Федеральное государственное автономное образовательное учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

**Факультет информатики, математики и компьютерных наук**

**КУРСОВАЯ РАБОТА**

Алгоритмы машинного обучения и их применение.

по направлению подготовки 09.03.04 "Программная инженерия"

образовательная программа «Программная инженерия»

Выполнил:

Студент группы 18ПИ-1

Яхтин Леонид Александрович

Руководитель:

Шимко Алексей Андреевич,

преподаватель

Нижний Новгород

**Содержание**

[**1.** **Введение** 3](#_Toc42555824)

[**2.** **Постановка задачи** 4](#_Toc42555825)

[**2.1.** **Описание задачи** 4](#_Toc42555826)

[**2.2.** **Входные/Выходные данные** 4](#_Toc42555827)

[**2.3.** **Пример работы программы** 4](#_Toc42555828)

[**3.** **Обзор существующих решений.** 6](#_Toc42555829)

[**3.1.** **Стандартные методы машинного обучения.** 6](#_Toc42555830)

[**3.2.** **Свёрточная нейронная сеть с 2 выходами.** 6](#_Toc42555831)

[**3.3.** **Сети для Object detection.** 7](#_Toc42555832)

[**4.** **Описание выбранного решения** 9](#_Toc42555833)

[**4.1.** **Выбор данных для обучения** 9](#_Toc42555834)

[**4.2.** **Предобработка/аугментация данных** 9](#_Toc42555835)

[**4.3.** **Выбор архитектуры сети** 10](#_Toc42555836)

[**4.4.** **Обучение модели** 13](#_Toc42555837)

[**4.5.** **Борьба с переобучением** 14](#_Toc42555838)

[**4.6.** **Процесс обучения** 16](#_Toc42555839)

[**4.7** **Демонстрационное приложение** 16](#_Toc42555840)

[**5.** **Результаты** 17](#_Toc42555841)

[**6.** **Дальнейшее развитие** 18](#_Toc42555842)

[**7.** **Список литературы** 19](#_Toc42555843)

[8. **Приложение** 20](#_Toc42555844)

# **Введение**

Машинное обучение — это обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. Различают два типа обучения. Обучение по прецедентам, или индуктивное обучение, основано на выявлении общих закономерностей по частным эмпирическим данным. Особую популярность с развитием вычислительных мощностей, особенно графических ускорителей, получили нейронные сети, способные решать широкий круг задач, особенно при анализе и обработке изображений.

Одной из задач, где активно применяются нейронные сети является задача локализации и классификации объектов на изображении. Задачей классификации является задача определения класса объекта по некоторому набору признаков. Локализация – это задача определения положения объекта на изображении. Довольно часто при локализации определяется ограничивающая рамка (далее bounding box) объекта. Однако, зачастую задача локализации + классификации заменяется задачей детектирования объектов (object detection). Решение задачи детектирования объектов позволяет определять классы и положения нескольких объектов (в то время как задачи локализации и классификации применимы для одного объекта на изображении). Тем не менее, в ситуациях, когда важна скорость получения результата и на изображении необходимо обнаружить только один объект, локализация + классификация является более подходящим выбором. Поэтому, задача локализации и классификации изображений актуальна и сейчас, и в данной работе будет рассмотрена задача локализации к классификации персонажей мультсериала “The Simpsons”.

# **Постановка задачи**

## **Описание задачи**

Спроектировать и обучить модель, используя библиотеку глубокого обучения Tensorflow 2, определяющую персонажа из мультсериала “The Simpsons” на изображении и обводящую его прямоугольником (далее – bounding box). Основными критериями оценки работы модели будет точность определения персонажей и обрамляющих прямоугольников, а также ее быстродействие.

## **Входные/Выходные данные**

* Входные данные для обучения модели:

Текстовый файл в формате:

Путь до изображения,x1,y1,x2,y2,имя персонажа

Где x1, y1 – координаты верхнего левого угла bounding box,

x2, y2 – координаты нижнего правого угла bounding box

* Выходные данные для обучения модели:

Csv файл со значениями функций потерь на обучающей и тестовой выборках и learning rate

h5 файл – сохраненная модель

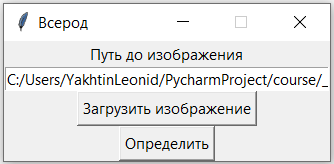
* Входные данные для приложений/обученной модели:

На вход принимается путь до изображения (на нем находится один из персонажей “The Simpsons”), который вводится в соответствующее поле приложения или выбирается из соответствующего окна выбора.

