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 2020

**Содержание**

[1. Введение 3](#_Toc9889027)

[2. Постановка задачи 4](#_Toc9889028)

[2.1 Описание задачи 4](#_Toc9889029)

[2.2 Входные/Выходные данные 4](#_Toc9889030)

[2.3 Пример работы программы 4](#_Toc9889031)

[3. Обзор существующих решений 6](#_Toc9889032)

[4. Описание выбранного решения 9](#_Toc9889033)

[4.1 Описание основных переменных 9](#_Toc9889034)

[4.2 Описание функций 9](#_Toc9889035)

[4.3 Описание логики алгоритма 10](#_Toc9889036)

[5. Описание тестирования программы 13](#_Toc9889037)

[6. Заключение 15](#_Toc9889038)

[7. Список литературы 16](#_Toc9889039)

[8. Приложение 17](#_Toc9889040)

**Введение.**

Машинное обучение (machine learning) – это один из подразделов искусственного интеллекта, ключевой особенностью которого является не прямое решение поставленной задачи, решение задачи путем анализа множества решений подобных задач. 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. Однако, зачастую задача локализации + классификации заменяется задачей детектирования объектов, так как она позволяет находить на изображении (в отличии от локализации + классификации). Тем не менее, в ситуациях, когда важна скорость получения результата и на изображении необходимо обнаружить только 1 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 “Симсоны”.

**Постановка задачи.**

**Описание задачи.**

Спроектировать и обучить модель, используя библиотеку глубокого обучения Tensorflow 2, определяющую персонажа и мультсериала “Симссоны” 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 приложений/обученной модели:

На вход принимается путь до изображения, 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.

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,

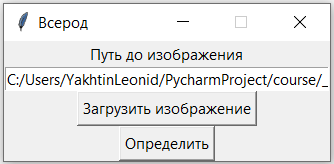
x2,y2 – координаты нижнего правого угла bounding box

Выходные данные для обучения модели:

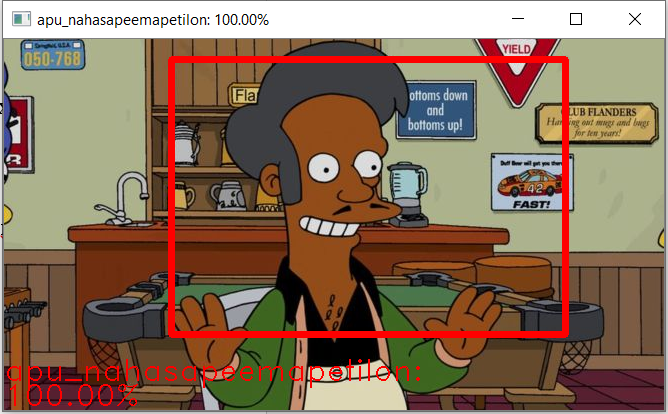
Csv файл со значениями функций потерь на обучающей и валидационной выборках и learning rate

h5 файл – сохраненная модель

**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:

1. Стандартные методы машинного обучения.

Решить 2 отдельных 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 значения пикселей изображений “выпрямляют” в 1 вектор, в результате чего изображение рассматривается не как единое целое, а как набор пикселей и теряется связь между пикселями.

1. Свёрточная нейронная сеть с 2 выходами – описано в пункте
2. 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 и bounding box этого объекта в выделенной области.

Так автор датасета, который был использован для обучения модели данной работы, использовал Faster R-CNN. Однако, автор отмечает несколько недостатков своего решения. Одним из них является скорость работы, так для обработки 1 изображения ему требовалось 8 секунд на CPU или 0.98 секунды на GPU (Tesla k80). Также автор упоминает 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 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”, 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 эта часть использовалась для обучения.

