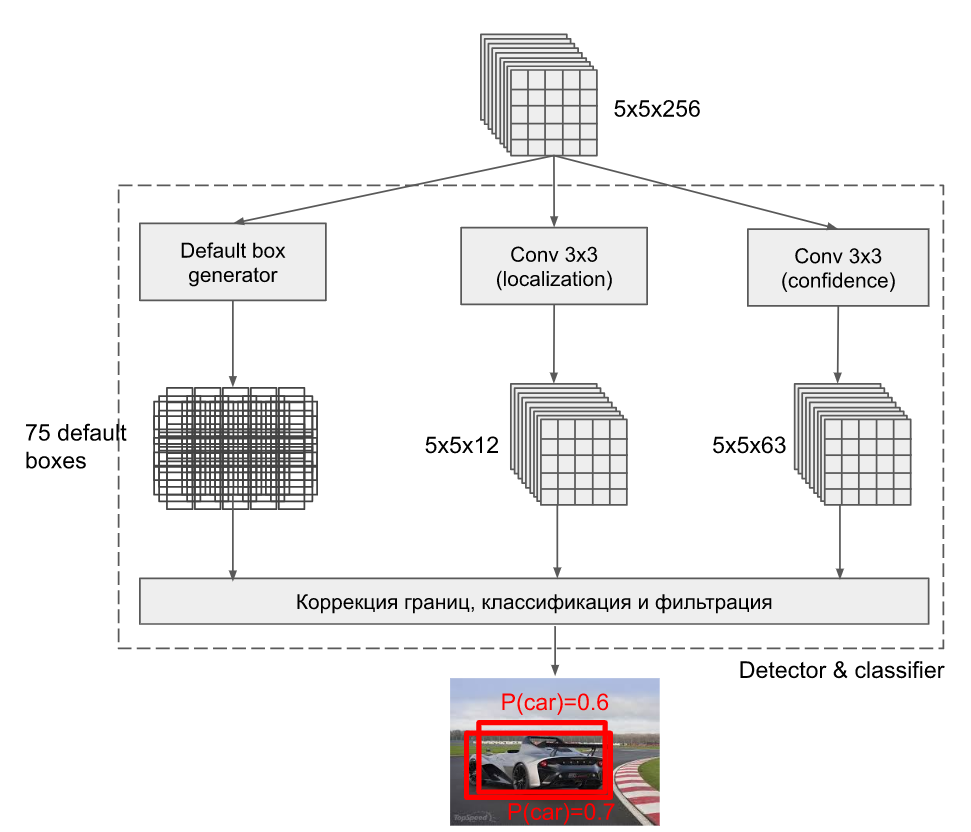


Рисунок 9 – Вид блока Detector & classifier 4

В блоке на рисунке 9 выполняются 3 операции:

* + - 1. Изображение размечается точками (центрами рамок) и производится генерация нескольких рамок для каждого из центров
      2. При локализации рамки генерируется вектор коррекции положения рамки, состоящий из изменения центра, длины и ширины относительно её первоначального состояния
      3. Формируется вектор «уверенности» по каждому из классов, к которому применяется softmax-функция, возвращающая вероятности классов

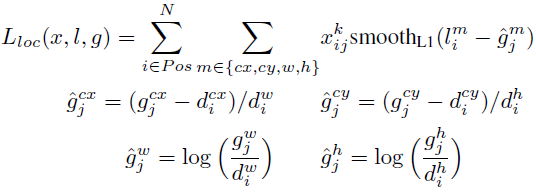
В конечном итоге, появляется множество рамок, фильтруемых по вероятности принадлежности к классу.

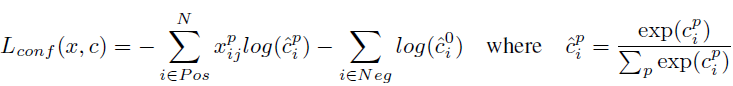
Основное отличие MobileNet SSD от SSD – это наличие более эффективных с точки зрения вычислений нейронных сетей.

Как и во всех нейронных сетях, в SSD присутствует функция потерь, при помощи которой и оптимизируется сеть.



Она состоит из двух слагаемых:





Здесь:

 - функция потерь локализации, которая представляет собой сглаженную L1-функцию потерь между параметрами предсказанной рамки l и фактической рамки g, то есть сдвиги точки центра и ширина и длина.

 - функция потерь уверенности, которая представляет собой softmax-потерю по множеству классов

 - индикатор совпадения i рамки с j фактической рамкой класса p.

N – количество совпадающих рамок

Одним из критериев оценки того, насколько хорошо была предсказана рамка является IOU, Intersection Over Union, что является отношением площади пересечения двух рамок к их общей площади. На рисунке 10 показано, как он высчитывается.

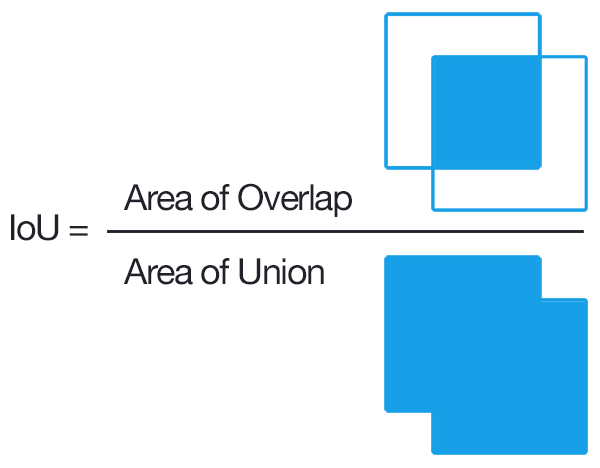


Рисунок 10 – Подсчет IOU

Помимо этого, данный критерий используется в алгоритме Non-max suppression для подавления рамок, предсказанных для одного объекта несколько раз.

## Выбор алгоритма трекинга

В данной задаче трекинг является очень важной составляющей.

Во-первых, трекинг необходим для слежения за объектом при упрощенном формировании данных.

Во-вторых, трекинг нужен для слежения за объектом при потере его из кадра. Основной алгоритм – как только детектирование объекта успешно, рамка передается в трекер, который продолжает следить за объектом.

В библиотеке OpenCV наиболее эффективно работающими алгоритмами трекинга являются CSRT и KCF. Сравним их.

CSRT работает на основе обучения фильтра корреляции со сжатыми признаками (HoG-преобразования и цвет). После этого фильтр используется для поиска области вокруг последней известной позиции объекта.

KCF также работает на основе обучения фильтра с шаблонами, содержащими объект и кусками изображения, в которых его нет. Это позволяет трекеру искать область вокруг предыдущей позиции и понимать, что ближайшие куски изображения будут содержать объект с наибольшей вероятностью.

Сравнив CSRT и KCF на практике, была получена таблица 2. В контексте данной задачи, сделав выводы из таблицы, необходимо использовать оба трекера.

Трекер CSRT будет использоваться при формировании набора данных, поскольку в данном случае важна точность для получения качественного набора данных.

Трекер KCF будет использоваться при трекинге объекта на практике, поскольку он в два раза быстрее CSRT и достаточно точен. Помимо этого, он способен перезапускаться, если уверен, что объект потерян, в то время как CSRT продолжает работать даже если объект потерян. Это позволяет снова производить детектирование и обновлять рамки вокруг объекта.

Таблица 2 – Сравнение алгоритмов трекинга CSRT и KCF

|  |  |  |
| --- | --- | --- |
|  | **Плюсы** | **Минусы** |
| **CSRT** | - Медленнее, но точнее KCF  - Устойчив при непредсказуемом движении  - Обучается на единственном шаблне  - Может восстанавливаться при неудачах  - Может быть устойчив при пропуске кадров  - Адаптируется к масштабу, повороту | - Плохо восстанавливается при неудачах  - Фокусируется на окружающих областях, что может привести к смещению  - Не восстанавливается, если объект уходит за кадр  - Не восстанавливается при нескольких последовательных неудачах |
| **KCF** | - В два раза быстрее CSRT  - Адаптируется к масштабу, повороту  - Обучается на единственном шаблне  - Способность к перезапуску при неудаче | - Плохо восстанавливается при неудачах  - Не восстанавливается при нескольких последовательных неудачах  - Не учитывает движении при расчете |

## Сравнение характеристик Raspberry Pi 3 и современного ноутбука

Изначально планировалось реализовывать данную задачу исключетельно на микрокомпьютере Raspberry Pi 3. Проанализируем его характеристики.



Рисунок 11 – Внешний вид Raspberry Pi 3

На рисунке 11 представлен внешний вид микрокомпьютера Raspberry Pi 3, а в таблице 3 – его характеристики.

Как можно увидеть из таблицы 3, аппаратная составляющая данного устройства не способна выполнить задачу детектирования с приемлемой производительностью.

Однако, другим вариантом решения данной задачи является использование более мощно компьютера, находящегося с Raspberry в одной локальной сети Wi-Fi, на котором будут производиться все трудоемкие вычисления. В таком случае, на Raspberry Pi будет ложиться задача только обработки изображения и его посылки на ПК. В свою очередь, компьютер будет отправлять сигнал обратной связи для формирования управления.

Таблица 3 – Характеристики Raspberry Pi 3 B+

|  |  |
| --- | --- |
| **Процессор** | 64-битный четырёхъядерный ARMv8 Cortex-A53 процессор с тактовой частотой 1.4 ГГц; 16 КБ cache L1 и 512 КБ cache L2 |
| **Графический процессор** | Двухъядерный процессор (GPU) VideoCore IV® (3D GPU @ 300 МГц, видео GPU @ 400 МГц) поддерживает стандарты OpenGL ES 2.0, OpenVG, MPEG-2, VC-1 и способен кодировать, декодировать и выводить Full HD-видео (1080p, 30 FPS, H.264 High-Profil) |
| **ОЗУ** | 1 ГБ SDRAM LPDDR2 |
| **Ethernet** | 10/100/1000 Мбит Gigabit Ethernet (через USB 2.0) (контроллер LAN7515 — USB 2.0 Hub и Ethernet) |
| **Wi-Fi/Bluetooth** | 2.4 ГГц и 5 ГГц IEEE 802.11.b/g/n/ac WI-FI и Bluetooth 4.2 Low Energy (BLE), обеспечиваемые микросхемой Cypress CYW43455 |
| **Видео вход** | 1 x CSI-2 для подключения камеры по интерфейсу MIPI |
| **USB-порты** | 4 порта USB 2.0 через USB hub в Microchip LAN7515 |
| **Периферия** | 40 портов ввода-вывода общего назначения (GPIO), UART (Serial), I²C/TWI, SPI с селектором между двумя устройствами; пины питания: 3,3 В, 5 В и земля. |
| **ОС** | Ubuntu, Debian, Fedora, Arch Linux, Gentoo, RISC OS, Android, Firefox OS, NetBSD, FreeBSD, Slackware, Tiny Core Linux, Windows 10 IOT |

В качестве ПК был выбран ноутбук MSI GL63 8SC. Его характеристики представлены в таблице 4. Данное устройство является достаточно мощным для того, чтобы производить обучение на произвольном наборе данных и детектирование. Помимо этого, на нем же будет тестироваться детектирование, поэтому важно знать характеристики камеры. Кроме того, будет рассматриваться возможность посылки сигнала управления на AtMega.

Таблица 4 – Наиболее важные характеристики MSI GL63 8SC для данной задачи

|  |  |
| --- | --- |
| **Процессор** | Inter Core i7-8750H CPU 2.20 GHz |
| **Оперативная память** | 8 Gb |
| **Видеокарта** | NVIDIA GTX 1650 |
| **Сеть** | Gb LAN Killer E2500 V2  Intel Wireless-AC 9560 + BT 2x2 |
| **Веб-камера** | HD type (30fps, 720p) |

Как можно видеть, данное устройство идеально подходит для работы с задачами обработки изображений, детектирования и тонкой настройки. В нем установлена видеокарта NVIDIA GTX 1650, поддерживающая вычисления на GPU.

## Вывод

На основе проделанного выше анализа можно прийти к выводу о том, что задача детектирования произвольного объекта в реальном времени реализуема. Для этого необходимо будет выполнить следующие подзадачи:

1. Формирование набора данных на основе алгоритма трекинга

2. Дообучение модели детектирования MobileNet SSD на сформированном наборе данных

3. Подбор гиперпараметров модели MobileNet SSD при помощи генетического алгоритма

4. Детектирование объекта в видеопотоке

5. Автоматизация пунктов 1, 2 и 4

6. Связь Raspberry Pi и ноутбука

Все вычислительно сложные операции будут выполняться на ноутбуке MSI, а Raspberry Pi будет играть роли IP-камеры, посылающей видеопоток на ноутбук.

# Проектно-конструкторская часть

## Составляющие автономной мобильной платформы



Рисунок 12 – Функциональная схема микроконтроллера автономной платформы Robot4

Для выполнения вышепоставленных задач была использована мобильная платформа на основе микрокомпьютера Raspberry Pi3 и микроконтроллера AtMega1281, представленная на рисунке 12.

Микрокомпьютер Raspberry Pi 3 Model B+ - это функциональная сборка, оптимальная для построения автоматизированных систем благодаря своей компактности и большому объему оперативной памяти. Он снабжен контроллерами Wi-Fi и Bluetooth и может работать с внешними устройствами ввода-вывода, в том числе с камерой. Микрокомпьютер Raspberry Pi служит для отправки команд на AtMega1281. К нему подключена камера для определения положения в пространстве.

Микроконтроллер AtMega1281 является высокопроизводительным восьмиразрядным микроконтроллером на базе микросхемы AVR RISC и имеет 54 линии ввода-вывода общего назначения, 32 рабочих регистра общего назначения, счетчик реального времени, 2 восьмиразрядных таймера-счетчика и 4 шестнадцатиразрядных.

Ввод-вывод данных на AtMega1281 организован с помощью портов USART и COM-портов управляющих устройств. Данный микроконтроллер имеет два порта USART, один из которых предназначен для управления с Raspberry Pi, а второй используется для передачи данных через Bluetooth или Wi-Fi.