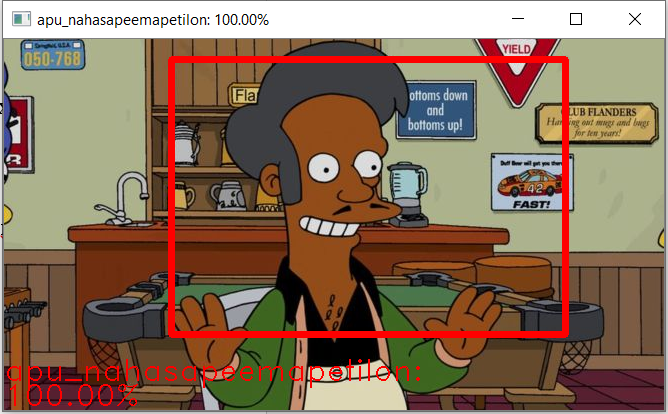
* Выходные данные для приложений/обученной модели:

Изображение с именем найденного персонажа и его bounding box.

## **Пример работы программы**

****

*Изображение №1 – Интерфейс приложения*

****

*Изображение №2 – Результат работы приложения*

****

*Изображение №3 – результат после обучения модели*

# **Обзор существующих решений.**

Существует несколько подходов к решению данной задачи:

## **Стандартные методы машинного обучения.**

Решить две отдельные задачи – множественной классификации для определения имен персонажей и регрессии для определения границ bounding box.

Для множественной классификации можно использовать стратегию One-vs-all или All-vs-all и несколько (равное числу классов или квадрату числа классов соответственно) линейных классификаторов или решающих деревьев.

Однако, у этого подхода есть несколько проблем. Во-первых, потребуется использовать большое количество моделей. Во-вторых, что куда важнее, линейные модели и решающие деревья, как правило, плохо работают с изображениями, так как в силу своей природы эти модели не учитывают связь между соседними пикселями, рассматривая каждый как независимое свойство, из-за чего данные модели очень чувствительны, например, к положению объектов в разных частях изображений (то есть модель обладает плохой обобщающей способностью).

## **Свёрточная нейронная сеть с 2 выходами.**

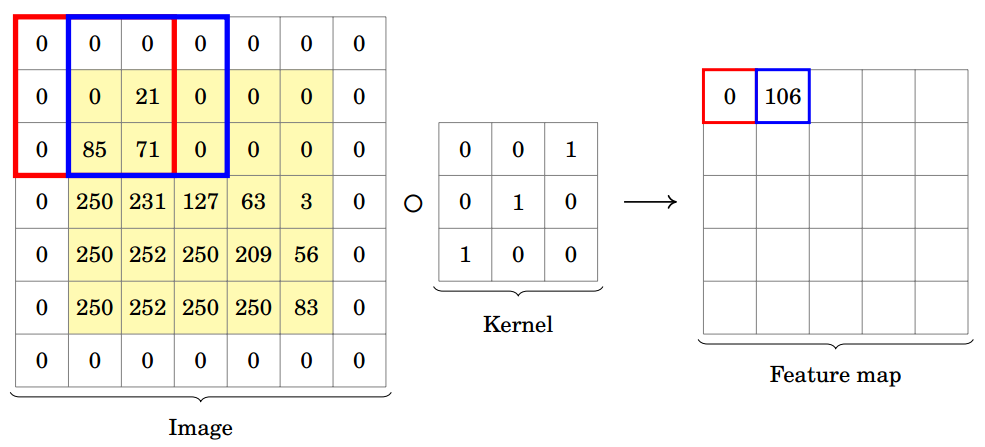
Свёрточные нейронные сети являются стандартным подходом при решении задач над изображениями. Отличительной особенностью архитектуры этих моделей является применение операции свёртки, для которой используется матрица весов (называемая ядром свертки), как правило, небольших размеров. Результатом применения сверточного слоя является выходное изображение, состоящее из одного или нескольких каналов, где пиксель каждого канала является суммой константы и элементов матрицы, полученной в результате вычисления произведения Адамара для ядра свертки и матрицы, последовательно и с определенным сдвигом выделенной из исходного изображения. Стоит отметить, что для вычисления всех пикселей, принадлежащих одному каналу результирующего изображения, используется одна и та же матрица. Таким образом, свёрточные сети обладают значительно меньшим числом “обучаемых” параметров, в сравнении с нейронными сетями с прямой связью.