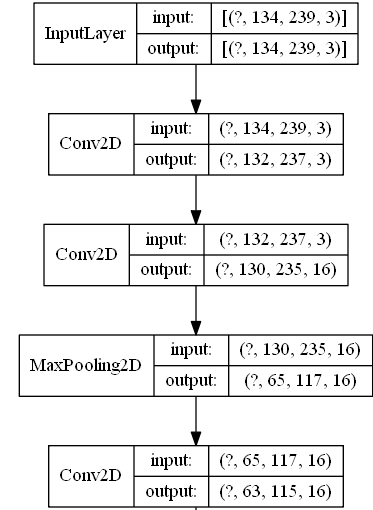
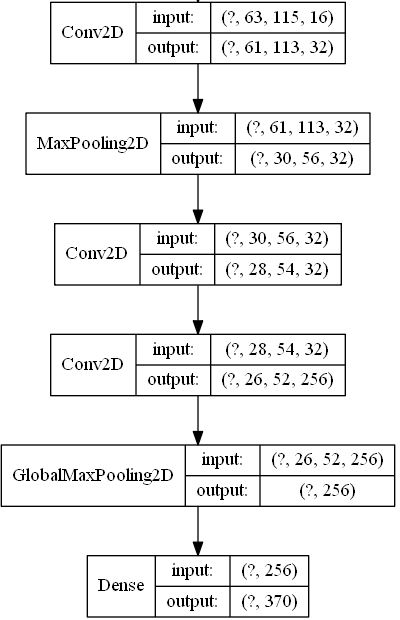
**Предобработка/аугментация данных.**

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 нейронная сеть почти всегда определяла bounding box в одной области в независимости от расположения персонажа. Для борьбы с этим были добавлены отраженные версии изображений.

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.

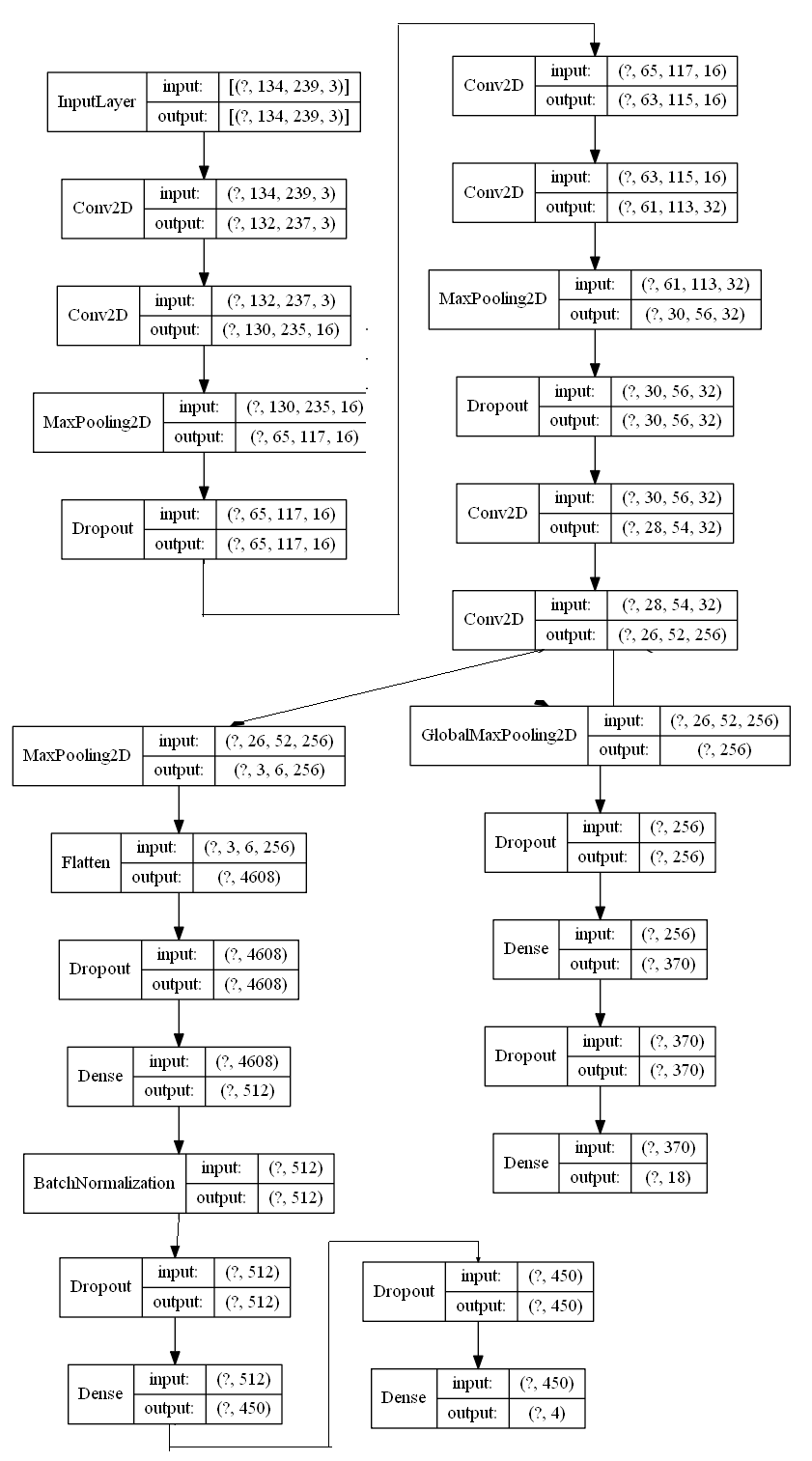
*Изображение №4 – тетстовая модель классифмкации.*