## Сравнение программных сред MatLab и Python

Современные мобильные системы автономного типа должны ориентироваться в динамической среде, где перемещаются несколько объектов. Для анализа такой среды естественно использовать техническое зрение. Формируемый для искусственных систем видеопоток несёт большой объём информации, которую нужно извлечь и использовать.

Неотъемлемой частью задачи технического зрения является распознавание (детектирование) образов. Данная задача имеет множество способов решения. У каждого способа есть свои преимущества и недостатки (вычислительная сложность, точность и занимаемая память). Необходимо выбрать наиболее оптимальный способ для её реализации на мобильной платформе Raspberry Pi.

В идеале входными данными является изображение с камеры мобильной платформы любого качества и размера. На выходе необходимо получить правильно распознанный объект на изображении.

Основная цель – максимально быстро и эффективно распознать объект на изображении с минимальными задержками, рисунок 13.

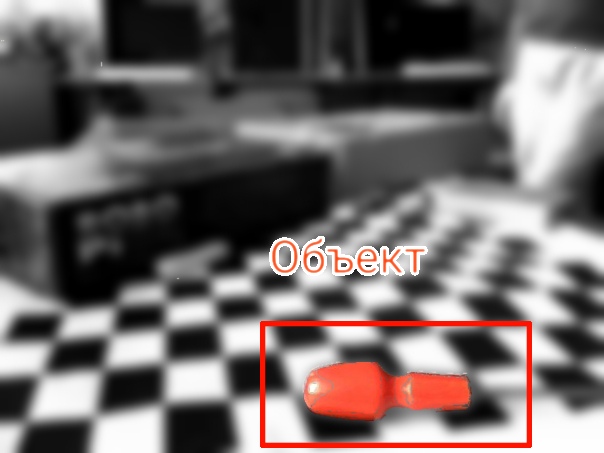


Рисунок 13 – Пример выполненной задачи

Было рассмотрено и проанализировано две среды разработки для решения задачи обучения модели детектирования – математический пакет MatLab и Python (рисунок 14).

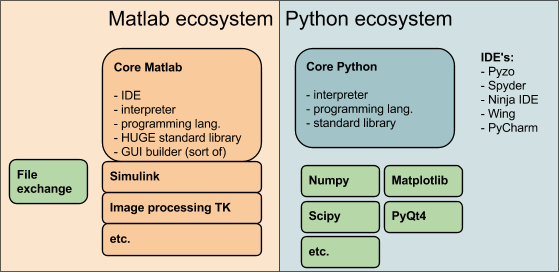


Рисунок 14 – Сравнение MatLab и Python [8]

Таблица 5 – Сравнение возможностей сред разработки Python и Matlab в рамках поставленной задачи

|  |  |  |
| --- | --- | --- |
|  | Python | Matlab |
| Наличие библиотек (пакетов, фреймворков) машинного обучения | Да | Да |
| Наличие большого встроенного инструментария | Требуется отдельная установка библиотек | Да |
| Удобство построения архитектур нейронных сетей любой сложности | Да | Нет |
| Возможность сброки в .exe-файл | Да | Требуется ядро MatLab, занимающее много места |
| Бесплатный | Да | Нет |
| Возможность использовать в коммерческих продуктах | Да | С ограничениями и налогом |

Как можно видеть в таблице 5 на рисунке 14, в целом, среды довольно похожи. Однако, у каждой среды есть свои преимущества и недостатки. Для того, чтобы выяснить, какая среда наилучшим образом подойдет для решения данной задачи, попробуем решить задачу распознавания в обеих средах.

Для этого возьмём набор данных Fashion MNIST, состоящий из десяти классов одежды, рисунок 15.



Рисунок 15 – Часть набора данных Fashion MNIST

Fashion MNIST имеет 70000 изображений и меток. Будем использовать 60000 для обучения нейронной сети и 10000 для оценки точности. Каждое изображение данного набора примеров для обучения имеет размер 28х28 и является черно-белым.

Перед обучением все изображения обрабатываются, поскольку значения пикселей изображений находятся в диапазоне от 0 до 255, в то время как для обучения требуется диапазон пикселей от 0 до 1.

Начнём тестирование удобства работы со среды Python.

Построение модели будет производиться в библиотеке машинного обучения tensorflow keras. Большая часть глубокого обучения состоит в объединении простых слоев. Первый слой - tf.keras.layers.Flatten. Он переформирует 2d-массив 28х28 пикселей в 1d-массив 784 пикселей. Затем идут 2 слоя tf.keras.layers.Dense. Это плотно связанные нейронные слои. Первый содержит 128 нейронов, а второй – 10 узлов tf.nn.softmax, который возвращает массив из десяти вероятностных оценок, сумма которых равна 1. Каждый узел содержит оценку, которая указывает вероятность того, что определенное изображение относится к одному из 10 классов.

При компиляции модели формируется несколько настроек: Loss function (функция потери, измеряющая точность модели), Optimizer (оптимизатор для регуляризации модели и уменьшения вероятности переобучения), Metrics (метрики, для контроля этапов обучения).

Обучение модели состоит из трёх условных этапов:

1) Подача изображений и меток на вход модели

2) Модель учится ассоциировать изображения и метки

3) Модель делает прогнозы на основе тестовых меток и изображений и оценивает точность.

Помимо этого, указывается количество эпох обучения модели. В нашем случае укажем пять, рисунок 16.

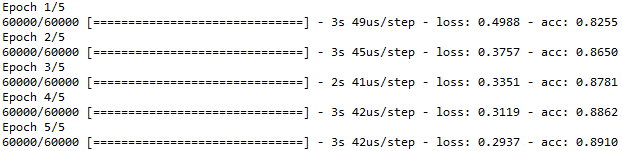


Рисунок 16 – Процесс обучения модели

Оценив точность, можно получить, что она равна примерно 87%.

Для тестирования готовой модели используются ключевые слова model.predict.

Прогнозирование метки одного изображения выдаст следующий массив:

array([1.3320119e-05, 3.7148382e-08, 1.6435710e-06, 5.8136345e-07,

2.2625582e-06, 4.7396559e-02, 1.1001509e-05, 1.0058163e-01,

2.2156924e-04, 8.5177135e-01], dtype=float32)

Каждый элемент массива описывает уверенность модели в принадлежности данного изображения к определенной метки. В данном случае максимальное значение имеет десятый элемент, рисунок 17.

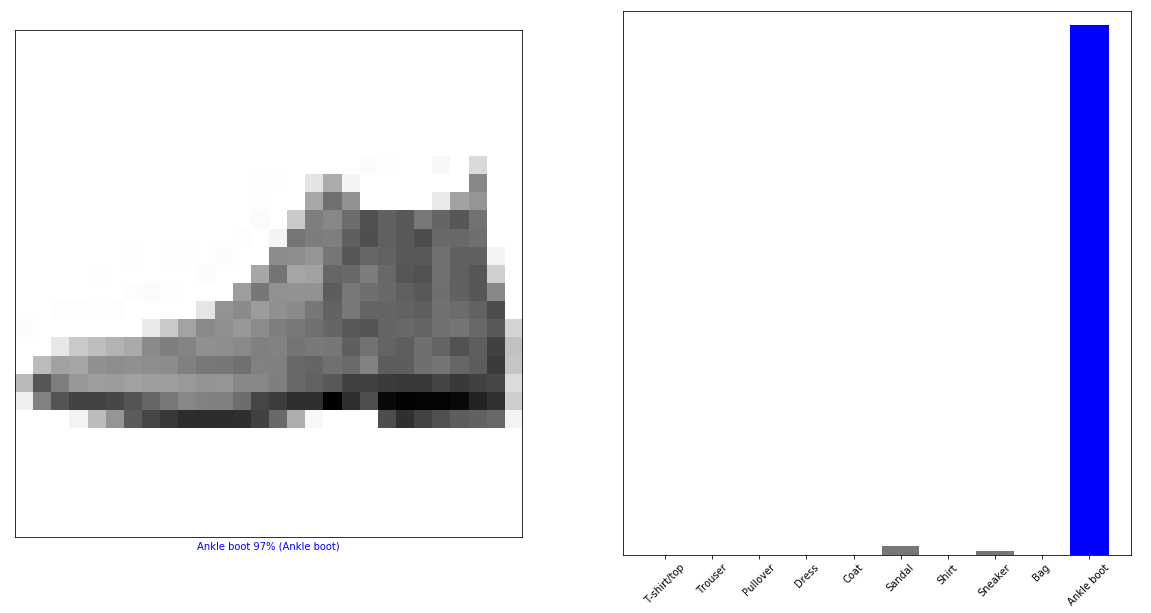


Рисунок 17 – Результат прогнозирования метки для изображения

Таким образом, модель осуществила правильное предсказание.

Используя данный принцип, можно научить модель распознавать любой объект.

Время распознавания: 0.016

Ошибка неверного включения в класс:13.38 %

Вес построенной модели: 413 Кб.

Количество эпох: 5.

Следует упомянуть, что с архитектурой модели можно экспериментировать. Keras позволяет строить нейронные сети любой размерности и архитектуры. Например, добавление сверточных слоёв в данную архитектуру способно увеличить вероятность предсказания верной метки.

Воспользуемся инструментарием MatLab для решения схожей задачи. Для этого воспользуемся командой nnstart, рисунок 18.

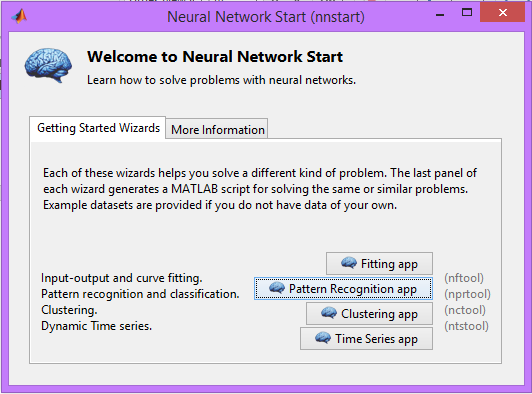


Рисунок 18 – Интерфейс тулбокса нейронной сети MatLab

Для выполнения поставленной задачи необходимо выбрать Pattern recognition and classification.

Далее будет вызван интерфейс для быстрой настройки нейронной сети и загрузки данных в нее.

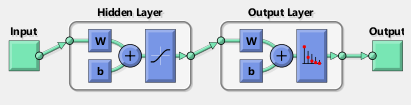


Рисунок 19 – Блок-схема предустановленной нейронной сети MatLab

Теперь нейронной сети требуется подать матрицу входных данных и матрицу целей.



Рисунок 20 – Интерфейс для подачи данных в нейронную сеть

Из всех изображений набора данных была получена одна матрица свойств (пикселей) размерностью [количество свойств (пикселей) х количество изображений]. Каждой строке этой матрицы соответствует элемент матрицы-столбца целей. Построим нейронную сеть, аналогичную той, что была построена в библиотеке Python tensorflow с количеством нейронов, равным 128.

Оценим построенную модель.

Время распознавания: 0.02

Ошибка неверного включения в класс:16.35 %

Вес построенной модели: 4889 Кб.

Количество эпох: 30.

Количество эпох было выбрано большим, поскольку на 5 эпохах было видно явное недообучение модели.

Python имеет больше преимуществ с точки зрения возможностей реализации архитектур различной сложности, а также является бесплатным продуктом. Помимо этого, архитектура нейронной сети tensorflow показала себя лучшей по всем параметрам относительно аналогичной архитектуры Matlab. Следовательно, выбор был сделан именно в пользу Python.

## Выбор программной составляющей для решения задачи

В качестве операционной системы для разработки была выбрана Windows 10, поскольку на сегодняшний день она является наиболее знакомой и популярной среди пользователей. Однако, следует упомянуть, что необходимые алгоритмы машинного обучения также работают на Linux и Linux-подобных ОС.

Как было сказано выше, в качестве среды разработки был выбран Python. Однако, помимо этого необходимо установить библиотеки машинного обучения, которых, в общем и целом, достаточно много. Следовательно, необходимо провести сравнение. Для этого составим таблицу сравнения двух самых популярных библиотек глубокого машинного обучения – tensorflow (Google) [9] и pytorch (Facebook) [10].

Таблица 6 – Сравнение двух популярных библиотек машинного обучения на Python

|  |  |  |
| --- | --- | --- |
|  | Tensorflow | Pytorch |
| Вычислительные графы | Статические графы | Динамические графы |
| Простота освоения | Менее прост в освоении | Прост в освоении |
| Популярность | Более популярен | Меньшее сообщество пользователей |
| Наличие удобного интерфейса для слежения за процессом обучения | Tensorboard | Нет, но есть возможность связать Pytorch с Tensorboard |

Таблица 6 показывает основные отличия между tensorflow и pytorch. Tensorflow, в отличие от Pytorch, сначала инициализирует весь вычислительный граф модели и затем запускает модель. В то же время Pytorch способен манипулировать графом прямо во время обучения. Помимо этого, tensorflow сложнее освоить, он менее интуитивен по сравнению с pytorch. Однако, tensorflow более популярен, что означает большее количество реализованных решений в открытом доступе с использованием данной библиотеки. Кроме того, tensorflow имеет очень удобный интерфейс, в котором можно проследить за процессом обучения модели. Несмотря на то, что в pytorch нету такого интерфейса, есть возможность объединить его с tensorboard, что, однако, требует дополнительных настроек и установки tensorflow от пользователя.

Отсюда был сделан выбор в пользу tensorflow, который значительно увеличит скорость работы как в процессе установки, так и при поиске решения, от которого можно будет отталкиваться при выполнении задачи.

## Тестирование Object Detection API

Для того, чтобы протестировать Object Detection API на работоспособность достаточно запустить ноутбук object\_detection\_tutorial.ipynb, находящийся в репозитории tensorflow models. На рисунке 21 можно видеть результат детектирования.



Рисунок 21 – Тестирование Object Detection API на одном изображении

Изменив данную программу, можно проверить скорость детектирования объектов с веб-камеры в реальном времени. На рисунке 22 показан результат детектирования объектов предобученной моделью. Скорость детектирования – 17 кадров в секунду благодаря ускорению на GPU.