Также в свёрточных нейронных сетях применятся слои субдискретизации (pooling layers), осуществляющих уплотнение группы пикселей (как правило, размера 2 на 2) до 1 пикселя, являющегося максимумом/минимумом/суммой/средним арифметическим группы пикселей. Это позволяет решить несколько задач:

* Уменьшение размеров изображения, что способствует ускорению обучения;
* Фильтрация ненужных/избыточных признаков, что способствует уменьшению переобучения;
* Преобладание факта наличия определенного признака над его положением в изображении, что повышает инвариантность нейронной сети к масштабу изображения и положению объектов на нем;

Только после чередования свёрточных слоев и слоев субдискретизации полученное результирующее изображение выпрямляется в вектор свойств, который подается на вход полносвязным слоям.

Таким образом, свёрточные нейронные сети позволяют решить ряд проблем (таких, как плохая обобщающая способность относительно положения объектов на изображении и склонность и переобучению из-за большого числа обучаемых параметров), свойственных нейронным сетям, состоящим только из полносвязных слоев, что делает их более предпочтительными при работе с изображениями.

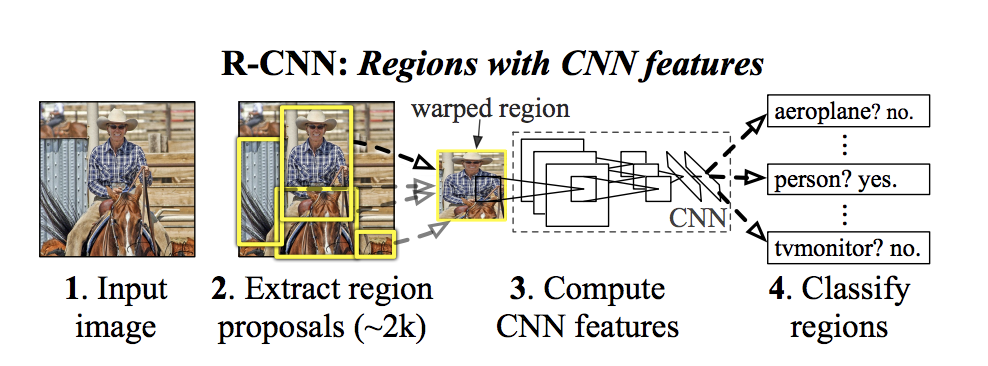
 

*Изображение №4 – демонстрация применения свёрточного слоя и слоя субдискретизации*

## **Сети для Object detection.**

Данные нейронные сети позволяют определять и выделять в bounding boxes сразу несколько объектов на 1 изображении. К ним можно отнести R-CNN, Fast-RCNN, Faster-RCNN, SSD, YOLO, Retina-Net и так далее.

Их основная идея заключается в выделении областей, где могут находиться объекты, и уже определение класса объекта (или его отсутствия) и bounding box этого объекта в выделенной области.



*Изображение №5 – структура R-CNN сети.*

Так автор датасета, который был использован для обучения модели данной работы, использовал Faster R-CNN. Однако отмечается несколько недостатков этого решения. Одним из них является скорость работы, так для обработки 1 изображения требовалось 8 секунд на CPU или 0.98 секунды на GPU (Tesla k80). Также упоминается о снижении точности предсказания нескольких персонажей, слишком больших и пересекающихся bounding boxes. Кроме того, нельзя не отметить серьезные затраты на обучение модели (автору датасета потребовалось 10 часов для обучения при использовании GPU Tesla k80). Именно из-за совокупности этих факторов было принято решение отказаться от модели object detection.

# **Описание выбранного решения**

## **Выбор данных для обучения**

Для обучения модели был выбран датасет “The Simpsons Characters Data. Image dataset of 20 characters from The Simpsons”, представленный на сайте kaggle.com и состоящий из 2 частей:

Данные для простой классификации персонажей на изображении

Размеченные данные для классификации определения положения (координаты bounding boxes) персонажей (около 6.5 тысяч изображений с 18 персонажами). Именно вторая часть использовалась для обучения.