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 уменьшает число обучаемыз параметров. Также для еще большего уменьшения числа обучаемых парраметров тестировалась комбинация сверток с ядрами 1x3 и 3x1 всесто 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 увевиливает точность и способствует переобучению модели. Для локализации применяется MaxPooling с размером ядра 8x8 и результат преобразуется в одномерный вектор (здесь MaxPooling применяется всесто GlobalMaxPooling для получения вектора свойств больщего 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 актифации. Использование ELU в в среднем показывает более стабильно высокие результаты на этапе обучения, но в итоговой модели используется ReLU, так как при нем была максилатьная суммарная “точность” предсказаний.



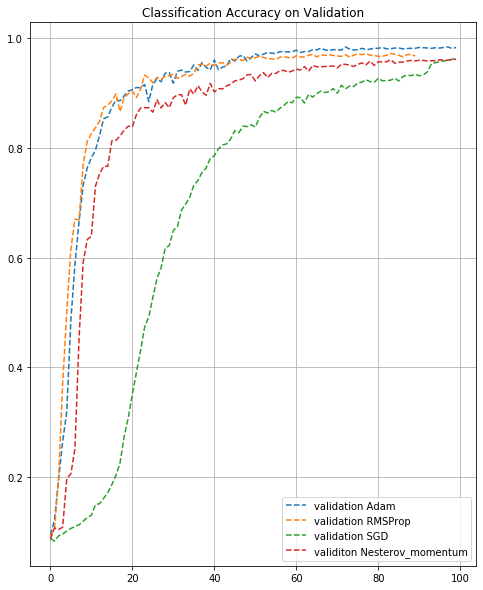
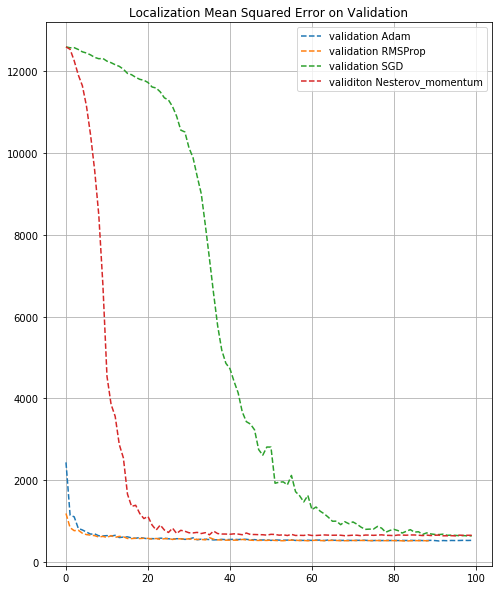
*Изображение №5 – структура итоговой модели.*

**Обучение модели.**

В качестве функций потерь были использованы log(MSE) для вызода, 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 протестированны SGD, SGD momentum, Nesterov momentum, RMSProp и Adam. Наилудший 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 значительно отставал от результатов других оптимизаторов.