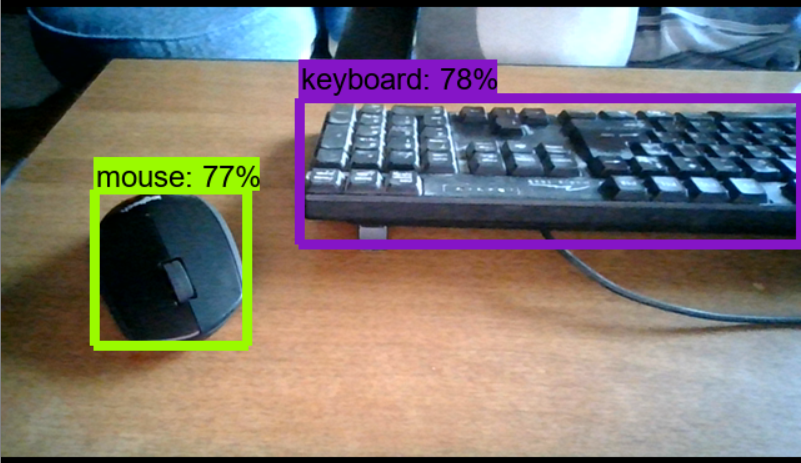


Рисунок 22 – Детектирование объектов заранее обученной нейронной сетью MobileNet SSD

## Написание программы формирования набора данных для тонкой настройки модели MobileNet SSD

Первым шагом при тонкой настройке любой модели глубокого обучения является формирование набора данных. При этом обучение модели производится не с нуля, то есть вполне достаточно иметь 100-200 изображений для одного класса. Однако, их количество не ограничено и чем больше данных, тем качественнее будет детектирование. При ручной разметке используется программа LabelImg, в которой каждому изображению ставится в соответствие .xml-файл с координатами рамок и названием класса. Однако, процесс разметки должен быть автоматизирован для максимального упрощения работы с программой.

Object Detection API имеет свою специфику входных данных, поэтому необходимо получить правильную структуру и формат набора данных. На рисунке 23 показана примерная структура папок набора данных.



Рисунок 23 – Структура папок набора данных

Рассмотрим подробнее, для чего нужна каждая из директорий.

* train\_annot – нужна для .xml-файлов, содержащих рамки изображений тренировочного набора данных
* train\_images – нужна для изображений тренировочного набора данных
* valid\_annot – нужна для .xml-файлов, содержащих рамки изображений тестового набора данных
* valid\_images – нужна для изображений тестового набора данных

valid\_annot и valid\_images нужны для контроля качества модели в реальном времени при обучении.

Необходимо разработать программу на основе алгоритма трекинга, которая будет автоматически заполнять данный набор данных. Она будет работать следующим образом:

1. Пользователь запускает программу и вводит имя класса объекта, для которого необходимо сформировать рамки. При этом запускается видеопоток
2. Пользователь нажимает «s» на клавиатуре для формирования рамки вокруг объекта при помощи удержания левой кнопки мыши
3. Пользователь нажимает «Пробел» для запуска алгоритма трекинга с выбранной рамкой
4. Пользователь нажимает «t» для формирования файла изображения и .xml-файла с рамкой и помещения их в папки train\_images и train\_annot соответственно
5. Пользователь нажимает «v» для формирования файла изображения и .xml-файла с рамкой и помещения их в папки valid\_images и valid\_annot соответственно
6. Пользователь нажимает «q» для выхода из программы. Также программу можно запустить заново для формирования набора данных для нового класса
7. Пользователь может добавить готовые файлы изображений и рамок в набор данных, предварительно переименовав их при этом
8. Пользователь нажимает «с» для запуска тонкой настройки модели для сформированного набора данных (данных пункт рассмотрен далее более подробно)

Блок-схема данного алгоритма приведена на рисунке 24.

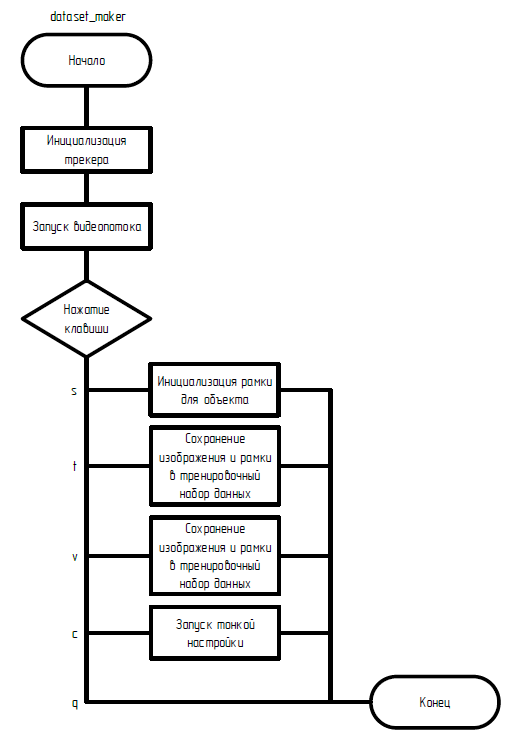


Рисунок 24 – Блока схема алгоритма автоматического формирования набора данных

Задача формирования набора данных решена. Теперь необходимо автоматизировать процесс тонкой настройки модели.

## Автоматизация процесса тонкой настройки модели детектирования

Как уже было сказано выше, Object Detection API имеет свою специфику входных данных. Он принимает на вход .record-файлы, которые получаются из изображений и файлов разметки и являются зашифрованным представлением, понятным любой модели из Object Detection API.

Помимо этого, требуется файл .pbtxt, являющийся файлом классовой разметки (списком классов с присвоенными им id-номерами), а также .config-файл, в котором указываются все настройки модели для дообучения.

Для начала сгенерируем .record-файлы. Для этого напишем программу generate\_tfrecord.py. Она будет принимать на вход .csv-файлы (рисунок 25, 26), для чего также будет написана программа xml\_to\_csv.py.

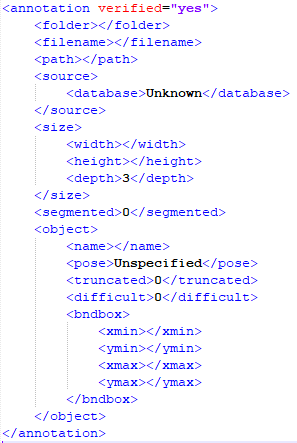


Рисунок 25 – Шаблон .xml-файла для одного изображения

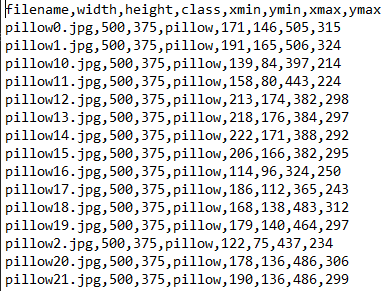


Рисунок 26 – Шаблон .csv-файла для набора изображений

После получения нужного формата, необходимо сформировать карту меток, которая выглядит так, как показано на рисунке 27.

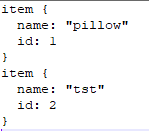


Рисунок 27 – Карта метод для набора данных из двух классов

Помимо этого, необходимо правильно задать настройки в .config-файле, в котором указываются пути к наборам данных, количество классов и настройки самой модели. На рисунках 28 и 29 нужные его части можно рассмотреть подробнее.

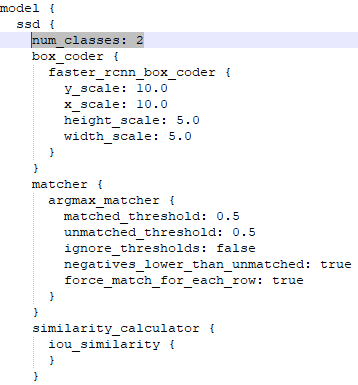


Рисунок 28 – Количество классов в файле .config

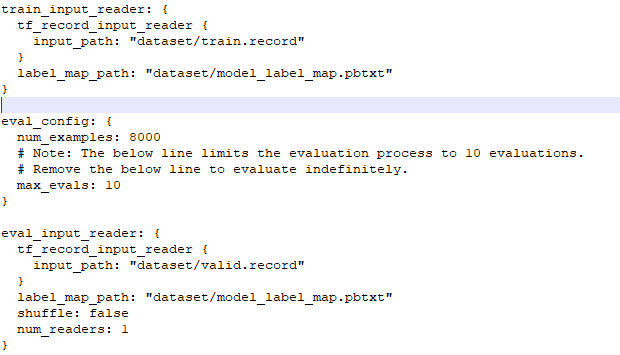


Рисунок 29 – Указание путей к набору данных в файле .config

На основе этих данных будет производиться автоматизация процесса тонкой настройки модели. В конечном итоге, блок-схема процесса обучения представлена на рисунке 30.

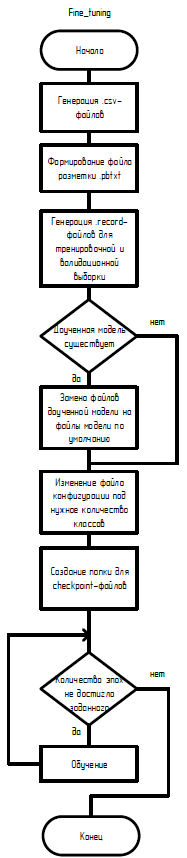


Рисунок 30 – Блок-схема автоматизированного процесса обучения

Помимо этого, необходимо оценить ограничения, накладываемые при обучении. Для этого нужно проаналзировать среду, в которой будет двигаться объект и само перемещение объекта для данной задачи, а также специфику модели. Были выявлены следующие ограничения:

* Количество изображений объекта с рамками – не менее 50;
* Камера не должна быть повернута, то есть параллельна полу (обеспечена ракурсная устойчивость);
* Объект может быть в любой точке сцены в кадре;
* Время обучения – около 1 часа 30 минут (получено экспериментальным путём, но может варьироваться);

Исходя из этих ограничений, сформируем набор данных, состоящий из двух классов. Первый класс – tst (лицо человека), второй класс – pillow (подушка). Примеры изображений указанных классов представлены на рисунке 31.

а)  б)

Рисунок 31 – Изображения классов тренировочного набора данных для проверки модели на работоспособность:

а – класс pillow; б – класс tst

Для запуска модели на дообучение следует ввести следующую команду в командном окне:

python train.py --logtostderr --train\_dir = training --pipeline\_config\_path = models/ssdmn\_fine\_tuned/ssd\_mobilenet\_v1\_coco.config

После этого, начнется процесс обучения, рисунок 32. Обычно достаточно около 15000 шагов для обеспечения конвергенции модели, что занимает около 90 минут.

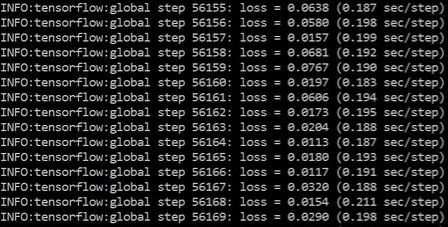


Рисунок 32 – Процесс обучения модели

Воспользуемся tensorboard для отслеживания процесса обучения модели и посмотрим на поведение функции потерь. Функция потерь успешной обученной модели представлена на рисунке 33.

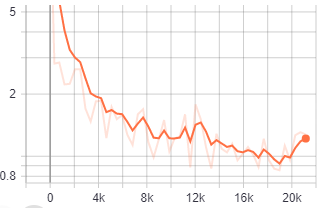


Рисунок 33 – Зависимость функции потерь от количества шагов (блекло-оранжевый график – функция потерь, ярко оранжевый график – её сглаженная версия)

На рисунке 33 видно, что loss-функция (функция потерь) уменьшается со временем и в конечном итоге её значение постепенно приближается к некоторому установившемуся, то есть обучение устойчиво.

Рассчитаем медиану и доверительный интервал для данной системы. Результат вычисления представлен на рисунке 34. Для вычисления скользящих бралось окно в 2000 эпох. Функция потерь – истинное значение функции стоимости в течение обучения, скользящее среднее – её усредненное значение, показывающее, что она постепенно уменьшается, скользящее СКО – границы, в которых могла находиться функция потерь.

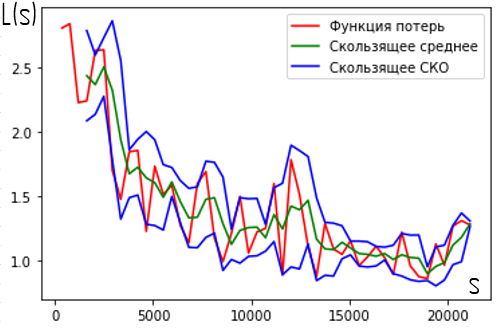


Рисунок 34 – Зависимость функции потерь от количества шагов и её доверительный интервал (красный график – функция потерь, зеленая линия – её скользящее среднее, синие линии – её доверительный интервал), 21000 эпох

Помимо этого, рассчитаем разброс по обучению для данных гиперпараметров, рисунок 35.

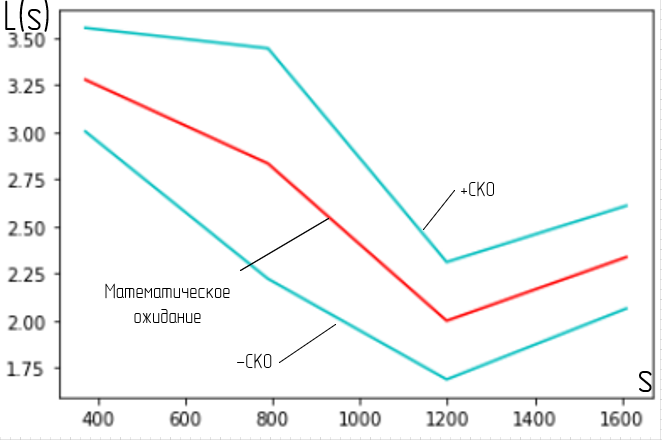


Рисунок 35 – График усредненной функции потерь для оценки разброса по обучению с оптимальными гиперпараметрами при 5 перезапусках, 2000 эпох

## Подбор гиперпараметров модели

Как и во всех задачах машинного обучения, одной из важнейших задач является подбор оптимальных гиперпараметров для обеспечения наилучшей точности модели. Object detection API имеет некие параметры по умолчанию, однако, нельзя гарантировать, что они оптимальны. Необходимо либо удостовериться в их оптимальности, либо подобрать наилучшие параметры самостоятельно.