## **Предобработка/аугментация данных**

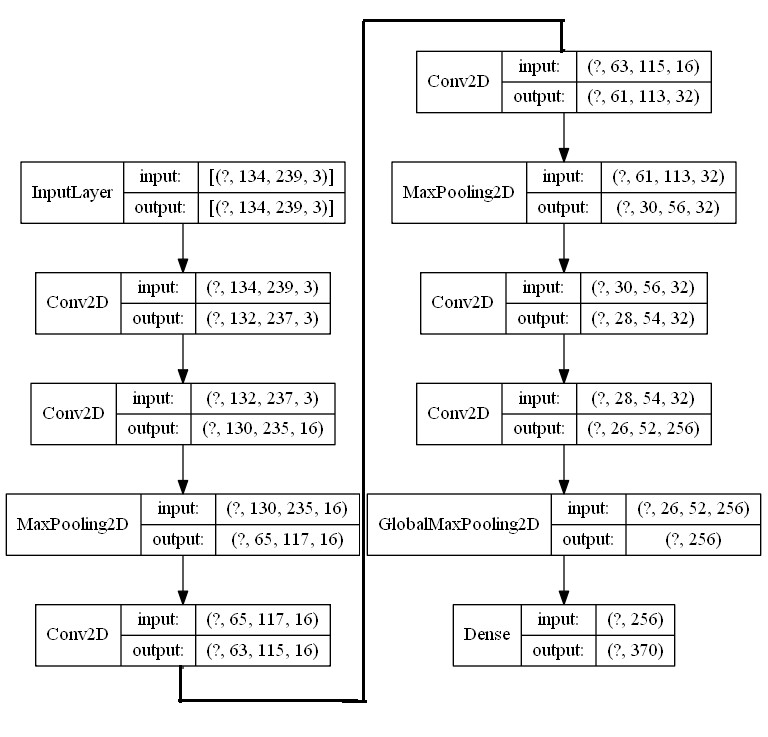
В качестве предобработки данных было принято несколько действий. Во-первых, все изображения в датасете имели разные размеры. Для решения этой проблемы были определены максимальные высота и ширина изображений (1072 на 1912) и пропорционально уменьшены в 8 раз (до 134 на 239), а замер был изменен размер всех изображений на полученный и изменены границы bounding boxes (коэффициент подбирался экспериментально таким образом, что изображение становилось достаточно небольшими, от чего зависит объем требуемой памяти для загрузки датасета, при этом персонажи оставались хорошо различимыми). Также у изображений изменялась яркость в пределах от -0,3 до 0,3. Уже в результате обучения выявилась особенность датасета – большинство персонажей было смешено к левой верхней границе, из-за чего нейронная сеть почти всегда определяла bounding box в одной области в независимости от расположения персонажа. Для борьбы с этим были добавлены отраженные версии изображений. Также тестировалось изменение контрастности изображения, но это значительно ухудшало метрики качества.

Значения каждого цвета пикселя были нормализованы (приводились к [0, 1] путем деления на 255).

Для кодирования классов персонажей применяется one hot encoding (получается одномерный вектор с длинной, равной количеству классов, со значениями 0 и 1 (классы совпадают и не совпадают соответственно)). Полученный датасет разделяется на 2 выборки: train – 85%, test – 15%.

## **Выбор архитектуры сети**

Для решения выбранной задачи было принято решение использовать сверточную нейронную сеть с 2 выходами - полносвязными слоями. Сверточные сети являются стандартным методом обработки изображений, так как они позволяют рассматривать не обособленно, а в совокупности в окружающими его пикселями, из-за чего не теряется целостность данных. Для подбора оптимальной архитектуры сверточной части сети сначала решалась только задача классификации.