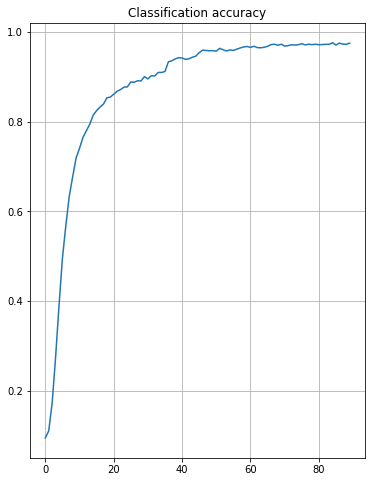
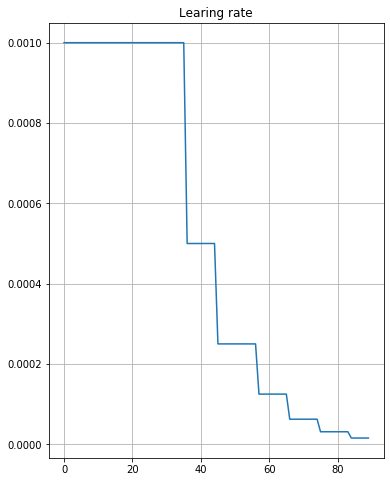
 

*Изображение №6 – графики метрик для разных алгоритмоы оптимизации*

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

Модеили обучались от 80 до 140 эпох, при этом эксперименты показали, что обучение более чем 100 эпох не улучшало метрики.

Для того, чтобы модель продолжала улучшать значения метрик на протяжении всего периоба обучения применялось уменьшение learning rate при выходе функции потерь на плато. Это реализовывалось callback-ом, уменьшаюшим 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 №7 – изменение изменение learning rate и classfication accuracy для RMSProp*

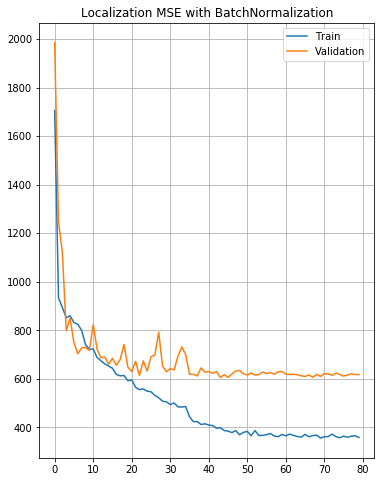
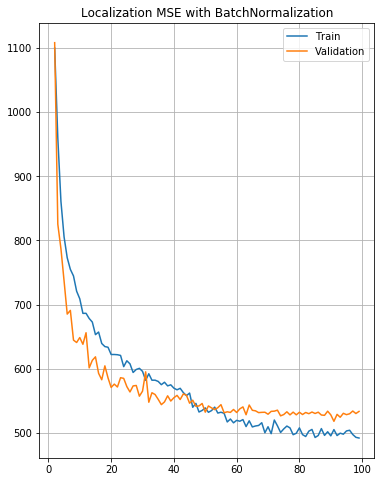
**Борьба с переобучением.**

Переобучение было одной из самой сложной проблем для данного датасета, так как дасет состоял из примерно 13 тысяч картинок, что относительно немного.

Для решения проблемы переобучения использовались следущие методы:

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

Для обучения использовалось 3 различных выроятности dropout:

* p = 0.3 – применяется менжду группами слоев conv2d + conv2d + maxpooling
* p = 0.5 – применяется между полносвязными слоями (особоенно 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 стандартномы распределению, что ускоряет обучение и в данной сети помогло уменьшить переобучение и MSE модели для выхода локализации.

*Изображение №8 – графики значений MSE с и без BatchNormalization.*

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