Подход AutoML (Automatic Machine Learning) постепенно набирает популярность среди пользователей, поскольку избавляет пользователя от монотонной настройки гиперпараметров пользователем вручную и делает это автоматически. Большая часть таких подходов реализовывается с помощью генетических алгоритмов.

Следует учесть, что AutoML имеет следующие недостатки:

1. Очень медленная работа (для получения какого-либо результата необходимо обучать заново и тестировать модель как минимум 10 раз)

2. Параметры генетического алгоритма тоже необходимо подбирать

3. Генетический алгоритм имеет тенденцию сходиться в локальный минимум

Несмотря на вышеперечисленные недостатки, было принято решение написать собственный генетический алгоритм для AutoML, который будет подбирать наиболее оптимальные гиперпараметры для модели.

Генетический алгоритм – это алгоритм, вдохновленный эволюционным законом природы. Каждая его особь проходит отбор по пригодности (fitness), способна мутировать и размножаться.

Для того, чтобы сделать входные данные «понятными» генетическому алгоритму, необходимо закодировать их в вектор, который будет являться особью.

Вектор состоит из следующих гиперпараметров, для которых были заданы границы:

* Начальный коэффициент обучения LR  [0.00001; 0.15]
* Коэффициент затухания DF  [0.7; 0.99]
* Значение оптимизатора момента MOV  [0.6; 0.99]
* Распад D  [0.7; 0.99]
* Параметр оптимизации   [0.5; 1.0]

По мере прохождения генетического алгоритма особи отбираются по фитнесс-функции, разможаются и мутируют. Рассмотрим каждый алгоритм подробнее.

Фитнесс-функция подсчитывается следующим образом:

1. В фитнесс-функцию подается особь – вектор закодированных гиперпараметров

2. Этот вектор расшифровывается и подается в модель

3. Модель обучается 200 шагов, после чего обучение останавливается

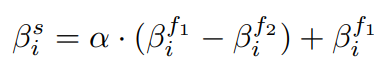
4. На основании последних значений (одного или нескольких) кривой функции потерь можно судить о точности и скорости сходимости модели

5. Последнее значение функции потерь возвращается в качестве фитнесс-функции (решается задача её минимизации на основе гиперпараметров)

6. Значения фитнесс функции, выходящие за установленные рамки, отбрасываются (значению фитнесс функции присваивается большое число)

На основе фитнесс-функции производится отбор турнирным методом.

Скрещивание производится по следующей формуле:

, где

 – индекс элемента особи,

– настраиваемый параметр,

 – скрещиваемые особи.

Мутация происходит с малой вероятностью. Каждый элемент особи произвольно с определенной вероятностью выбирается из данного промежутка с равномерной плотностью вероятностей.

Как показали эксперименты, гиперпараметры, подобранные генетическим алгоритмом, оказались близки к параметрам по умолчанию:

* Начальный коэффициент обучения LR = 0.004
* Коэффициент затухания DF = 0.95
* Значение оптимизатора момента MOV = 0.9
* Распад D = 0.9
* Параметр оптимизации  = 1.0

Следовательно, можно использовать их.

## Разработка интерфейса

В соответствии с вышеуказанными требованиями к набору данных был автоматизирован процесс формирования набора данных и вызова необходимых процедур и функций.

Рассмотрим, как работает автоматизированная процедура изменения файлов.

При нажатии пользователем на кнопку «c» программа выполняет следующие действия:

* + - 1. Из полученных .xml-файлов генерируются .csv-файлы
      2. Формируется файл разметки .pbtxt
      3. Дважды (для тренировочной и валидационной выборки) запускается программа generate\_tfrecord.py, на вход которой подаются входные csv-файлы и путь, где будут созданы .record-файлы
      4. Производится проверка на существование папки с моделью и, если она существует, папка очищается
      5. В папку распаковываются веса предобученной сети, а также копируется файл конфигурации
      6. В файле конфигурации прописываются нужные пути к набору данных, а также количество классов
      7. Создается папка, в которую записываются checkpoint-файлы обучающейся модели (веса сети)
      8. Начинается процесс тонкой настройки

Далее, рассмотрим алгоритм работы программы, производящей детектирование:

Из папки с checkpoint-файлами выбирается самый последний checkpoint

Запускается программа export\_inference\_graph.py, подготавливающая граф для детектирования. На её вход подается файл конфигурации и checkpoint

Производится копирование карты разметок для данной программы, а также устанавливаются пути к модели, checkpoint, и карте меток

Начинается детектирование

В конечном итоге, у пользователя есть две программы – программа для формирования набора данных и дообучения на этих данных и программа для детектирования.

Запуск программы создания данных для обучения занимает 3 секунды, а программы детектирования – около 30 секунд, что вполне приемлемо, поскольку необходимо загружать веса и создавать граф для детектирования.

## Объединение IP-камеры и вычислительной системы общим сервером

Как было сказано выше, необходимо объединить камеру raspberry pi с машиной, на которую ложатся все вычислительно ёмкие задачи. В качестве замены IP-камеры raspberry pi на время тестирования можно ограничиться камерой смартфона, установив приложение, которое транслирует изображение с камеры на определенной IP-адрес локальной сети.

Перейдя по этому адресу в браузере компьютера, можно видеть трансляцию с камеры, рисунок 36.

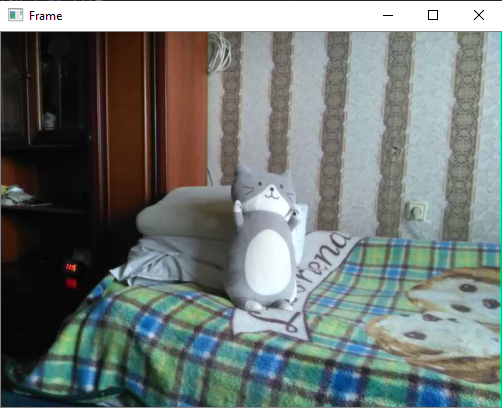


Рисунок 36 – Трансляция видеопотока с камеры смартфона

Библиотека OpenCV Python поддерживает функцию считывания такого видеопотока, что не составляет труда открыть его так же, как веб-камеру ноутбука [11].

После тестирования камеры было замечено, что есть небольшая задержка. Это связано с тем, что изображение со смартфона передается в разрешении 2К (1920х1080). Однако, это не проблема, поскольку разрешение камеры Raspberry гораздо ниже, следовательно, задержка тоже будет меньше.

Наличие данной функции обеспечивает пользователю гораздо более прострую работу с обеими программами, поскольку формировать набор данных и производить тестирование теперь можно с любых ракурсов, рисунки 37, 38.

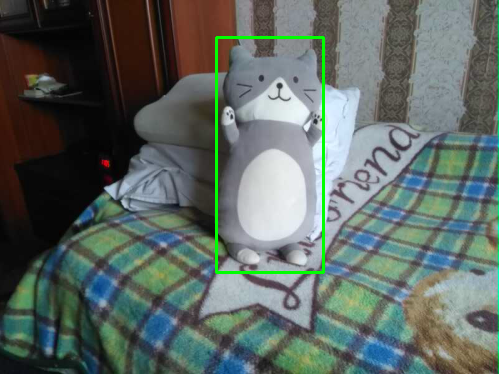


Рисунок 37 – Трекинг объекта с видеопотока камеры смартфона

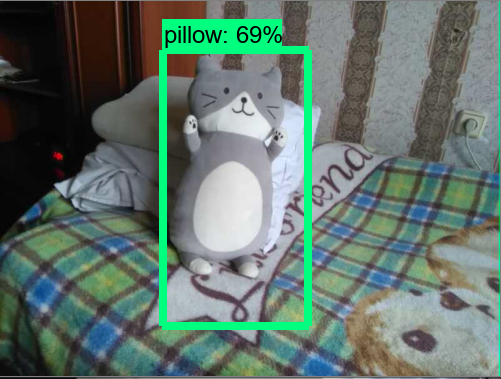


Рисунок 38 – Детекция объекта с видеопотока камеры смартфона

## Тестирование программно-аппаратной системы

Запустив программу детектирования, можно видеть, что оба объекта успешно детектируются. Результат представлен на рисунке 39.

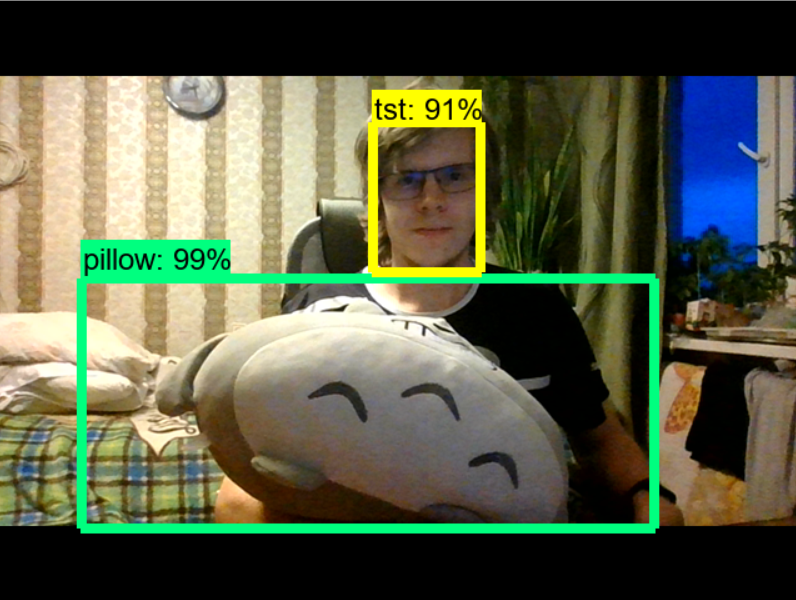


Рисунок 39 – Успешный результат детектирования при использовании доученной модели

Несмотря на это, на рисунке 40 видно, что модель плохо распознает класс pillow по двум причинам:

1. Недостаточно данных
2. Набор данных формировался в одной сцене (без изменения фона)

В связи с этим, модель начинает принимать фон за объект класса pillow. Есть два возможных решения данной проблемы:

1. Увеличить порог confidence (уверенности) сети с целью подавления нежелательных результатов
2. Увеличить набор данных, добавив больше примеров класса pillow

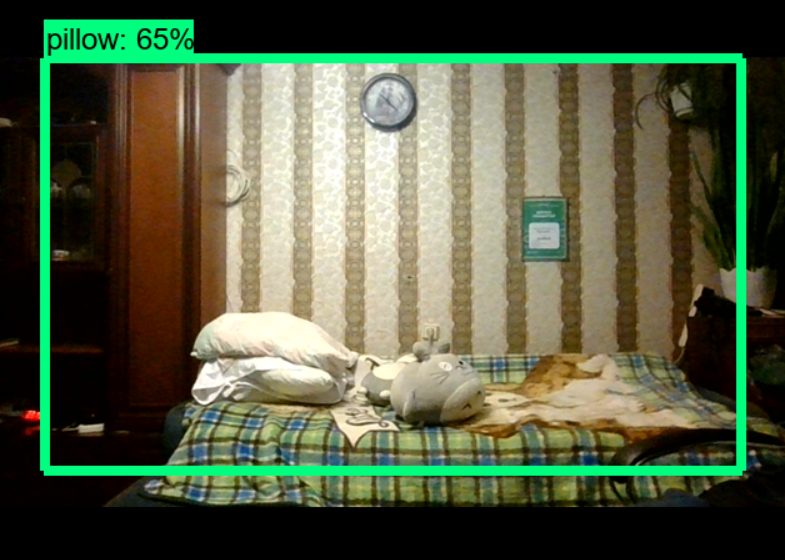


Рисунок 40 – Пример неверного детектирования класса pillow

Было отснято два видео с целью произведения оценки количества кадров с правильно детектированными классами в кадре к количеству всех кадров в видео. Получили следующие результаты: 90 правильных детекций к 109 кадру в видео – класс tst, 185 правильных детекций к 205 кадрам в видео – класс pillow.

Таким образом, было разработано два приложения: трекер для формирования набора данных и детектор. Разработанный нейросетевой детектор объектов может быть дообучен на 90 новых классах. Если количество классов превысит 90, велика вероятность, что детектирование начнет терять качество, поскольку архитектура данной сети не была рассчитана на это. Полученного результата более чем достаточно для решения задач управления в условиях помещений.

# Технологическая часть

## Экономическое обоснование дипломного проекта

В данном дипломном проекте была поставлена задача разработки детектора пользовательских объектов с возможности дообучения нейронной сети для управления автономной мобильной платформой. Данная задача разделяется на следующие подзадачи:

1. Разработка программного обеспечения для формирования пользовательского набора данных на основе алгоритма трекинга.

2. Автоматизация процесса дообучения нейронной сети на сформированном наборе данных.

3. Детектирование пользовательского объекта в реальном времени.

4. Связывание микрокомпьютера Raspberry Pi и стационарной вычислительной машины с более мощными ресурсами.

## Организация и планирование работ

Любая работа делится на три основных шага:

1) Исследовательская часть (выбор программной среды, архитектуры нейронной сети).

2) Конструкторская часть (разработка программного обеспечения для выполнения задачи);

3) Настройка, проведение экспериментов, выпуск сопроводительной документации (тестирование модели с разными гиперпараметрами, инструкция).

Для оптимального планирования работ разрабатывается календарный график, который показывает время начала каждой работы относительно начала проекта и её продолжительность.

Трудоемкость и состав исполнителей определяется по технологии для каждой конкретной работы.