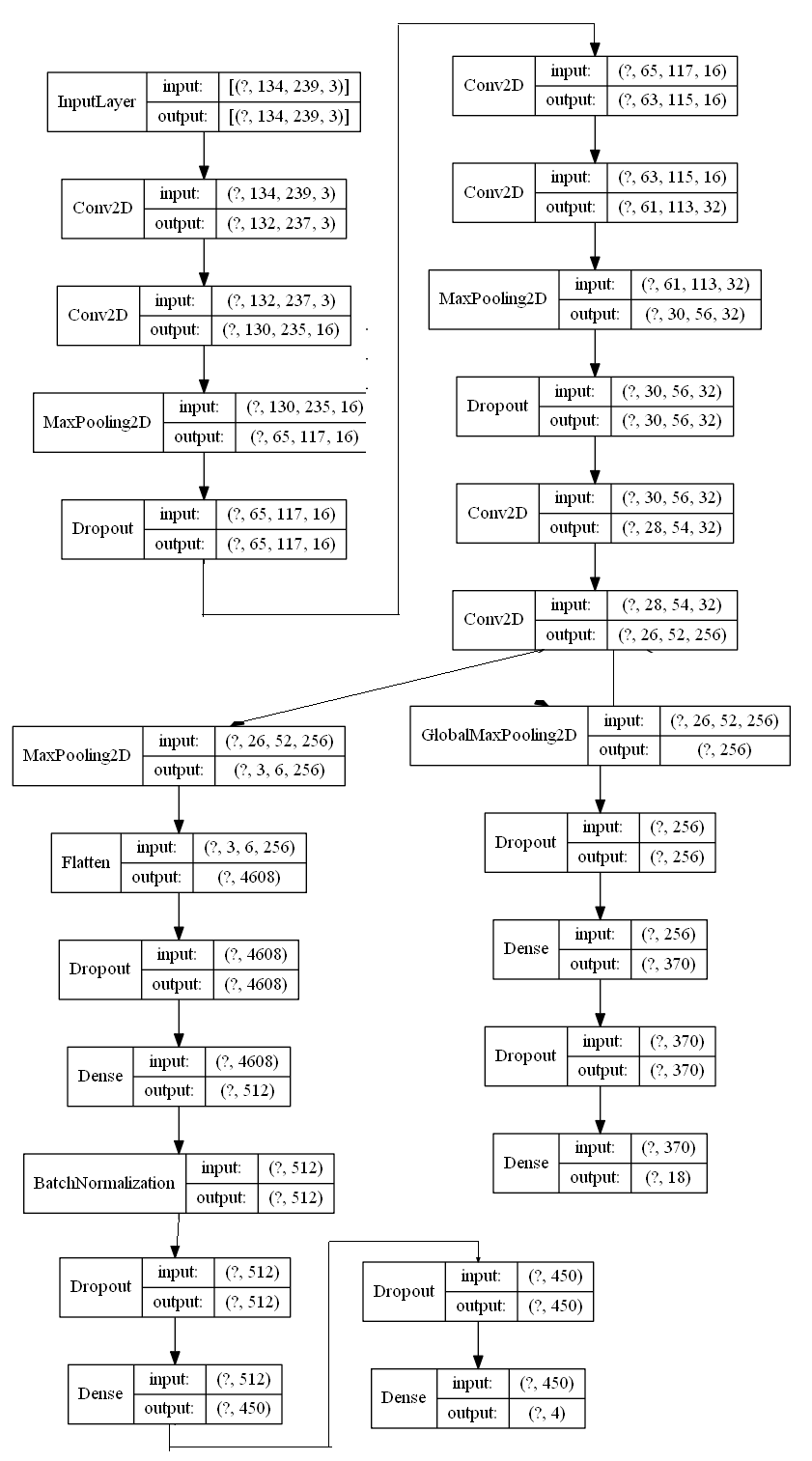
*Изображение №6 – тестовая модель классификации.*

В результате была полученная модель, состоящая из чередования 3 пар сверточных слоев (с ядром 3x3, шагом и отступом 1) и MaxPooling слоями (с ядром 2x2). В сети используется комбинация 2 сверток с ядром 3x3 вместо 1 5x5, так как это позволяет сохранить размер области, обхватываемой 1 “пикселем”, но значительно уменьшить число обучаемых параметров. Также для еще большего уменьшения числа обучаемых параметров тестировалась комбинация сверток с ядрами 1x3 и 3x1 вместо 3x3, но это значительно снижало точность и не уменьшало переобучение модели.

Для решения задачи локализации модели был добавлен дополнительный выход.

Для классификации к каналам, полученным после сверток применяется GlobalMaxPooling, в результате чего получается вектор свойств размера 256. Полученные значения проходят через 2 полносвязных слоя (размерами 370 и 18 (для определения классов)). Эксперименты показали, что увеличение числа слоев не увеличивает точность и способствует переобучению модели. Для локализации применяется MaxPooling с размером ядра 8x8 и результат преобразуется в одномерный вектор (здесь MaxPooling применяется вместо GlobalMaxPooling для получения вектора свойств большего размера, что помогает увеличить точность определения bounding box), к которому применяется 3 полносвязных слоя (размерами 512, 450 и 4 (предсказания координат bounding box) соответственно).