Таблица 7 – Планирование работ по теме

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Перечень работ | Исполнители | Трудоёмкость работы (дни) |
| 1 | 2 | 3 | 4 |
| 1 | 1. Разработка ТЗ. | | |
| 2 | 1.1. Определение целей создания системы. | Начальник отдела Инженер-  программист | 1 |
| 3 | 1.2. Разработка требований и ограничений к системе и ее частям. | Начальник отдела Инженер-  программист | 3 |
| 4 | 1.3. Расчет технико-экономических  показателей. | Инженер-  программист | 3 |
| 5 | 2. Разработка технического проекта. | | |
| 6 | 2.1. Проведение анализа методов разработки системы | Инженер-  программист | 5 |
| 7 | 2.2. Выбор методов и средств для разработки системы | Инженер-  программист | 1 |
| 8 | 3. Разработка рабочего проекта. |  |  |
| 9 | 3.1. Написание программы для автоматизации формирования набора данных | Инженер- программист | 3 |
| 10 | 3.2. Поиск и тонкая настройка алгоритма детектирования | Инженер- программист. | 5 |
| 11 | 3.3. Разработка пользовательского интерфейса | Инженер- программист | 3 |
| 12 | 4. Ввод системы в эксплуатацию |  |  |
| 13 | 4.1. Выявление и устранение  неполадок. | Инженер-программист | 5 |
| 14 | 4.2. Проведение экспериментов. | Инженер-программист | 3 |
| 15 | 4.3.Разработка технической документации | Инженер-программист | 3 |

По таблице 7 составляется оперативный календарный план-график выполнения работ.

## Определение стоимости специального оборудования

Для расчета стоимости материалов, необходимых для выполнения задачи составим таблицу.

Для данного проекта используется новое покупное оборудование, таблица 8.

Таблица 8 – Смета на приобретение специального оборудования и ПО

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Наименование изделий | Количество, шт | Стоимость, руб. | Общая стоимостьруб. |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Raspberry Pi 3 | 1 | 3500 | 3500 |
| 2 | AtMega1281 | 1 | 700 | 700 |
| 3 | Raspberry Pi камера 640х480 | 1 | 1000 | 1000 |
| Итого стоимость оборудования: | | | | 5200 |

Как можно видеть из таблицы 8, для получения необходимого функционала, достаточно иметь три вышеупомянутых составляющих. Дополнительные детали (в виде проводов, колес, корпуса) здесь не упомянуты и относятся к графе «прочие затраты» таблицы 11.

## Расчет основной заработной платы

Заработная плата непосредственных исполнителей рассчитывается исходя из количества исполнителей, их оклада и времени, затраченного на выполнения работ, таблица 9.

Так как календарные сроки проведения работ по проектированию неизвестны, то принято, что в месяце 20 рабочих дней.

Таблица 9 – Заработная плата основного персонала

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Исполнители  2 | Месячный оклад, руб.  3 | Оклад, руб./день  4 | Количество  отработанны х дней  5 | Выплаты, руб.  6 |
| 1 |
| 1 | Начальник  отдела. | 70000 | 3500 | 4 | 14000 |
| 2 | Инженер-  программист | 50000 | 2500 | 35 | 87500 |
| Итого выплаты по основной зарплате: | | | | | 101500 |

Выплаты по основной заработной плате составят 101500 руб.

Дополнительная заработная плата по данной теме не предусматривается.

Тарифы страховых взносов представлены в таблице 10.

Таблица 10 – Тарифы страховых взносов на 2020 год

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| В ПФР на обязательное пенсионное страхование | | в ФСС на страхование на случай  временной нетрудоспособности и материнства (ВНиМ) | | в ФФОМС  на ОМС |
| С суммы | С суммы | С суммы | С суммы | 5,1% |
| выплат в | превышения | выплат в | превышения |  |
| пределах | установленной | пределах | установленной |  |
| установленной | предельной | установленной | предельной |  |
| предельной | величины базы | предельной | величины базы |  |
| величины базы |  | величины базы |  |  |
| 22% | 10% | 2,9% | Не |  |
|  |  |  | начисляются |  |

Сот = (Зосн + Здоп) \* 0,3 = 101500 \* 0,3 = 30450 руб.

в) По данной теме расходы по статьям: расходы на научные и производственные командировки; оплата работ, выполняемых сторонними организациями; не предусматривается.

Таблица 11 – Расчет статей расходов

|  |  |  |
| --- | --- | --- |
| №  п/п | Наименование статей расхода | Затраты, руб. |
| 1 | Специальное оборудование и ПО | 5200 |
| 2 | Основная зарплата | 101500 |
| 3 | Дополнительная зарплата | - |
| 4 | Сумма страховых платежей | 30450 |
| 5 | Расходы на научные и производственные  командировки | - |
| 6 | Оплата работ, выполняемых сторонними  организациями | - |
| 7 | Прочие затраты (дополнительные составляющие платформы) | 3000 |
|  | Итого затрат: | 140150 |

## 

## Оценка экономической целесообразности разработки

Средняя цена внедрения подобного ПО на предприятие составляет около 1200000 руб.

Среднегодовая заработная плата Сср.г = 60000 руб.

Собщ = 60000 \* 12 = 720000 руб.

Сумма страховых платежей 34 % от суммы заработной платы: Сот = Зос\* 0,3 = 720000 \* 0,34 = 244800 руб.

Общая статья расходов на разработку с учетом оборудования составит примерно

С = 140150+ 720000 + 244800 = 1104950 руб.

Таким образом, предполагаемая прибыль составит

П = 1200000 – 1104950 = 95050 руб.

Заключение: Предприятие понесёт большие расходы при покупке аналогичного ПО, поскольку, как было указано выше, его стоимость составляет около 1200000 рублей. Однако, при создании данного программного комплекса вручную составит всего лишь 1104950 рублей, что позволяет сэкономить 95050 рублей.

## Характеристика параметров рабочего места

В данном разделе пояснительной записки рассматриваются экологические факторы реализации программного продукта.

В задачи исполнителя, осуществляющего вышеуказанную разработку, входит взаимодействие с ПЭВМ, что предполагает наличие ряда аспектов, имеющих пагубное воздействие непосредственно на человека, а, следовательно, снижающих эффективность его труда. Следовательно, на рабочем месте инженера - программиста должны быть гарантированы соответствующие условия.

Разработка программного обеспечения производится в комнате размерами: ширина – 3 метра, длина – 5 метров, высота потолка – 2.5 метра. Имеется 1 окно, направленное на северную сторону.

Категория тяжести определяется по затрачиваемой сотрудниками энергии. Работа инженера-программиста в соответствии с СанПиНом 2.2.4.548-96 относится к категории «1а», поэтому необходимо организовать помещение с учетом оптимальных норм микроклимата для категории работ 1а. Температура воздуха для теплого времени года – 23-25оС, для холодного – 22-24оС. Относительная влажность воздуха 60-40%.

В помещении может повышаться температура в связи с наличием электрического оборудования, что может негативно сказаться на здоровье работника.

Упомянутое помещение имеет одно рабочее место с ПЭВМ для работы одного исполнителя. Учитывая размеры комнаты можно сделать вывод о том, что системы теплоотвода (вентиляционные шахты, кондиционеры) не требуются. При необходимости проветривание возможно путем открытия окон. Система центрального отопления, в свою очередь, позволяет поддерживать нужную температуру в холодное время года.

Рабочее место с ПЭВМ выполнено в соответствии с ГОСТом 12.2.003-91. Мебель рабочего места выполнена в светлых тонах. Офисное кресло с возможностью регулировки высоты и наклона спинки предназначено для правильного сидячего положения пользователя. На месте имеется подставка для ног, обеспечивающая наиболее комфортное положение сотрудника. Рабочее место достаточно свободно и не ограничивает движения пользователя.

В соответствии с СанПиНом 2.2.2/2.4.1340-03 в помещении с ПЭВМ необходимо предусмотреть естественное и искусственное освещение. Естественное освещение реализуется при помощи светопроемов, имеющих устройство для защиты от избыточного солнечного излучения - жалюзи.

Одним из важных разделом охраны труда является правильное освещение рабочего места. В кабинетах завода или компании необходимо организовывать естественное освещение, то есть освещение помещений при помощи солнечего света (прямого или отраженного), который проходит сквозь световые проемы в наружных ограждающих конструкциях, в дневное время, как более экономичное, полезное и совершенное с точки зрения медико- санитарных требований по сравнению с искусственным освещением.

Необходимо произвести расчеты искусственного освещения для обеспечения безопасных условий труда.

В приведенном помещении используются газоразрядные лампы низкого давления (люминесцентные) – дневного света (ЛД), схема которых представлена на рисунке 41.

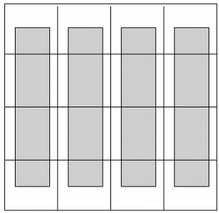


Рисунок 41 – Люминисцентная лампа

Нормативная освещенность рабочих поверхностей в помещениях, регламентирующаяся строительными правилами (СНиП 23-05-95), регулируется в зависимости от характеристики зрительной работы.

Метод коэффициента использования светового потока является одним из основных методов расчёта общего искусственного освещения. Для этого необходимо определить нужный световой поток лампы (Ф,) или светильника при известном их числе (N). Также вместо этого может определяться N при известном Фн. Расчет ведется по формуле:



где Фн, – световой поток одного светильника, лм, Ен – нормируемая освещённость, лк,

S – площадь помещения, м2,

Z – коэффициент, учитывающий отношение средней освещённости к минимальной (Z = 1.1 – для люминесцентных ламп; Z = 1,15 – для ламп накаливания),

Кз – коэффициент запаса, принимаемый в зависимости от загрязнённости воздуха в помещении по нормам СНиП 23-05-95,

N – число ламп в светильниках, при этом учитывается, что в светильнике может быть несколько ламп,

 – коэффициент использования светового потока (в долях единиц), который определяется по светотехническим таблицам. Он зависит от таких факторов, как кривая распределения света светильника и КПД, коэффициентов отражения потока, пола и стен, высоты подвеса светильника над расчётной поверхностью и индекса помещения, определяемого в зависимости от размеров.

В рассматриваемом помещении имеется верхнее центральное освещение из сочетания 16 светильников ЛПО09 и еще один светильник ЛПО09 над рабочим местом.

Исходные данные для расчета освещения показаны в Таблице 12.

Таблица 12 – Исходные данные

|  |  |
| --- | --- |
| Тип ламп | ЛЛ |
| Применяемые светильники | ЛПО09 |
| Длина помещения А, в м | 5 |
| Ширина помещения В, в м | 3 |
| Высота подвеса светильника hp, в м | 2 |
| Общее количество светильников, N | 17 |
| Поправочный коэффициент, Z | 1.2 |

Помещение имеет белый потолок ρп = 70, светлые стены ρс = 50, пол ρр =10. Для работы программиста, наименьшим различимым объектом является точка, размерами от 0.3 до 0.5 мм.

Площадь освещаемого помещения:

*S**ab*5315 *м*2 

з – коэффициент запаса определяется по таблице коэффициентов использования светового потока и принимается равным 1.4.

По формуле (2) рассчитывается индекс помещения:

*,*

где а – ширина помещения, м, b – длина помещения, м,

hp – высота подвеса светильника над расчётной поверхностью, м.

Тогда:

**

В соответствии с приведенными параметрами по специальной таблице определяется значение  = 0.46.

По таблице освещенности при различных разрядах зрительной работы для искусственного освещения (СНиП 23-05-95) определяется Енорм, при этом учитываются следующие характеристики зрительной работы исполнителя:

* 1. Высокая точность;
  2. Разряд зрительной работы – III;
  3. Фон – средний;
  4. Контраст различимого объекта с фоном – средний (в зависимости от настроек дисплея, но лучше всего использовать средние параметры контрастности).

Тогда, по специальной таблице определяется нормированное освещение Енорм.= 300 лк.

В результате световой поток по полученным значениям вычисляется как:

**

По таблице с техническими данными люминесцентных ламп по рассчитанному значению светового потока Фл и напряжению сети равному 220 В выбирается ближайшая стандартная лампа, поток которой не должен отличаться от Фл больше чем на –10 – +20%. В данном случае это будет лампа ЛДЦ20, мощностью 41 Вт и световым потоком  = 820 лм.

Таким образом, был проведен расчет характеристик необходимого

дополнительного источника искусственного освещения над рабочим столом.

Другой характеристикой рабочего места оператора ПЭВМ с монитором на базе плоских дискретных экранов (жидкокристаллические, плазменные) является его площадь. В соответствии с нормами она должна составлять не менее 4.5 м2. Данное помещение превышает эту норму, так как ранее было вычислено, что его площадь составляет 15 м2 и при этом в нем организовано только одно рабочее место.

Для внутренней отделки интерьера помещений с мониторами и ПЭВМ должны использоваться диффузно-отражающиеся материалы с коэффициентом отражения для потолка – 0.7-0.8; для стен – 0.5-0.6; для пола – 0.3-0.5. Поверхность пола в помещениях эксплуатации мониторов и ПЭВМ должна обладать антистатическими свойствами, быть ровной и нескользкой.

Также должен обеспечиваться допустимый нормами уровень шума на рабочих местах в производственных помещениях. В данном случае рабочее место оборудовано одной ПЭВМ, которая производит шум с уровнем не более 40 дБ, не превышающим допустимые значения, что соответствует требованиям СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы», а, следовательно, особых мероприятий по подавлению шума не требуется.

Достаточно важную роль играет положение монитора при работе программиста. Также должна также предусматриваться возможность регулирования экрана по разным направлениям. Монитор рекомендуется устанавливать на расстоянии не ближе 500 мм от глаз, верхняя часть дисплея должна быть на уровне глаз или чуть ниже. Также важно задать оптимальные характеристики экрана, такие как размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана, для обеспечения большей производительности работы на ПЭВМ. Монитор, установленный на рабочем месте в рассматриваемом помещении, соответствует нормам.

При эксплуатации средств вычислительной техники возникают электромагнитные поля, которые оказывают вредное воздействие на оператора ПЭВМ. Допустимые уровни электромагнитных полей (согласно СанПиН 2.2.2/2.4.1340-03), создаваемых ПЭВМ, представлены в Таблице 13.