Для выхода каждого сверточного слоя и полносвязных слоев, кроме последнего слоя классификации, где применяется функция активации Softmax, применяется ReLu или ELU функция активации. Использование ELU в среднем показывает более стабильно высокие результаты на этапе обучения, но в итоговой модели используется ReLU, так как при нем была максимальная суммарная “точность” предсказаний.



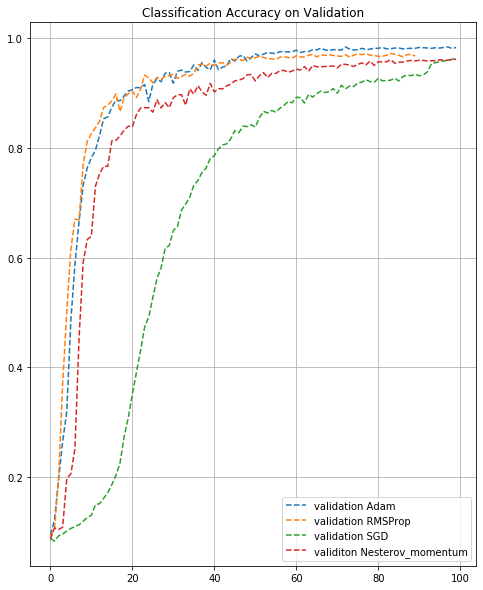
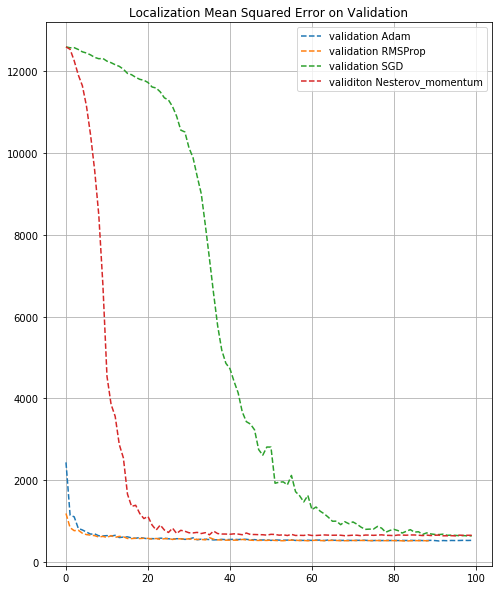
*Изображение №7 – структура итоговой модели.*

## **Обучение модели**

В качестве функций потерь были использованы log(MSE) для вывода, отвечающего за локализацию, а для выхода, отвечающего за классификацию применялась категориальная кросс энтропия. Для локализации использовался log(MSE), а не MSE, так как MSE и категориальная кросс энтропия выдавали ошибки разного порядка, из-за чего классификатор почти бы не обучался.

Итоговая функция потерь была равна:

В качестве оптимизационного алгоритма были протестированы SGD, SGD momentum, Nesterov momentum, RMSProp и Adam. Наилучший результат показал Adam, так как при нем достигалось минимальное значение функции потерь. RMSProp показал близкие к Adam результаты, а худший результат показал SGD, так как даже при увеличении для него числа эпох в почти в 2 раза результат отставал от результатов других оптимизаторов.

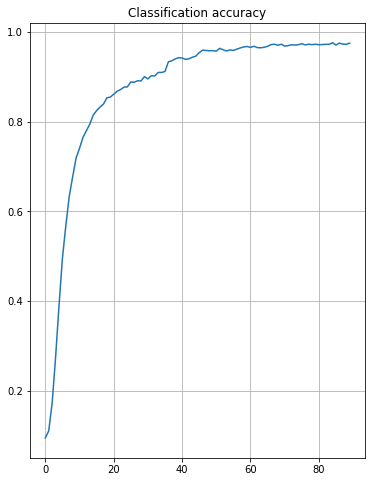
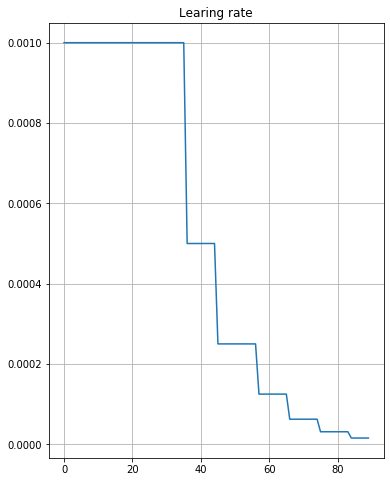
 

*Изображение №8 – графики метрик для разных алгоритмов оптимизации*

В качестве метрик для наглядной оценки моделей использовались точность классификации и средняя квадратичная ошибка.

Модели обучались от 80 до 140 эпох, при этом эксперименты показали, что обучение более чем 100 эпох не улучшало метрики. Для каждой архитектуры сети обучение повторялось от 3 до 5 раз, из которых выбиралась модель с наилучшими метриками, чтобы уменьшить возможность неудачного обучения из-за случайности.

Для того, чтобы модель продолжала улучшать значения метрик на протяжении всего периода обучения применялось уменьшение learning rate при выходе функции потерь на плато. Это реализовывалось callback-ом, уменьшающим learning rate в 2 раза при изменении validation loss менее чем на 0.01 на протяжении 5 эпох с cooldown в 5 эпох. Результативность замедления темпа обучения для улучшения качества метрик демонстрируется скачками увеличения точность классификации при уменьшении learning rate (особенно наглядно при первом уменьшении learning rate).



*Изображение №9 –изменение learning rate и classfication accuracy для RMSProp*

## **Борьба с переобучением**

Переобучение было одной из самых сложных проблем для данного датасета, так как дасет состоял из примерно 13 тысяч картинок, что относительно

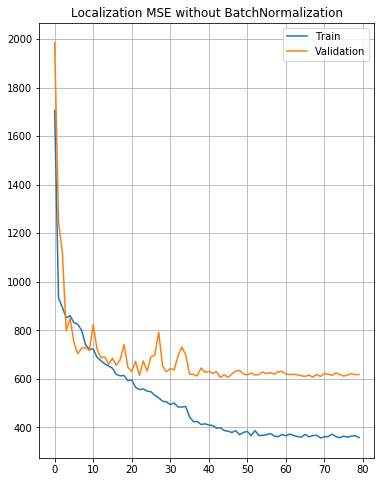
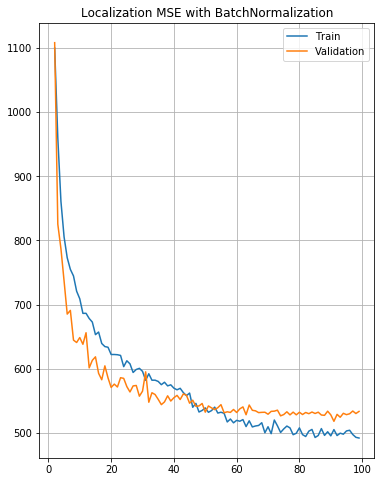
немного.

Для решения проблемы переобучения использовались следующие методы:

* Dropout – метод, основанный на исключение некоторых нейронов с вероятностью p на стадии обучения. Так как нейрон исключается из сети и не участвует при вычислении функции потерь и распространении ошибки, то исключение нейронов равносильно созданию новой нейронной сети.

Для обучения использовалось 3 различных вероятности dropout:

* p = 0.3 – применяется между группами слоев conv2d + conv2d + maxpooling
* p = 0.5 – применяется между полносвязными слоями (особенно полезно при слоях больших размеров)
* p = 0.15 – применяется перед последним слоем для итогового предсказания координат и перед первым полносвязным слоем классификации.
* BatchNormalization – приводит выход слоя к стандартному распределению, что ускоряет обучение, и в данной сети помогает уменьшить переобучение и MSE модели для выхода локализации.



*Изображение №10 – графики значений MSE с и без BatchNormalization.*

* Регуляризация – введение штрафа, зависящего от значений весов. Тестировалась L2 (сумма квадратов весов слоя) и L1 (сумма абсолютных значений весов слоя) нормализации со значениями (0.01, 0.005, 0.001, 0.0005) на полносвязных слоев, однако это приводило только замедлению обучения, ухудшению метрик точности (особенно MSE), при этом переобучение модели не уменьшалось.