Таблица 13 – Допустимые уровни электромагнитных полей

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование параметров | |  | ВДУ ЭМП |
| Напряженность электрического поля | в диапазоне частот 5 Гц - 2  кГц |  | 25 В/м |
| в диапазоне частот 2 кГц -  400 кГц |  | 2.5 В/м |
| Плотность магнитного потока | в диапазоне частот 5 Гц - 2  кГц |  | 250 нТл |
| в диапазоне частот 2 кГц -  400 кГц |  | 25 нТл |
| Электростатический потенциал экрана  видеомонитора | |  | 500 В |

Для уменьшения воздействия электромагнитного излучения используют мониторы с низким уровнем излучения, для этого в ЭЛТ могут устанавливаться дополнительные катушки компенсации. Воздействие электромагнитного поля также ослабляет специальный многослойный экран, имеющий проводящий слой и заземленный фильтр. В данном случае показатели электромагнитного поля не превышают допустимых показателей и соответствуют СанПиН 2.2.2/2.4.1340-03.

Необходимо, помимо всего прочего, обеспечить защиту работников от получения травм, связанных с электричеством. Прикосновение к проводам с перебоями может поразить рабочего электрическим током, если провода под напряжением.

Все используемое оборудование питается от сети переменного тока напряжением 220 В с частотой 50 Гц. Помещение имеет первый класс по классификации ПУЭ (правила устройства электроустановок), то есть там нет повышенной опасности получения электротравм.

## Утилизация люминесцентных ламп

Все люминисцентные лампы содержат ртуть (от 1 до 70 грам), которая является веществом первого класса опасности. Такая доза способна причинить вред здоровью человека.

Законодательство по RoHS (Ограничение Использования Опасных Веществ) описывает применение ртути, а также других потенциально опасных элементов в электротехническом и электронном оборудовании с целью ограничить применение основных опасных веществ в электрическом и электронном оборудовании, тем самым обеспечивая требуемый уровень безопасности людей и окружающей среды.

Рабочим элементом люминесцентных ламп является высокочистая ртуть марки «Р-0». Предельно допустимая концентрация (ПДК) содержания ртути в воздухе жилой зоны составляет 0.0003 мг на м3. При одной лампе в воздухе жилой зоны попадает 8 мг ртути, пары которой при равномерном распределении внутри расчетного помещения 5×3×2.5 (37.5 м3) даст до 0.021 мг/м3, что сильно (в 70 раз) превышает ПДК, однако в расчет не берется ртуть, содержащаяся в люминофоре. Использование в системе искусственного освещения 4-ех ламп типа ЛБ со сроком службы 1950 часов непрерывной работы увеличивает опасность превышения ПДК по ртути при бое нескольких светильников, следовательно, необходимо утилизировать лампы не реже 1 раза в год.

# ЗАКЛЮЧЕНИЕ

Рассмотрев подходы к детектированию и тонкой настроке моделей, можно сделать следующие выводы:

• Вариантов решения задачи формирования управления для мобильной автономной платформы при помощи компьютерного зрения довольно много и для того, чтобы добиться высокой производительности, их необходимо тестировать в различных комбинациях и реализациях;

• В рамках высокой ограниченности вычислительных ресурсов сложно подобрать оптимальный алгоритм детектирования, который, помимо всего прочего, будет адекватно работать;

• Использование вычислительных ресурсов микрокомпьютера Raspberry Pi 3 автономной мобильной платформы нерационально и необходимо прибегать к использованию переноса тяжелоёмких процессов на более мощную аппаратную составляющую при помощи локальной сети Wi-Fi;

• Перенос вычислительных мощностей при обучении и детектировании на графический процессор даёт большой прирост в производительности;

• Точность алгоритма детектирования после тонкой настройки напрямую зависит от количества подаваемых данных при обучении;

• Данные должны быть разнообразны и включать в себя объект с разных ракурсов и с разным фоном;

В ходе работы с алгоритмами детектирования и тонкой настройки с целью работы в рамках сцены, в которой происходит движение мобильной автономной платформы, а также их анализа формируются навыки работы с нейронными сетями различной архитектуры, приобретаются компетенции в области машинного обучения и необходимый для решения прикладных задач опыт.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Goodfellow I., Bengio Y., Courville A. Deep Learning, издательство: The MIT Press, 2016. — 800 с. (дата обращения 10.05.2020)

2. B. Mehlig. Artificial Neural Networks. Chalmers and Gothenburg University course, 2019. [Электронный ресурс] – Электрон. журн. – URL: https://arxiv.org/abs/1901.05639. (дата обращения: 01.05.2020)

3. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Электронный ресурс] – Электрон. журн. – URL: https://arxiv.org/abs/1704.04861. (дата обращения: 01.05.2020)

4. Mohamed, Elhassan & Sirlantzis, Konstantinos & Howells, Gareth. Application of Transfer Learning for Object Detection on Manually Collected Data, 2020. [Электронный ресурс] – Электрон. журн. – URL: https://www.researchgate.net/publication/335382472\_Application\_of\_Transfer\_Learning\_for\_Object\_Detection\_on\_Manually\_Collected\_Data. (дата обращения: 08.04.2020)

5. М.О. Корлякова, Д.С. Трушков, Н.В. Лохмачев, Е.Ю. Корлякова. Анализ моделей распознавания образов в системах технического зрения для мобильных автономных платформ. Наукоемкие технологии в приборо- и машиностроении и развитие инновационной деятельности в вузе: материалы Всероссийской научно-технической конференции, 13 – 15 ноября 2018 г. Т. 2. –Калуга: Издательство МГТУ им. Н. Э. Баумана, 2018. – С. 13–17. (дата обращения: 25.05.2020)

6. J. Pedoeem, R. Huang. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers, 2018 [Электронный ресурс] – Электрон. журн. – URL: https://arxiv.org/abs/1811.05588. (дата обращения: 02.05.2020)

7. A. Rosebrock. Deep Learning for Computer Vision with Python, 2017 г. — 210 с. [Электронный ресурс] URL: https://www.pyimagesearch.com/ (дата обращения: 05.05.2020)

8.Pyzo. Free and open-source computing environment based on Python, 2020 г. [Электронный ресурс] URL: https://pyzo.org/ (дата обращения: 03.04.2020)

9. Tensorflow. End-to-end open source platform for machine learning, 2020 г. [Электронный ресурс] URL: https://www.tensorflow.org/ (дата обращения: 05.05.2020)

10. Pytorch. An open source machine learning framework that accelerates the path from research prototyping to production deployment., 2020 г. [Электронный ресурс] URL: https://pytorch.org/ (дата обращения: 05.05.2020)

11. OpenCV. Open source computer vision and machine learning software library, 2020 г. [Электронный ресурс] URL: https://opencv.org/ (дата обращения: 05.05.2020)

# ПРИЛОЖЕНИЕ А Установка NVIDIA CUDA для ускорения работы модели детектирования

Для того, чтобы оправдать подход, включающий в себя вычисления на сервере, на ноутбук MSI также было установлено программное обеспечение NVIDIA CUDA и библиотека tensorflow\_gpu для переноса всех вычислений на видеопроцессор.

Во-первых, пользователю необходимо удостовериться в том, что на его машине установлена видеокарта, поддерживающая вычисления на GPU. Для этого необходимо перейти в диспетчер устройств и открыть вкладку «Видеоадаптеры».

На рисунке А.1 представлено окно диспетчера устройств с выделенной видеокартой.

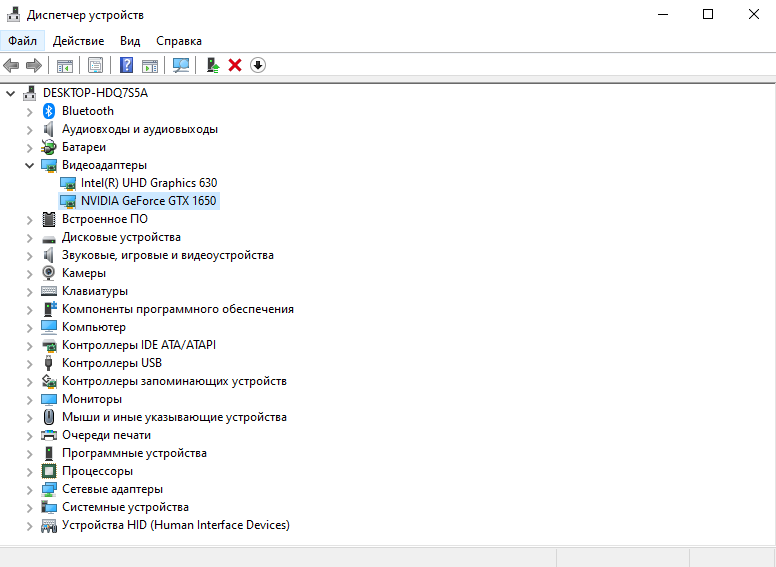


Рисунок А.1 – Окно диспетчера устройств

Далее необходимо перейти открыть каталог устройств, поддерживающих программно-аппаратную архитектуру параллельных вычислений CUDA в Wikipedia, таблица А.1.

Таблица А.1 – Часть таблицы устройств, поддерживающих технологию CUDA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Версия**  **спецификации** | **Архитектура** | **GPU** | **Видеокарты семейства GeForce** | **Видеокарты семейства Quadro, NVS** | **Видеокарты семейства Tesla** |
| 7.5 | Turing | TU102, TU104, TU106 | GeForce RTX 2080 Ti, RTX 2080 Super, ... GTX 1660, GTX 1650 Super, GTX 1650 | Quadro RTX 8000, Quadro RTX 6000, Quadro RTX 5000, Quadro RTX 4000,  Quadro T2000, Quadro T1000 | Tesla T4 |

Как можно видеть из рисунка А.1 и таблицы А.1, GTX 1650 поддерживает CUDA, что означает, что перенос вычислений на графический процессор возможен. Следовательно, можно приступать к установке.

Для начала, необходимо создать новую виртуальную среду для данного проекта, поскольку для каждого проекта требования разные и зависимости могут пересекаться. Для этого необходимо открыть Anaconda, зайти во вкладку Environments и нажать Create. Появится окно, в котором необходимо ввести название среды и выбрать версию Python. В данном случае виртуальная среда будет называться tensorflow\_gpu, a Python будет иметь версию 3.6, рисунок А.2.

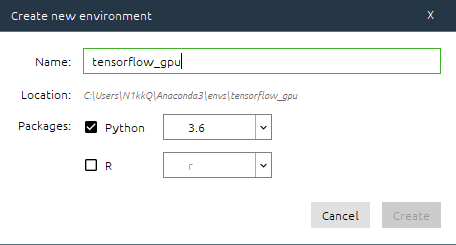


Рисунок А.2 – Создание новой виртуальной среды Anaconda

Теперь необходимо перейти в консоль Anaconda Prompt и активировать данную среду командой conda activate tensorflow\_gpu, рисунок А.3. Теперь можно устанавливать все необходимые библиотеки для проекта.

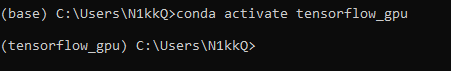


Рисунок А.3 – Активация созданной виртуальной среды

Теперь необходимо установить стандартные библиотеки Python, которые находятся в файле requirements.txt командой pip install -r requirements.txt [3]. Это установит все необходимые для данного проекта библиотеки, включая tensorflow-gpu==1.15, рисунок А.4.



Рисунок А.4 – Установка нужных для проекта библиотек

Далее необходимо установить последние обновления драйверов NVIDIA. Для этого достаточно установить NVIDIA Geforce experience [4] и загрузить последний вышедший драйвер, рисунок А.5.

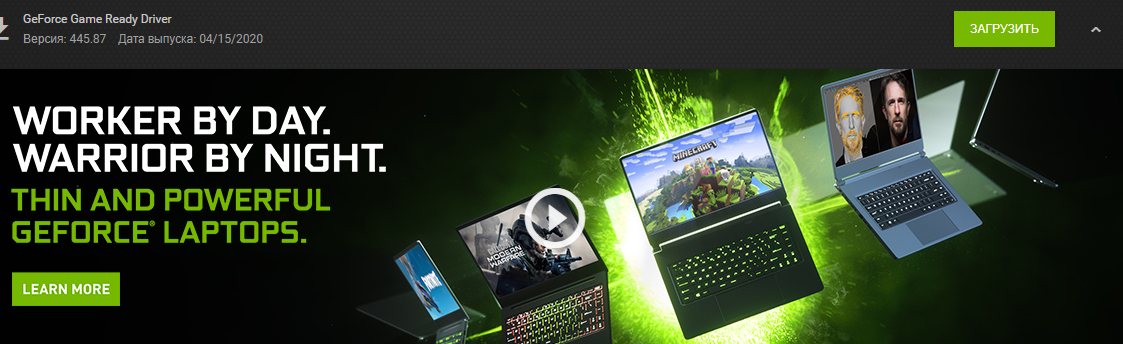


Рисунок А.5 – Обновление драйверов NVIDIA

Теперь нужно скачать CUDA Toolkit, при этом выбрав версию, соответствующую версии спецификации видеокарты (таблица А.1). Например, CUDA SDK 9 поддерживает только версии 3.0 – 7.2, что не подойдет для GTX 1650. Следовательно, необходимо установить именно версию CUDA SDK 10.0 или выше. Скачать нужную версию можно на официальном сайте NVIDIA.

После установки CUDA необходимо поместить некоторые из её папок в переменные среды Windows. Чтобы это сделать, достаточно открыть окно переменных среды и добавить в системную переменную Path следующие пути:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\bin;

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\extras\CUPTI\libx64;

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\include;

Кроме того, требуется установить cuDNN SDK – инструментарий разработки программного обеспечения, включающий в себя библиотеку примитивов для глубоких нейронных сетей. Для этого достаточно скачать данное расширение, создать на системном диске папку “tools/cuda” и поместить все содержимое скачанного архива туда. После этого, необходимо добавить следующий путь в переменные среды:

C:\tools\cuda\bin;

В конечном итоге, в переменной Path должны быть пути, изображенные на рисунке А.6.

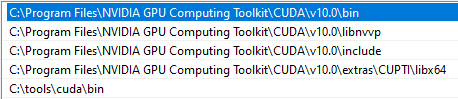


Рисунок А.6 – Вид переменной среды Path

Для проверки правильности установки всех расширений, в консоли достаточно запустить Python командой “python”, импортировать tensorflow командой “import tensorflow as tf” и проверить его версию командой “tf.\_\_version\_\_”. Данный процесс показан на рисунке А.7.

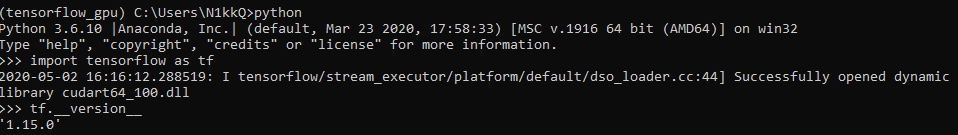


Рисунок А.7 – Проверка библиотеки tensorflow-gpu на работоспособность

Из рисунка А.7 видно, что все настройки произведены верно.

# ПРИЛОЖЕНИЕ Б Настройка Object Detection API

Перед тем, как начать настраивать Object Detection API, нужно удостовериться, что на компьютере установлен Git Bash. Git Bash – набор инструментов для простого взаимодействия с веб-сервисом для хостинга IT-проектов GitHub.

Для начала работы с TensorFlow Object Detection API необходимо

«клонировать» репозиторий tensorflow/models на текущую машину. Для этого нужно воспользоваться командой git clone <ссылка на репозиторий>.

Чтобы Object Detection API работал, необходимо скомпилировать его буферы протоколов. Они служат как бинарная альтернатива текстовому формату XML, но являются более простыми, компактными и быстрыми в связи с тем, что осуществляется передача оптимизированных бинарных данных. Для этого нужно скачать и установить protoc-3.4.0-win32.zip из официального репозитория protocol buffers.

После этого, перейдя в папку models, достаточно выполнить следующую команду в командном окне:

"C:/Program Files/protoc/bin/protoc" object\_detection/protos/\*.proto --python\_out=.

Помимо этого, для того, чтобы в будущем была возможна тонкая настройка, необходимо создать системную переменную среды PYTHONPATH и поместить туда следующие пути к папкам:

<Путь к репозиторию tensorflow models>\models\research\object\_detection;

<Путь к репозиторию tensorflow models>\models\research\slim;

<Путь к репозиторию tensorflow models>\models\research;

Теперь можно приступать к работе с API.

# ПРИЛОЖЕНИЕ В Код Python для формирования набора данных

from data\_utils import xml\_to\_csv, generate\_pbtxt, make\_config

import xmltodict

import imutils

import time

import cv2

import os

PATH\_TO\_DATA = 'dataset'

os.chdir(PATH\_TO\_DATA)

FULL\_PATH = os.getcwd()

template\_name = 'template.xml'

with open(template\_name, "r", encoding='utf-8') as file:

template\_xml = file.read()

obj\_class = str(input())

xml\_d = xmltodict.parse(template\_xml)

tr\_ims\_p = os.listdir(FULL\_PATH + '\\' + 'train\_images')

val\_ims\_p = os.listdir(FULL\_PATH + '\\' + 'valid\_images')

tr\_cur\_obj\_lst = [int(el.replace(obj\_class,'').replace('.jpg','')) for el in tr\_ims\_p if obj\_class in el]

val\_cur\_obj\_lst = [int(el.replace(obj\_class,'').replace('.jpg','')) for el in val\_ims\_p if obj\_class in el]

counter\_t = max(tr\_cur\_obj\_lst)+1 if len(tr\_cur\_obj\_lst) != 0 else 0

counter\_v = max(val\_cur\_obj\_lst)+1 if len(val\_cur\_obj\_lst) != 0 else 0

OPENCV\_OBJECT\_TRACKERS = {

"csrt": cv2.TrackerCSRT\_create,

"kcf": cv2.TrackerKCF\_create,

"boosting": cv2.TrackerBoosting\_create,

"mil": cv2.TrackerMIL\_create,

"tld": cv2.TrackerTLD\_create,

"medianflow": cv2.TrackerMedianFlow\_create,

"mosse": cv2.TrackerMOSSE\_create

}

tracker = OPENCV\_OBJECT\_TRACKERS["csrt"]()

initBB = None

print("[INFO] starting video stream...")

vs = cv2.VideoCapture(0)

time.sleep(1.0)

while True:

ret, frame = vs.read()

frame = frame

if ret is False:

break

frame = imutils.resize(frame, width=500)

(H, W) = frame.shape[:2]

frame\_s = frame.copy()

if initBB is not None:

(success, box) = tracker.update(frame)

if success:

(x, y, w, h) = [int(v) for v in box]

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("s"):

initBB = cv2.selectROI("Frame", frame, fromCenter=False, showCrosshair=True)

tracker.init(frame, initBB)

elif key == ord("t"):

if success:

im\_name = obj\_class + str(counter\_t) + '.jpg'

im\_full\_name = FULL\_PATH + '\\train\_images\\' + im\_name

cv2.imwrite(im\_full\_name, frame\_s)

anot\_dict = xml\_d.copy()

xml\_d["annotation"]["folder"] = 'train\_images'

xml\_d["annotation"]["filename"] = im\_name

xml\_d["annotation"]["path"] = im\_full\_name

xml\_d["annotation"]["size"]["width"] = str(W)

xml\_d["annotation"]["size"]["height"] = str(H)

xml\_d["annotation"]["object"]["name"] = obj\_class

xml\_d["annotation"]["object"]["bndbox"]["xmin"] = str(x)

xml\_d["annotation"]["object"]["bndbox"]["xmax"] = str(x + w)

xml\_d["annotation"]["object"]["bndbox"]["ymin"] = str(y)

xml\_d["annotation"]["object"]["bndbox"]["ymax"] = str(y + h)

im\_anot = xmltodict.unparse(anot\_dict, pretty=True)

im\_anot = im\_anot.split('<?xml version="1.0" encoding="utf-8"?>\n')[1] + '\n'

xml\_name = FULL\_PATH + '\\train\_annot\\' + im\_name[:-4] + '.xml'

with open(xml\_name, "w") as file:

file.write(im\_anot)

print('{} was saved to training folders!'.format(im\_name[:-4]))

counter\_t += 1

elif key == ord("v"):

if success:

im\_name = obj\_class + str(counter\_v) + '.jpg'

im\_full\_name = FULL\_PATH + '\\valid\_images\\' + im\_name

cv2.imwrite(im\_full\_name, frame\_s)

anot\_dict = xml\_d.copy()

xml\_d["annotation"]["folder"] = 'valid\_images'

xml\_d["annotation"]["filename"] = im\_name

xml\_d["annotation"]["path"] = im\_full\_name

xml\_d["annotation"]["size"]["width"] = str(W)

xml\_d["annotation"]["size"]["height"] = str(H)

xml\_d["annotation"]["object"]["name"] = obj\_class

xml\_d["annotation"]["object"]["bndbox"]["xmin"] = str(x)

xml\_d["annotation"]["object"]["bndbox"]["xmax"] = str(x + w)

xml\_d["annotation"]["object"]["bndbox"]["ymin"] = str(y)

xml\_d["annotation"]["object"]["bndbox"]["ymax"] = str(y + h)

im\_anot = xmltodict.unparse(anot\_dict, pretty=True)

im\_anot = im\_anot.split('<?xml version="1.0" encoding="utf-8"?>\n')[1] + '\n'

xml\_name = FULL\_PATH + '\\valid\_annot\\' + im\_name[:-4] + '.xml'

with open(xml\_name, "w") as file:

file.write(im\_anot)

print('{} was saved to validation folders!'.format(im\_name[:-4]))

counter\_v += 1

elif key == ord('c'):

for directory in ['valid\_annot','train\_annot']:

image\_path = os.path.join(os.getcwd(), directory )

xml\_df = xml\_to\_csv.xml\_to\_csv(image\_path)

xml\_df.to\_csv('{}\_labels.csv'.format(directory), index=None)

print('{} was successfully converted xml to csv.'.format(directory))

if directory == 'train\_annot':

with open('model\_label\_map.pbtxt', "w") as file:

file.write(generate\_pbtxt.generate\_pbtxt(xml\_df))

print('pbtxt-file was created!')

os.system("python ..\\data\_utils\\generate\_tfrecord.py --csv\_input=..\\dataset\\train\_annot\_labels.csv --output\_path=..\\dataset\\train.record --image\_dir=..\\dataset\\train\_images")

os.system("python ..\\data\_utils\\generate\_tfrecord.py --csv\_input=..\\dataset\\valid\_annot\_labels.csv --output\_path=..\\dataset\\valid.record --image\_dir=..\\dataset\\valid\_images")

os.chdir('../models')

if 'ssdmn\_fine\_tuned' in os.listdir():

os.system('rmdir ssdmn\_fine\_tuned /s /q')

os.system('mkdir ssdmn\_fine\_tuned')

os.chdir('ssdmn\_fine\_tuned')

os.system('tar xzvf ../ssd\_mobilenet\_v1\_coco\_11\_06\_2017.tar.gz')

os.chdir('..')

os.system('copy ssd\_mobilenet\_v1\_coco.config ssdmn\_fine\_tuned')

config\_path = 'ssdmn\_fine\_tuned/ssd\_mobilenet\_v1\_coco.config'

config\_data = make\_config.make\_config(config\_path, len(xml\_df['class'].unique()))

with open(config\_path, "w") as file:

file.write(config\_data)

os.chdir('..')

if 'training' in os.listdir():

ret = os.system('rmdir training /s /q')

ret = os.system('mkdir training')

ret = os.system('python train.py --logtostderr --train\_dir=training --pipeline\_config\_path=models/ssdmn\_fine\_tuned/ssd\_mobilenet\_v1\_coco.config')

elif key == ord("q"):

break

cv2.destroyAllWindows()

vs.release()

# ПРИЛОЖЕНИЕ Г Код Python для детектирования

import numpy as np

import tensorflow as tf

from tensorflow.compat.v1 import ConfigProto

from tensorflow.compat.v1 import InteractiveSession

import cv2

import time

import os

from utils import label\_map\_util

from utils import visualization\_utils as vis\_util

config = ConfigProto()

config.gpu\_options.allow\_growth = True

session = InteractiveSession(config=config)

os.chdir('training')

last\_model = max([int(el[11:].replace('.index','')) for el in os.listdir() if el[-1] == 'x'])

os.chdir('..')

os.chdir('models/ssdmn\_fine\_tuned')

if 'ssdmn' in os.listdir():

os.system('rmdir ssdmn /s /q')

os.chdir('../..')

infer\_cmd = '''

python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path models/ssdmn\_fine\_tuned/ssd\_mobilenet\_v1\_coco.config --trained\_checkpoint\_prefix training/model.ckpt --output\_directory models/ssdmn\_fine\_tuned/ssdmn

'''.replace('model.ckpt', 'model.ckpt-' + str(last\_model))

os.system(infer\_cmd)

os.chdir('dataset')

os.system('copy model\_label\_map.pbtxt \"../models/ssdmn\_fine\_tuned\" /Y')

os.chdir('..')

#PATH\_TO\_MODEL = 'models/original\_model/'

PATH\_TO\_MODEL = 'models/ssdmn\_fine\_tuned/'

#MODEL\_NAME = PATH\_TO\_MODEL + 'ssd\_mobilenet\_v1\_coco\_11\_06\_2017'

MODEL\_NAME = PATH\_TO\_MODEL + 'ssdmn'

PATH\_TO\_CKPT = MODEL\_NAME + '/frozen\_inference\_graph.pb'

#PATH\_TO\_LABELS = PATH\_TO\_MODEL + 'mscoco\_label\_map.pbtxt'

PATH\_TO\_LABELS = PATH\_TO\_MODEL + 'model\_label\_map.pbtxt'

with open(PATH\_TO\_LABELS, 'r') as f:

data = f.read().split('item')[1:]

NUM\_CLASSES = len(data)

cap = cv2.VideoCapture(0)

detection\_graph = tf.Graph()

with detection\_graph.as\_default():

od\_graph\_def = tf.compat.v1.GraphDef()

with tf.io.gfile.GFile(PATH\_TO\_CKPT, 'rb') as fid:

serialized\_graph = fid.read()

od\_graph\_def.ParseFromString(serialized\_graph)

tf.import\_graph\_def(od\_graph\_def, name='')

label\_map = label\_map\_util.load\_labelmap(PATH\_TO\_LABELS)

categories = label\_map\_util.convert\_label\_map\_to\_categories(label\_map, max\_num\_classes=NUM\_CLASSES, use\_display\_name=True)

category\_index = label\_map\_util.create\_category\_index(categories)

def load\_image\_into\_numpy\_array(image):

(im\_width, im\_height) = image.size

return np.array(image.getdata()).reshape((im\_height, im\_width, 3)).astype(np.uint8)

IMAGE\_SIZE = (12, 8)

with detection\_graph.as\_default():

with tf.compat.v1.Session(graph=detection\_graph) as sess:

while True:

start\_time = time.time()

ret, image\_np = cap.read()

# Expand dimensions since the model expects images to have shape: [1, None, None, 3]

image\_np\_expanded = np.expand\_dims(image\_np, axis=0)

image\_tensor = detection\_graph.get\_tensor\_by\_name('image\_tensor:0')

# Each box represents a part of the image where a particular object was detected.

boxes = detection\_graph.get\_tensor\_by\_name('detection\_boxes:0')

# Each score represent how level of confidence for each of the objects.

# Score is shown on the result image, together with the class label.

scores = detection\_graph.get\_tensor\_by\_name('detection\_scores:0')

classes = detection\_graph.get\_tensor\_by\_name('detection\_classes:0')

num\_detections = detection\_graph.get\_tensor\_by\_name('num\_detections:0')

# Actual detection.

(boxes, scores, classes, num\_detections) = sess.run(

[boxes, scores, classes, num\_detections],

feed\_dict={image\_tensor: image\_np\_expanded})

# Visualization of the results of a detection.

vis\_util.visualize\_boxes\_and\_labels\_on\_image\_array(

image\_np,

np.squeeze(boxes),

np.squeeze(classes).astype(np.int32),

np.squeeze(scores),

category\_index,

use\_normalized\_coordinates=True,

line\_thickness=8)

cv2.imshow('object detection', cv2.resize(image\_np, (800,600)))

print(time.time() - start\_time)

if cv2.waitKey(25) & 0xFF == ord('q'):

cap.release()

cv2.destroyAllWindows()

break

# ПРИЛОЖЕНИЕ Д Код Python для подбора гиперпараметров при помощи генетического алгоритма

from data\_utils import xml\_to\_csv, generate\_pbtxt, make\_config

from tensorboard.backend.event\_processing import event\_accumulator

import matplotlib.pyplot as plt

import numpy as np

import os

from deap import algorithms

from deap import base

from deap import creator

from deap import tools

import random

import array

import math

def model\_train(num\_steps=200000, initial\_learning\_rate=0.004, decay\_factor=0.95, momentum\_optimizer\_value=0.9, decay=0.9, epsilon=1.0):

cur\_dir = os.getcwd()

PATH\_TO\_DATA = 'dataset'

os.chdir(PATH\_TO\_DATA)

for directory in ['valid\_annot','train\_annot']:

image\_path = os.path.join(os.getcwd(), directory )

xml\_df = xml\_to\_csv.xml\_to\_csv(image\_path)

xml\_df.to\_csv('{}\_labels.csv'.format(directory), index=None)

print('{} was successfully converted xml to csv.'.format(directory))

if directory == 'train\_annot':

with open('model\_label\_map.pbtxt', "w") as file:

file.write(generate\_pbtxt.generate\_pbtxt(xml\_df))

print('pbtxt-file was created!')

os.system("python ..\\data\_utils\\generate\_tfrecord.py --csv\_input=..\\dataset\\train\_annot\_labels.csv --output\_path=..\\dataset\\train.record --image\_dir=..\\dataset\\train\_images")

os.system("python ..\\data\_utils\\generate\_tfrecord.py --csv\_input=..\\dataset\\valid\_annot\_labels.csv --output\_path=..\\dataset\\valid.record --image\_dir=..\\dataset\\valid\_images")

os.chdir('../models')

if 'ssdmn\_fine\_tuned' in os.listdir():

os.system('rmdir ssdmn\_fine\_tuned /s /q')

os.system('mkdir ssdmn\_fine\_tuned')

os.chdir('ssdmn\_fine\_tuned')

os.system('tar xzvf ../ssd\_mobilenet\_v1\_coco\_11\_06\_2017.tar.gz')

os.chdir('..')

os.system('copy ssd\_mobilenet\_v1\_coco.config ssdmn\_fine\_tuned')

config\_path = 'ssdmn\_fine\_tuned/ssd\_mobilenet\_v1\_coco.config'

config\_data = make\_config.make\_config(config\_path, len(xml\_df['class'].unique()),num\_steps=num\_steps, initial\_learning\_rate=initial\_learning\_rate, decay\_factor=decay\_factor, momentum\_optimizer\_value=momentum\_optimizer\_value, decay=decay, epsilon=epsilon)

with open(config\_path, "w") as file:

file.write(config\_data)

os.chdir('..')

if 'genetic\_algorithm\_training' in os.listdir():

os.system('rmdir genetic\_algorithm\_training /s /q')

os.system('mkdir genetic\_algorithm\_training')

os.system('python train.py --logtostderr --train\_dir=genetic\_algorithm\_training --pipeline\_config\_path=models/ssdmn\_fine\_tuned/ssd\_mobilenet\_v1\_coco.config')

os.chdir('genetic\_algorithm\_training')

ga\_dir\_lst = os.listdir()

events\_name = list(filter(lambda x: 'events' in x, ga\_dir\_lst))[0]

ea = event\_accumulator.EventAccumulator(events\_name,

size\_guidance={ # see below regarding this argument

event\_accumulator.COMPRESSED\_HISTOGRAMS: 500,

event\_accumulator.IMAGES: 4,

event\_accumulator.AUDIO: 4,

event\_accumulator.SCALARS: 0,

event\_accumulator.HISTOGRAMS: 1,

})

ea.Reload()

data = ea.Scalars('Losses/TotalLoss')

steps = [el[1] for el in data][1::]

values = np.array([el[2] for el in data][1::])

os.chdir(cur\_dir)

return steps, values

def fitness\_function(ind):

if (ind[0]>1.0) or (ind[1]>1.0) or (ind[2]>1.0) or (ind[3]>1.0) or (ind[4]>1.0):

return 999999,

else:

\_, values = model\_train(num\_steps=400, initial\_learning\_rate=ind[0], decay\_factor=ind[1], momentum\_optimizer\_value=ind[2], decay=ind[3], epsilon=ind[4])

print(values)

if values==[]:

return 999999,

else:

return values[-1],

def choice\_based(ind, indpb):

new\_ind = toolbox.clone(ind)

for i in range(0, len(ind)):

if np.random.uniform() < indpb:

if i==0:

new\_ind[i] = np.random.uniform(LR\_START, LR\_END)

if i==1:

new\_ind[i] = np.random.uniform(DF\_START, DF\_END)

if i==2:

new\_ind[i] = np.random.uniform(MOV\_START, MOV\_END)

if i==3:

new\_ind[i] = np.random.uniform(D\_START, D\_END)

if i==4:

new\_ind[i] = np.random.uniform(E\_START, E\_END)

return new\_ind,

IND\_SIZE = 5

LR\_START = 0.00001

LR\_END = 0.15

DF\_START = 0.7

DF\_END = 0.99

MOV\_START = 0.6

MOV\_END = 0.99

D\_START = 0.7

D\_END = 0.99

E\_START = 0.5

E\_END = 1.0

N\_POP = 8

NGEN = 5

CXPB = 0.8

MUTPB = 0.05

creator.create("FitnessMin", base.Fitness, weights=(-1.0,))

creator.create("Individual", list, typecode='d', fitness=creator.FitnessMin)

toolbox = base.Toolbox()

toolbox.register("LR", np.random.uniform, LR\_START, LR\_END)

toolbox.register("DF", np.random.uniform, DF\_START, DF\_END)

toolbox.register("MOV", np.random.uniform, MOV\_START, MOV\_END)

toolbox.register("D", np.random.uniform, D\_START, D\_END)

toolbox.register("E", np.random.uniform, E\_START, E\_END)

toolbox.register("individual", tools.initCycle, creator.Individual,

(toolbox.LR, toolbox.DF, toolbox.MOV, toolbox.D, toolbox.E), n=1)

toolbox.register("population", tools.initRepeat, list, toolbox.individual)

toolbox.register("mate", tools.cxBlend, alpha=0.2)

toolbox.register("mutate", choice\_based, indpb=0.9)

toolbox.register("select", tools.selRoulette)

toolbox.register("evaluate", fitness\_function)

pop = toolbox.population(n=N\_POP)

hof = tools.HallOfFame(1)

stats = tools.Statistics(lambda ind: ind.fitness.values)

stats.register("min", np.min)

pop, log = algorithms.eaSimple(pop, toolbox, cxpb=CXPB, mutpb=MUTPB, ngen=NGEN, stats=stats, halloffame=hof, verbose=True)

'''

all\_graphs\_values = []

for i in range(0,5):

steps, values = model\_train(num\_steps=2000)

all\_graphs\_values.append(values)

all\_graphs\_values = [el[0:4] for el in all\_graphs\_values]

mean\_values = np.mean(all\_graphs\_values, axis=0)

std\_values = np.std(all\_graphs\_values, axis=0)

y\_up = mean\_values + std\_values

y\_down = mean\_values - std\_values

plt.plot(steps, mean\_values, 'r', steps, y\_up, 'c', steps, y\_down, 'c')

'''

# ПРИЛОЖЕНИЕ Е Код Python для обучения нейронной сети на наборе данных Fashion MNIST и её тестирования

#Подключаем библиотеки

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

fashion\_mnist = keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',

'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

plt.figure(figsize=(10,10))

for i in range(25):

plt.subplot(5,5,i+1)

plt.xticks([])

plt.yticks([])

plt.grid(False)

plt.imshow(train\_images[i], cmap=plt.cm.binary)

plt.xlabel(class\_names[train\_labels[i]])

model = keras.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(128, activation=tf.nn.relu),

keras.layers.Dense(10, activation=tf.nn.softmax)

])

model.compile(optimizer=tf.train.AdamOptimizer(),

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

model.fit(train\_images, train\_labels, epochs=5, run\_id='clothes-classifier')

model.save("clothes-classifier.tfl")

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)

predictions = model.predict(test\_images)

def plot\_image(i, predictions\_array, true\_label, img):

predictions\_array, true\_label, img = predictions\_array[i], true\_label[i], img[i]

plt.grid(False)

plt.xticks([])

plt.yticks([])

plt.imshow(img, cmap=plt.cm.binary)

predicted\_label = np.argmax(predictions\_array)

if predicted\_label == true\_label:

color = 'blue'

else:

color = 'red'

plt.xlabel("{} {:2.0f}% ({})".format(class\_names[predicted\_label],

100\*np.max(predictions\_array),

class\_names[true\_label]),

color=color)

def plot\_value\_array(i, predictions\_array, true\_label):

predictions\_array, true\_label = predictions\_array[i], true\_label[i]

plt.grid(False)

plt.xticks([])

plt.yticks([])

thisplot = plt.bar(range(10), predictions\_array, color="#777777")

plt.ylim([0, 1])

predicted\_label = np.argmax(predictions\_array)

thisplot[predicted\_label].set\_color('red')

thisplot[true\_label].set\_color('blue')

num\_rows = 5

num\_cols = 3

num\_images = num\_rows\*num\_cols

plt.figure(figsize=(2\*2\*num\_cols, 2\*num\_rows))

for i in range(num\_images):

plt.subplot(num\_rows, 2\*num\_cols, 2\*i+1)

plot\_image(i, predictions, test\_labels, test\_images)

plt.subplot(num\_rows, 2\*num\_cols, 2\*i+2)

plot\_value\_array(i, predictions, test\_labels)

# Возьмём изображение из тестового набора данных

img = test\_images[0]

#Добавим изображение в пакет, где он является единственным членом

img = (np.expand\_dims (img, 0))

predictions\_single = model.predict(img)

print(predictions\_single)

plot\_value\_array(0, predictions\_single, test\_labels)

\_ = plt.xticks(range(10), class\_names, rotation=45)

np.argmax(predictions\_single[0])

# ПРИЛОЖЕНИЕ Ж Код MatLab готовой обученной модели классификтора Fashion MNIST, сгенерированный MatLab

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)

%MYNEURALNETWORKFUNCTION neural network simulation function.

%

% Auto-generated by MATLAB, 22-May-2020 13:10:37.

%

% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:

%

% X = 1xTS cell, 1 inputs over TS timesteps

% Each X{1,ts} = Qx784 matrix, input #1 at timestep ts.

%

% and returns:

% Y = 1xTS cell of 1 outputs over TS timesteps.

% Each Y{1,ts} = Qx10 matrix, output #1 at timestep ts.

%

% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<\*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1

x1\_step1.xoffset = ...

x1\_step1.gain = ...

x1\_step1.ymin = -1;

% Layer 1

b1 = ...

IW1\_1 = ...

% Layer 2

b2 = ...

LW2\_1 = ...

% ===== SIMULATION ========

% Format Input Arguments

isCellX = iscell(X);

if ~isCellX

X = {X};

end

% Dimensions

TS = size(X,2); % timesteps

if ~isempty(X)

Q = size(X{1},1); % samples/series

else

Q = 0;

end

% Allocate Outputs

Y = cell(1,TS);

% Time loop

for ts=1:TS

% Input 1

X{1,ts} = X{1,ts}';

Xp1 = mapminmax\_apply(X{1,ts},x1\_step1);

% Layer 1

a1 = tansig\_apply(repmat(b1,1,Q) + IW1\_1\*Xp1);

% Layer 2

a2 = softmax\_apply(repmat(b2,1,Q) + LW2\_1\*a1);

% Output 1

Y{1,ts} = a2;

Y{1,ts} = Y{1,ts}';

end

% Final Delay States

Xf = cell(1,0);

Af = cell(2,0);

% Format Output Arguments

if ~isCellX

Y = cell2mat(Y);

end

end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function

function y = mapminmax\_apply(x,settings)

y = bsxfun(@minus,x,settings.xoffset);

y = bsxfun(@times,y,settings.gain);

y = bsxfun(@plus,y,settings.ymin);

end

% Competitive Soft Transfer Function

function a = softmax\_apply(n,~)

if isa(n,'gpuArray')

a = iSoftmaxApplyGPU(n);

else

a = iSoftmaxApplyCPU(n);

end

end

function a = iSoftmaxApplyCPU(n)

nmax = max(n,[],1);

n = bsxfun(@minus,n,nmax);

numerator = exp(n);

denominator = sum(numerator,1);

denominator(denominator == 0) = 1;

a = bsxfun(@rdivide,numerator,denominator);

end

function a = iSoftmaxApplyGPU(n)

nmax = max(n,[],1);

numerator = arrayfun(@iSoftmaxApplyGPUHelper1,n,nmax);

denominator = sum(numerator,1);

a = arrayfun(@iSoftmaxApplyGPUHelper2,numerator,denominator);

end

function numerator = iSoftmaxApplyGPUHelper1(n,nmax)

numerator = exp(n - nmax);

end

function a = iSoftmaxApplyGPUHelper2(numerator,denominator)

if (denominator == 0)

a = numerator;

else

a = numerator ./ denominator;

end

end

% Sigmoid Symmetric Transfer Function

function a = tansig\_apply(n,~)

a = 2 ./ (1 + exp(-2\*n)) - 1;

end