## **Процесс обучения**

Для обучения, преимущественно, использовалось GPU NVidia Geforce RTX 2060 (6Gb), а также NVidia GeForce GTX 1050 Ti. Процесс обучения занимал 20-25 минут (15 секунд на эпоху) и 32-40 минут (24 секунды на эпоху) для видеокарт соответственно.

## **Демонстрационное приложение**

Для оценки времени, требуемого для предсказания персонажа и его bounding box, и удобного применения нейронной сети использовалось простое GUI приложение, реализованная с использованием tkinter. Приложение загружает обученную модель (важно загружать модель без ее компиляции, так как при обучении использовалась пользовательская функция потерь). Каждое поступающее изображение преобразуется в необходимый для модели формат (масштабируется до размеров 134 на 239 пикселей, значения каждого цветового канала пикселя нормализуется), добавляется пустая размерность, передается на вход загруженной модели, выход локализации модели масштабируется для первоначальных размеров изображения, после чего отображается изображение в bounding box, именем персонажа и его вероятностью.

# **Результаты**

В результате работы были рассмотрены различные методы локализации и классификации объектов на изображении, получен опыт использования таких библиотек, как tensorflow 2 и opencv. Была разработана и обучена модель, определяющая 18 персонажей (и их положение) из мультсериала “The Simpsons”, а также подготовлено простое демонстрационное приложение с графическим пользовательским интерфейсом использования модели на практике. Полученная модель продемонстрировала точность предсказания персонажа 98.21% и среднюю квадратичную ошибку 532 на тестовых данных. Также была выполнена одна их ключевых поставленных задач – скорость работы модели: на само предсказание требуется от 0.03 до 0.1 секунды на CPU, что значительно превосходит скорость работы решения автора датасета.

# **Дальнейшее развитие**

Одним из возможных направлений развития полученной модели является улучшение точности предсказания bounding box персонажей, так как иногда модель выделяет слишком большой bounding box, особенно когда на изображении присутствует несколько персонажей.

Еще одним направлением развития является увеличения количества предсказываемых персонажей (классов классификации). Это потребует значительного увеличения датасета.

# **Список литературы**

1. Introduction to Deep Learning – онлайн курс (проверено 26.05.2020) [<https://www.coursera.org/learn/intro-to-deep-learning/>]
2. CS231n: Convolutional Neural Networks for Visual Recognition Spring 2017 – цикл лекций (проверено 26.05.2020) [<http://cs231n.stanford.edu/2017/>]
3. Нейронные сети и компьютерное зрение – онлайн курс (проверено 26.05.2020) [<https://stepik.org/course/50352/syllabus>]
4. The Simpsons Characters Data – датасет для обучения и статья с писанием решения автора датасета [<https://www.kaggle.com/alexattia/the-simpsons-characters-dataset>]
5. Официальный сайт tensorflow [<https://www.tensorflow.org/tutorials>]
6. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms - Understanding object detection algorithms [<https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>]

# **Приложение**

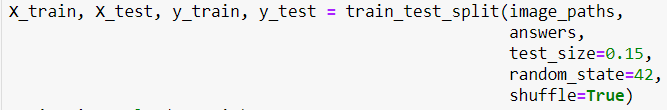
Ссылка на github:

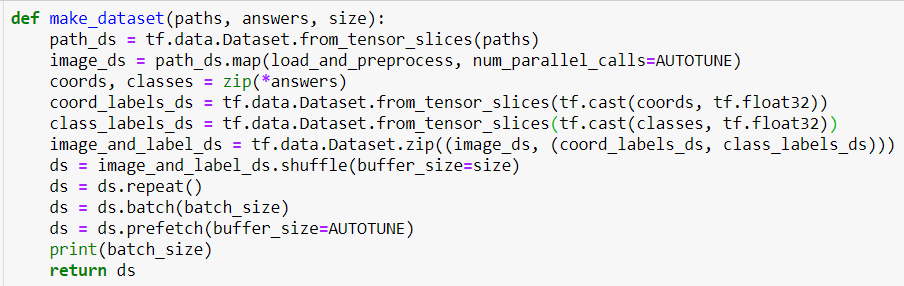
https://github.com/NoVarlok/coursework-ML

Предобработка данных:



Загрузка датасета:





Структура модели:



Обучение:

