**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 27.04.04 – Управление в технических системах | |
| **Программа** | Управление и информационные технологии в технических системах | |
| **Факультет** | ФКТИ | |
| **Кафедра** | АПУ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Шестопалов М.Ю. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**магистра**

**Тема: ИССЛЕДОВАНИЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ КЛАССИФИКАЦИИ ИЗ ПАКЕТА ПРОГРАММ KERAS/TENSORFLOW НА ПРИМЕРЕ ЗАДАЧИ РАСПОЗНАВАНИЯ ДОРОЖНОЙ ОБСТАНОВКИ БЕСПИЛОТНОГО АВТОМОБИЛЯ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент(ка) |  |  |  | Кузубов А.О. |
|  |  | *подпись* |  |  |
| Руководитель | ассистент, к.т.н. |  |  | Никонов А.Н. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты | доцент каф.АПУ, к.т.н. |  |  | Белаш О.Ю. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | доцент каф.АПУ, к.т.н. |  |  | Власенко С.В. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | доцент каф.ПЭ, д.э.н. |  |  | Сулейманкадиева А.Э. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2019

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

|  |
| --- |
| Утверждаю |
| Зав. кафедрой АПУ |
| Шестопалов М.Ю. |
| « » 20 г. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент(ка) | Кузубов А.О. | | Группа | | | 3391 |
| Тема работы: Исследование нейросетевых моделей классификации из пакета программ Keras/TensorFlow на примере задачи распознавания дорожной обстановки беспилотного автомобиля | | | | | | |
| Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина) (СПБГЭТУ «ЛЭТИ») | | | | | | |
| Исходные данные (технические требования): описание задачи | | | | | | |
| Содержание ВКР:  Описание предметной области, средства разработки, проектирование, разработка, заключение. | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал. | | | | | | |
| Дополнительные разделы: Технико-экономическое обоснование | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | |
| «06» февраля 2019г. | | | «03» июня 2019г. | | | |
| Студент | |  | | Кузубов А.О. |
| Руководитель ассистент, к.т.н. | |  | | Никонов А.Н. |
| *(Уч. степень, уч. звание)* | |  | |  |

**КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

|  |
| --- |
| Утверждаю |
| Зав. кафедрой АПУ |
| Шестопалов М.Ю. |
| « » 20 г. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент(ка) | | | Кузубов А.О. | Группа | | 3391 | |
| Тема работы: Исследование нейросетевых моделей классификации из пакета программ Keras/TensorFlow на примере задачи распознавания дорожной обстановки беспилотного автомобиля | | | | | | | |
| №  п/п | Наименование работ | | | Срок вы-  полнения | |
| 1 | Обзор литературы и материалов по теме работы | | | 6.02 –  11.04 | |
| 2 | Составление технического задания | | | 11.04 –  15.05 | |
| 3 | Разработка набора данных | | | 15.04 –  27.04 | |
| 4 | Реализация | | | 27.04 –  17.06 | |
| 5 | Оформление пояснительной записки | | | 17.07 –  24.07 | |
| 6 | Оформление иллюстративного материала | | | 24.05 –  27.05 | |

|  |  |  |
| --- | --- | --- |
| Студент |  | Кузубов А.О. |
| Руководитель ассистент, к.т.н. |  | Никонов А.Н. |
| *(Уч. степень, уч. звание)* |  |  |

Реферат

Ключевые слова – ИССКУСТВЕННЫЕ НЕЙРОННЫЕ СЕТИ, PYTHON, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, ДОРОЖНЫЕ ЗНАКИ

**Актуальность** обусловлена повышением требований к разрабатываемым системам автоматического управления автомобилем, высокой опасностью участниками движения на дороге. Данная работа рассматривает возможности улучшения точности распознавания дорожных знаков за счет автоматизированной генерации обучающего набора. Второй задачей данной работы является выбор существующей архитектуры нейронной сети, наиболее подходящей для применения в реальной обстановке с использованием малых ресурсов на устройстве.

**Целью** исследования является решение процессов сборов данных на основе компьютерного зрения для прикладной задачи управления автомобилем в автоматическом режиме.

В ходе работы были выполнены поставленные задачи: анализ предметной области, знакомство с существующими методами решения задачи, сравнены теоретические и практические аспекты применения данных методов. Так же в ходе работы был разработан инструментарий для обучения разных моделей НС, анализа и сравнения результатов обучения.

Результаты данной работу могут найти практическое применения в высших научных заведениях, в научно-конструкторских организациях, а также в коммерческой деятельности.

**Степень достоверности** полученных данных обеспечивается теоретическими основами работы нейронных сетей. А также испытаниями на стенде. Результаты данной работы не рекомендуются для использования в реальных системах, так как не была применена проверка на реально полученных данных.

**Личный вклад.** В работе создан инструмент для генерации обучающего набора. Разработан инструмент для обучения, тестирования работы нейронной сети.

Abstract

In this paper, we study the problem of automatic character recognition. A formal statement of the problem made, as well as approaches to the solution of this problem are considered. Aspects of work of each architecture of convolutional neural network on an example of recognition of objects on the image are studied

Оглавление

[Оглавление 6](#_Toc10283207)

[Определения и обозначения 8](#_Toc10283208)

[Введение 9](#_Toc10283209)

[1. ОБЩИЕ СВЕДЕНИЯ О НЕЙРОННЫХ СЕТЯХ 10](#_Toc10283210)

[1.2. Модель нейронов 13](#_Toc10283211)

[1.3. Архитектура НС 15](#_Toc10283212)

[1.4. Сети прямого распространения 16](#_Toc10283213)

[1.5. Сверточные нейронные сети 16](#_Toc10283214)

[1.6. Обучение НС 19](#_Toc10283215)

[1.6.1. Обучение с учителем 19](#_Toc10283216)

[1.6.2. Обучение без учителя 19](#_Toc10283217)

[1.6.3. Обучение с подкреплением 20](#_Toc10283218)

[1.6.4. Обучение, основанное на коррекции ошибок 20](#_Toc10283219)

[1.6.5. Обучение Хебба 20](#_Toc10283220)

[1.6.6. Конкурентное обучение 20](#_Toc10283221)

[1.6.7. Обучение Хопфилда 21](#_Toc10283222)

[1.6.8. Обучение Больцмана 22](#_Toc10283223)

[1.6.9. Генетический алгоритм 23](#_Toc10283224)

[2. Сравнение архитектур компьютерного зрения 25](#_Toc10283225)

[2.2. VGG16 28](#_Toc10283226)

[2.3. Network-in-network 31](#_Toc10283227)

[2.4. InceptionV3 32](#_Toc10283228)

[2.5. Xception 35](#_Toc10283229)

[2.6. ResNet50 37](#_Toc10283230)

[2.7. MobileNet 38](#_Toc10283231)

[2.8. MobileNetV2 39](#_Toc10283232)

[2.9. DenseNet 40](#_Toc10283233)

[2.10. NASNet 41](#_Toc10283234)

[3. Распознание дорожных знаков на базе библиотеки tensorflow 42](#_Toc10283235)

[3.2. Общая схема работы системы 42](#_Toc10283236)

[3.3. Сбор и предварительная обработка данных 42](#_Toc10283237)

[3.4. Обучение нейронной сети 46](#_Toc10283238)

[3.5. Обучение на 28 классах 47](#_Toc10283239)

[3.5.1. Определение моделью xception 49](#_Toc10283240)

[3.5.2. Определение моделью Inceptionv3 51](#_Toc10283241)

[3.5.3. Определение моделью mobileNet 54](#_Toc10283242)

[3.5.4. Определение моделью mobileNetv2 57](#_Toc10283243)

[3.5.5. Определение моделью DenseNet 59](#_Toc10283244)

[3.6. Обучение на 51 классе 62](#_Toc10283245)

[3.6.1. Распознавание Xception 64](#_Toc10283246)

[3.6.2. Распознавание resnet 66](#_Toc10283247)

[3.6.3. Работа mobileNet 70](#_Toc10283248)

[3.6.4. Распознавание mobileNetV2 73](#_Toc10283249)

[3.6.5. Распознавание сети DenseNet 76](#_Toc10283250)

[3.7. Вывод к главе 78](#_Toc10283251)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 80](#_Toc10283252)

[4.2. Себестоимость проекта 80](#_Toc10283253)

[4.3. Вывод 83](#_Toc10283254)

[Заключение 84](#_Toc10283255)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 85](#_Toc10283256)

Определения и обозначения

**Искусственная нейронная сеть (ИНС) —** математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса [1]. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

**Нейрон (искусственный)** — узел искусственной нейронной сети, являющийся упрощённой моделью естественного нейрона. Математически, искусственный нейрон обычно представляют, как некоторую нелинейную функцию от единственного аргумента — линейной комбинации всех входных сигналов. Данную функцию называют функцией активации или функцией срабатывания, передаточной функцией. Полученный результат посылается на единственный выход. Такие искусственные нейроны объединяют в сети — соединяют выходы одних нейронов с входами других.

**Система управления (СУ)** — систематизированный (строго определённый) набор средств сбора сведений о подконтрольном объекте и средств воздействия на его поведение, предназначенный для достижения определённых целей. Объектом системы управления могут быть как технические объекты, так и люди. Объект системы управления может состоять из других объектов, которые могут иметь постоянную структуру взаимосвязей.

**Very-large-scale-inte-grated (VLSI)** — (англ.) интегральные схемы

**РНС** - рекуррентная нейронная сеть

**Кластеризация (или кластерный анализ)** — это задача разбиения множества объектов на группы, называемые кластерами.

Введение

Современный мир сейчас переживает бурный рост популярности нейросетевых технологий. Повсеместно используются алгоритмы распознавания изображений, текста, речи, основанные на работе с нейронной сетью. Однако, применение ИНС является ресурсно-затратной задачей, так как требует длительных расчетов при предсказании значений, а также при обучении. Одной из причин сложности расчетов является несовершенность применяемых архитектур нейронных сетей, а также методов обучения сетей.

1. ОБЩИЕ СВЕДЕНИЯ О НЕЙРОННЫХ СЕТЯХ

Появление искусственных нейронных сетей (ИНС) прежде всего связано с изучением работы мозга. Среди первых работ связанных с изучением нейронных сетей была статья о логическом исчислении идей и нервной активности У. Маккалока и У. Питтса [1] Мозг представляет собой чрезвычайно сложный, нелинейный, параллельный компьютер (систему обработки информации). Он обладает способностью организовывать свои структурные компоненты, называемые нейронами, так, чтобы они могли выполнять конкретные задачи (такие как распознавание образов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые современные компьютеры. Более точно, мозг последовательно выполняет ряд задач распознавания (например, распознавание знакомого лица в незнакомом окружении). На это у него уходит около 100-200 миллисекунд, в то время как выполнение аналогичных задач даже меньшей сложности на компьютере может занять несколько дней.

Понятие развития нейронов связано с понятием пластичности мозга — способности настройки нервной системы в соответствии с окружающими условиями. Именно пластичность играет самую важную роль в работе нейронов в качестве единиц обработки информации в человеческом мозге. Аналогично, в искусственных нейронных сетях при компьютерном моделировании используются основополагающие принципы работы нейронной сети:

* 1. Знания поступают в нейронную сеть из окружающей среды и используются в процессе обучения
  2. Для накопления знаний применяются связи между нейронами, называемые синаптическими весами.

Отличительной способностью нейронных сетей — создавать обобщения. Под обобщением понимается способность получать обоснованный результат на основании данных, которые не встречались в процессе обучения. Эти свойства позволяют нейронным сетям решать сложные и масштабные задачи, которые на сегодняшний день считаются трудноразрешимыми. Однако на практике при автономной работе нейронные сети не могут обеспечить готовые решения. Их необходимо интегрировать в сложные системы. В частности, комплексную задачу можно разбить на последовательность относительно простых, часть из которых может решаться нейронными сетями.

Использование нейронных сетей обеспечивает также полезные свойства системы [1]:

* 1. *Нелинейность.* Нейронные сети, построенные из соединений нелинейных нейронов, сами являются нелинейными. Нелинейность обеспечивает устойчивость обучения нейронной сети в нелинейных системах.
  2. *Отображение входной информации в выходную.* Большая неформализованная информация может быть представленная в обобщенном и сжатом виде.
  3. *Адаптивность.* Нейронные сети обладают способностью адаптировать свои синаптические веса к изменениям окружающей среды. В частности, нейронные сети, обученные действовать в определенной среде, могут быть легко переучены для работы в условиях незначительных колебаний параметров среды.

Более того, для работы в нестационарной среде, где статистика изменяется с течением времени, могут быть созданы нейронные сети, изменяющие синаптические веса в реальном времени. Известно, что чем выше адаптивные способности системы, тем более устойчивой будет ее работа в нестационарной среде, однако адаптивная система с параметрами, быстро изменяющимися во времени, может также реагировать и на посторонние возбуждения, что вызовет потерю производительности. Для того чтобы использовать все достоинства адаптивности, основные параметры системы должны быть достаточно стабильными, чтобы можно было не учитывать внешние помехи, и достаточно гибкими, чтобы обеспечить реакцию на существенные изменения среды. Эта задача обычно называется дилеммой стабильности-пластичности

* 1. *Очевидность ответа.* В контексте задачи классификации образов можно разработать нейронную сеть, собирающую информацию не только для определения конкретного класса, но и для увеличения достоверности принимаемого решения. Впоследствии эта информация может использоваться для исключения сомнительных решений.
  2. *Контекстная информация.* Знания представляются в самой структуре нейронной сети с помощью ее состояния активации. Каждый нейрон сети потенциально может быть подвержен влиянию всех остальных ее нейронов. Как следствие, существование нейронной сети непосредственно связано с контекстной информацией
  3. *Отказоустойчивость.* Принимая в расчет распределенный характер хранения информации в нейронной сети, можно утверждать, что только серьезные повреждения структуры нейронной сети существенно повлияют на ее работоспособность. При мелких же повреждениях, снижение качества работы нейронной сети происходит медленно.
  4. *Масштабируемость.* Параллельная структура нейронных сетей потенциально ускоряет решение некоторых задач и обеспечивает масштабируемость нейронных сетей в рамках технологии VLSI (very-large-scale-inte-grated).
  5. *Единообразие анализа и проектирования.* Нейронные сети являются универсальным механизмом обработки информации. Это означает, что одно и то же проектное решение нейронной сети может использоваться во многих предметных областях. Это свойство проявляется несколькими способами.

Нейроны в той или иной форме являются стандартными составными частями любой нейронной сети.

Эта общность позволяет использовать одни и те же теории и алгоритмы обучения в различных нейросетевых приложениях.

Модульные сети могут быть построены на основе интеграции целых модулей.

Нейросетевые сети в системах управления могут использоваться 2-мя способами [2]:

* 1. *режим online* — Нейросеть обучается и одновременно формирует управляющее воздействие на входе исполнительного устройства системы управления. Цель обучения сети и цель управления объектом совпадают, что отражается в задании единой целевой функции системы. Сеть обучается в реальном времени, в темпе протекания процессов в системе.
  2. *режим offline* — Работа сети состоит из двух этапов: 1) предварительного этапа обучения сети заданной оптимальной функции управления и 2) этапа воспроизведения аппроксимации этой функции в режиме управления объектом при тех же условиях или близких к ним. Целевые функционалы обучения сети и управления объектом могут отличаться друг от друга. Такой вариант применения нейронной сети для управления — так называемое супервизорное управление — нашел преимущественное распространение до настоящего времени, хотя процесс синтеза нейросетевого контроллера и настройка его параметров в этом случае протекает не в реальном времени

Нейронные сети можно классифицировать по:

* 1. типу входной информации
  2. по характеру обучения
  3. по характеру передачи сигнала
  4. по структуре
  5. Модель нейронов

Основой работы любой нейронной сети является наличие накопителей информации. Такие накопители информации называются нейронными. Их механизм работы основан на работе биологического нейрона головного мозга.

Нейрон (Рисунок 1) — электрически возбудимая клетка нервной системы, которая обрабатывает, хранит и передает информацию с помощью электрических и химических сигналов. Имея связи с другими нейронами, с помощью дентритов и аксонов, передает импульс, усиливая или ослабляя его силу.

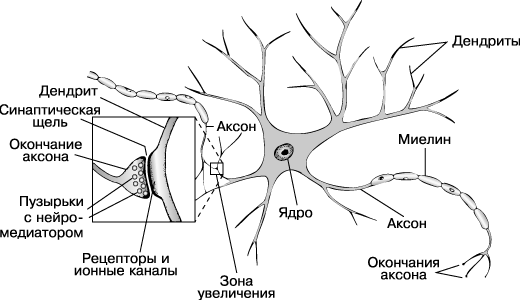


Рисунок 1 - Биологический нейрон

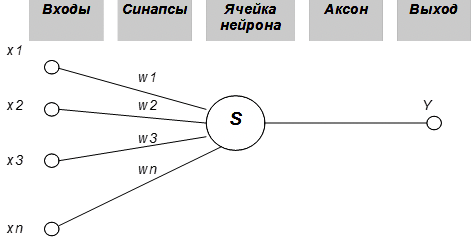


Рисунок 2 - Искусственный нейрон

Искусственный нейрон построен по схожей схеме (рис. 2), что и биологический. Это узел также состоит из несколько входов и несколько выходов. На вход принимает несколько аргументов, суммирует эти аргументы, перемножает или вывод единственный (общий) аргумент из всех по какой-нибудь линейной комбинации. На выходе значение нелинейной функции этого общего аргумента. Данную функцию называют функцией срабатывания. Полученный результат посылается на выход или сразу на несколько выходов, к другим нейронам. Связанные между собой нейроны и есть нейронные сети.

* 1. Архитектура НС

На сегодняшний день существует большое множество реализаций нейронных сетей с разными структурами:

* + - * перцептрон Розенблатта
      * радиально-базисные функции
      * самоорганизующиеся карты
      * сплайн-модель Хакимова
      * сеть Джордана
      * Ворда
      * Хопфилда
      * Кохонера
      * нейронный газ
      * когнитрон/неокогнитрон
      * хаотическая нейронная сеть
      * осцилляторная НС
      * сеть встречного распространения
        + сеть Смирнова
        + вероятностная сеть Решетова
        + сети адаптивного резонанса
        + сверточная НС
        + импульсная
        + сеть глубокого доверия (deep belief network) или Байесовская сеть
        + генетический алгоритм

Как видно, за последнее время накопилось очень много разных реализации ИНС. Однако многие из них основаны на общих принципах. Рассмотрим эти основные принципы и наиболее известные архитектуры.

* 1. Сети прямого распространения

Сети прямого распространения – это как правило многослойный перцептрон. Эта сеть обычно состоит из некоторого количества входов, узлов источника, по-другому — сенсорных элементов, прощупывающих внешнюю среду ИНС. Это множество входов образует входной слой. Далее они передают сигналы скрытым слоям, которые производят вычислительные операции и никак не связаны с окружающим миром. Далее они передаю сигнал выходному слою, который, в свою очередь, и предоставляет информацию выходных нейронах.

Как видно в сетях прямого распространения сигнал всегда движется в одну сторону: от входа к выходу, проходя все слои по одному разу.

Это удобно использовать для большинства обучающих алгоритмов.

Обучение может производится разными алгоритмами, к примеру.

* 1. Сверточные нейронные сети

Данная архитектура нейронных сетей была предложена французским ученым Яном Лекуном. Сверточные нейронные сети как правило используются для распознавания изображений. Также применяются в составе технологий Google «Глубинное обучение»

Фундамент для создание такой архитектуры является открытие некоторых особенностей зрительной коры, в которых клетки, реагирующие на прямые световые лучи под разными углами, назовем простые клетки, и клетки, которые активируются от совокупной реакции нескольких простых клеток — сложные. В мозге живого организма, сверточные нейронные сети работают на основе чередования сверточных слоев и субдискретизирующих слоев, в котором каждый из слоев отвечает за свою функциональность.

Так структура искусственных нейронных сетей — многослойная, однонаправленная. Обучение как правило используется на основе стандартных методов, к примеру метод обратного распространения ошибки. Функция активации — любая. Основной отличительной особенностью сверточных нейронных сетей, наличием у нее операции свертки. Операция свертки — это каждый фрагмент изображения умножается на матрицу свертки, еще называют ядром свертки, поэлементно, результат же суммируется и записывается в соответствующую позицию в новом изображении. Для того, чтобы при операции свертки у изображения не появлялось рамок, изображение, проходя через операцию свертки, уменьшается в размерах, сохраняя важные для задачи детали. Еще операцию свертки изображения можно назвать фильтрацией, где матрица свертки — это цифровой фильтр.

В отличие от обычного перцептрона, в котором каждый нейрон искусственной нейросети связан со всеми из предыдущего, в сверточной нейронной сети используется небольшая матрица, которая перемещается по всему входящему слою. Эта матрица и формирует сигнал активации для соответствующего нейрона следующего слоя. То есть на каждом слое используется некоторая небольшая матрица весов, одинаковая для всех нейронов в слое. Благодаря ее использованию уменьшается размер используемой нейронной сети, снижает ресурсные потребности для ЭВМ, и обеспечивает одинаковую обработку всех нейронов и более быстрое обучение, так как в сверхточной сети меньше тренируемых параметров.

Однако, сверхточные в задачах распознавания и получения конечной информации из изображения используются только на первых слоях ИНС. Так как они являются цифровыми фильтрами и обрабатывают только соседние пиксели, находящиеся в окне свертки. Они помогают получить первичные простейшие признаки на изображении, линии, образование простейших фигур. Но взаимное расположение признаков на изображении они не видят. Для дальнейшей обработки полученных результатов применяются полносвязные слои (перцептроны)

В сверточной нейросети набор весов не один, и результатами свертки могут быть самые разные признаки. Причем они могут меняться от слоя к слою. Так следующий слой в после свертки получит изображение как карту признаков. И проанализировав следующей, или той же, матрицей свертки увидит новые признаки, как например простые фигуры: круг, квадрат, и т. д. В конце последний слой анализируя свои признаки может определить где находится дом или чье-нибудь лицо. Отметим, что матрицы свертки не задаются заранее, а является результатом обучения. Это можно воспринимать как у перцептрона набор синапсов: все со всеми. Однако, здесь матрица как бы сохраняет одинаковые веса для соответствующих групп нейронов, блоков сверточного изображения. Каждый проход обучения создает свой собственный набор признаков, делая ИНС многоканальной — много независимых карт признаков на одном слое. Кстати блоки изображения (упомянутые выше) могут быть самых разных размеров, от и до , эти блоки задают размер матрицы. Одна матрица не «шагает» по блокам, она может «переступить» признаки, а делает маленькие шаги, по 1-3 пикселя.

Cубдискретизация — операция, выполняющая уменьшение размерности полученных карт признаков. Как правило при реализации сверточных искусственных сетей информация о наличии признака намного важнее знания о координатах на изображении этих признаков. Поэтому можно использовать операцию субдискретизации. В которой, из нескольких соседних нейронов на карте признаков выбирается наиболее важный или наиболее выраженный и принимается за один нейрон. Получается сжатая карта признаков эта операция ускорят дальнейшие вычисления, на следующих слоях, а также ИНС становится инвариантной к размерности исходного входного вектора, т.е. изображения.

Рассмотрим работу сверточной ИНС более подробно. Сеть состоит из небольшого количества слоев. На начальный слой, на вход, поступает изображение. Это изображение проходит серию сверточных слоев, в которых чередуется свертка и субдискретизация. Свертка образует карты признаков. Далее наименее важные отсеиваются. От слоя к слою карта признаков уменьшается в размере, но, при этом, увеличивается количество его каналов. Это обеспечивает способность ИНС распознавать более сложные признаки с более сложными иерархиями признаков. После прохождения через несколько слоев карта признаков вырождается в вектор или даже скаляр. Таких карт признаков становится очень много. На выходе сверточной нейронной сети может дополнительно использоваться несколько слоев полносвязной нейронной сети — к примеру, перцептрон. На вход ему подаются оконечные карты признаков, перцептрон же выводит оконченный результат в понятном пользователю виде.

* 1. Обучение НС

Самым важным свойством нейронных сетей является их способность обучаться на основе данных окружающей среды и в результате обучения повышать свою производительность. Повышение производительности происходит со временем в соответствии с определенными правилами. Обучение нейронной сети происходит посредством интерактивного процесса корректировки синаптических весов. В идеальном случае нейронная сеть получает знания об окружающей среде на каждой итерации процесса обучения.

* + 1. Обучение с учителем

Если нейронная сеть обучается с использованием заранее известных правильных ответов, то такой алгоритм обучения называется - обучение с учителем. Необходимо отметить, что при обучении с учителем требуется большая выборка, чтобы в достаточной мере сформировать рабочую и гибкую нейронную сеть.

* + 1. Обучение без учителя

Алгоритм обучения без учителя применяется тогда, когда известны только входные данные. На основе их сеть учится выдавать наилучшие выходные - одинаковые при схожих входных.

Они могут приняться в кластеризации и нахождении образов от самих обьектов.

* + 1. Обучение с подкреплением

Еще один вид обучения нейронной сети, при котором НС обучается, взаимодействуя со средой. Откликом среды являются сигналы подкрепления, а не ошибки, как при обучении с учителем.

* + 1. Обучение, основанное на коррекции ошибок

Метод обучения перцептрона, предложенный Фрэнком Розенблаттом.

**Алгоритм** представляет собой такой метод обучения, при котором вес связи не изменяется до тех пор, пока текущая реакция перцептрона остается правильной. Если выход не совпадает с желаемым, то параметры сети будут скорректированы. Для вычисления величины коррекции используется разница между реальным и желаемым выходом сети.

* + 1. Обучение Хебба

Обучение Хебба основано на физиологических и психологических исследованиях.

**Алгоритм** обучения можно представить в виде правил из 2 частей:

* 1. Если два нейрона по обе стороны синапса активизируются одновременно, то прочность этого соединения возрастает.
  2. Если два нейрона по обе стороны синапса активизируются асинхронно, то такой синапс ослабляется или вообще отмирает.
     1. Конкурентное обучение

**Алгоритм**. В отличие от обучения Хебба, в котором множество нейронов могут возбуждаться одновременно, при конкурентном обучении выходные нейроны соревнуются между собой за активацию. То есть из всего множества выходных нейронов используется только один – с наибольшим выходом. При обучении модифицируются только веса победившего нейрона. И так наилучший подходящий нейрон становится ближе к примеру.

**Применение**. Конкурентное обучение очень удобно использовать для изучения статистических свойств, используемых в задачах классификации входных образов. В этом случае каждый нейрон выходного слоя отвечает за один образ.

* + 1. Обучение Хопфилда

Эта модель обучения применяется для рекуррентных полносвязных нейронных сетей. В процессе работы динамика таких сетей сходится к одному из положений равновесия. Это можно сравнить с термодинамикой, где система всегда стремится к покою. Эти положения равновесия определяются заранее в процессе обучения, они являются локальными минимумами функционала, называемого энергией сети.

**Алгоритм**. Обучение сети заключается в том, что находятся веса матрицы взаимодействий так, чтобы запомнить векторов — эталонных образов, составляющих память системы. Вычисление коэффициентов основано на следующем правиле: для всех запомненных образов матрица связи должна удовлетворять уравнению

поскольку именно при этом условии состояния сети будут устойчивы — попав в такое состояние, сеть в нём и останется.

Запоминаемые векторы должны иметь бинарный вид. Расчёт весовых коэффициентов проводится по следующей формуле:

,где — размерность векторов, — число запоминаемых выходных векторов, — номер запоминаемого выходного вектора, — i-я компонента запоминаемого выходного j-го вектора.

Это выражение может стать более ясным, если заметить, что весовая матрица может быть найдена вычислением внешнего произведения каждого запоминаемого вектора с самим собой и суммированием матриц, полученных таким образом. Это может быть записано в виде

,где — i-й запоминаемый вектор-столбец.

Расчёт этих весовых коэффициентов и называется обучением сети, которое проводится только за одну эпоху.

* + 1. Обучение Больцмана

Эта модель обучения получила свое название в честь Людвига Больцмана и его работы. Представляет собой стохастический алгоритм обучения, основанный на идеях стохастической механики. Применяется для обучения рекуррентных нейронных сетей

**Алгоритм.** Как и в сети Хопфилда, в машине Больцмана есть понятие «Энергии». И расчет этой энергии производится таким же способом, как и в сети Хопфилда.

,где — сила связи между нейронами и , — состояние, нейрона i, — порог для нейрона .

Как и в машине Хопфилда задачей обучения машины Больцмана является стабилизация состояния сети. Однако (в отличие от сети Хопфилда) машина Больцмана ищет не локальные минимумы (и не «застревает» них), а старается попасть в глобальные минимумы. Идея использования «теплового шума» для выхода из локальных минимумов и повышения вероятности попадания в более глубокие минимумы принадлежит С. Кирпатрику. На основе этой идеи разработан алгоритм имитации отжига.

,где — некий аналог теплового шума сети, — сумма весов связей К-го нейрона со всеми активными в нейронами.

**Применение**. Хотя возможности обучения машины Больцмана ограничены на практике, эти проблемы могут быть решены применением архитектуры ограниченной машины Больцмана. В этой архитектуре связи существуют только между скрытыми и видимыми нейронами, но при этом отсутствуют между нейронами одного класса. Такая архитектура изначально использовалась Полом Смоленски в 1986 году под названием Harmonium, но приобрела популярность только после изобретения Хинтоном быстрых алгоритмов обучения в середине 2000-х годов.

* + 1. Генетический алгоритм

Генетический алгоритм – это просто метод поиска наилучшей комбинации путем перебора, где выбираются наилучшие и наихудшие вариации решений. Этот процесс можно сравнить с естественным отбором в природе. Производится отбор, при котором остаются только те, которые удовлетворяют требованиям. Хорошие же результаты скрещиваются – производится рекомбинация решений.

**Алгоритм**. Некоторым, обычно случайным, образом создаётся множество генотипов начальной популяции. Они оцениваются с использованием «функции приспособленности», в результате чего с каждым генотипом ассоциируется определённое значение — «приспособленность», которое определяет насколько хорошо фенотип, им описываемый, решает поставленную задачу.

Из полученного множества производится отбор — «селекция». Наихудшие решения отбрасываются. К наилучшим применяются «Генетические операторы». Как правило это скрещивание — производит операцию рекомбинации решений-кандидатов или «мутация» — при котором удачные решения несильно изменяются. Получается новое поколение решении, которое также проходит описанные выше действие.

Это процесс селекции будет проводится, по не будет выполнен критерий остановки алгоритма:

* + - * + нахождение глобального, либо субоптимального решения
        + исчерпание числа поколений, отпущенных на эволюцию
        + исчерпание времени, отпущенного на эволюцию

1. Сравнение архитектур компьютерного зрения

Все современные архитектуры компьютерного зрения, способные извлекает и анализировать полезную информацию из изображения, их последовательности, видеоряда основаны на сверточных архитектурах, основная идея которых основана на переиспользовании одних и тех же нейронных слоев, и сетей с разными локальными участками входов.

Данный подход схож со строением головного мозга и функциональными зонами зрительной коры:

* + - * + V1 - зона, в которой выделяются локальные признаки локальных небольших участков, считываемого с печатки глаза.
        + V2 - выделение признаков с учетом бинокулярного зрения
        + V3 - распознание цвета, текстуры объектов. Сегментация и группировка.
        + V4 - распознание геометрических фигур, и очертания объектов. Модуляция посредством внимания человека - активация нейронов неравномерна и зависит от того, на что человек обращает внимание
        + V5 - Распознание движения, выделение объектов, их скорость
        + V6 - обобщение данных о картине. Изменения картинки вследствие движения человека.
        + V7 - распознание сложных объектов, в том числе человеческих лиц.

В ИНС также могут ввести механизмы внимание, помогающие выбрать, какие именно входы и промежуточные выходы сети нужно учитывать.

Основным преимуществом сверточных нейронных сетей над полносвязными, является то, что они по своей архитектуре подготовлены для работы с изображениями. Так как полносвязные архитектуры не учитывают топологию и структуры изображения, они анализируют изображение по отдельным пикселям и связям между ними. Однако, структура картинки играет важную роль - взаимное расположение близлежащих пикселов позволяет выделить объект на изображении или распознать символ. К тому же еще одним плюсом сверхточных нейронных сетей является то, что обработка локальной части от изображения происходит независимо от конкретного расположения этого участка. К примеру, если вы хотите определить, что за цифра на фотографии, вам не важно, что цифра отступает от левого края изображения на 100 или 200 пикселов. Однако в конце концов, на конечный результат анализа изображения влияет и конечное взаимное расположение объектов на изображении. Так, чтобы понять, что на изображении чье-то лицо мы должны точно быть уверены, что на изображении есть нос, рот, глаза, и что глаза расположены над носом, а нос над ртом. Поэтому в популярных глубоких нейронных сетях последним слоем или последними думы слоями так же используются и полносвязные слои уже для более автоматического выделения структур и их комбинации.

**Определим формальное понятие операции свертки**. Прежде всего разберемся с понятиями каналов в изображении. Обычно цветные картинки, подающиеся на вход нейронной сети, представлены в виде набора прямоугольных матриц, как правило трех. Каждая прямоугольная матрица задает уровень своего цвета для каждого пикселя конечного изображения. Для изображений RGB таких каналов три, отвечающих за красный, зеленый и синий цвета. Так изображение размером 200 на 200 пикселов будет иметь 120 000 значений пикселов для трех матриц разных цветов. Для черно-белых изображений канал будет только один. Далее будем предполагать, что в каждом пикселе конечного изображения будет стоять одномерный тензор - вектор чисел, его числа - каналами.

Такие же матрицы мы будем получать и после преобразование сверхточным слоем - в них по-прежнему сохраниться пространственная структура, соответствующая исходной картинке. Однако каналов теперь будет больше. В каждом казали мы будем выделять определенные признаки изображения. А каждая матрица будет называться картой признаков.

Сверка — это линейное преобразование:

 , где — результат свертки на уровне , а — ее вход, карта признаков в слое . По-другому данную операцию можно представить, как линейное преобразование к квадратному окну предыдущего уровня. Скалярное умножение пикселов из окна на вектор свертки. К примеру представим скалярное перемножения матрицы весов   размера 5 на 5 на ядро свертки размера . В матрице выбираются окна размером и скалярно перемножаются на . Таких перемножений получается 9 штук.

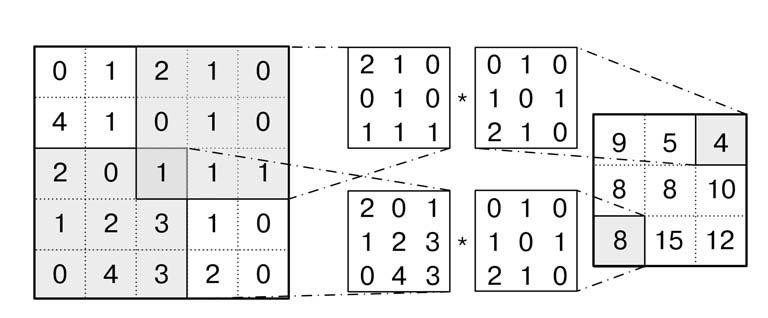


Рисунок 3 - Свертка изображения

На рисунке мы отобразили свертку признаков матрицы с помощью ядра свертки через сказанное произведение .

Такое перемножение обладает как раз теми свойствами, оговоренными выше:

* + - * + Свертка сохраняет структуру входа. Расположение пикселов и в последующем характеристик локальных участков сохраняется;
        + Операция свертки обладает свойством разрежённости, благодаря тому, что значение каждого из нейронов очередного слоя зависит только от небольшой доли входных нейронов. Если сравнивать с полносвязной ИНС, в котором нейрон последующего слоя связан со всеми нейронами предыдущего;
        + Свертка многократно переиспользует одни и те же веса, так как они повторно применяются к различным участкам входа.

После каждого сверхточного слоя необходимо выделить наиболее значимые признаки, для этого применяют функцию активации:

, где — функция активации. Как правило используют функцию активации ReLU, особенно в очень глубоких сетях. Также могут встречаться такие функции активации как и .

* 1. VGG16

Это классическая глубокая сверхточная нейронная сеть предложенная K. Simonyan и A. Zisserman из Оксфордского университета [3]. При тестировании на наборе ImageNet [4] данная нейронная сеть достигает точности 92.7%.

**Архитектура** основана на использовании большого количества сверточных слоев. Так на вход подается изображение размером , проходит через набор сверхточных слоев. Ядро свертки при этом на каждом сверхточном слое очень мало . После прохождения операции свертки производится операция max-polling, в котором на окне размером с шагом 2 выбирается наибольшее (иногда наименьшее) значение, выбирая наиболее важные признаки. При этом после прохождения каждого слоя карта признаков, или по другом число каналов изображения увеличивается, а размер изображения уменьшается, на входе оно равно 3 далее 128, 256, 512 и наконец 4096 каналов признаков. Полученный тензор - вектор размером 4096 подается на полносвязную сеть, на выходе которого тензор размера 1000 показывают вероятностные характеристики принадлежности объекта на изображении к определенному классу.

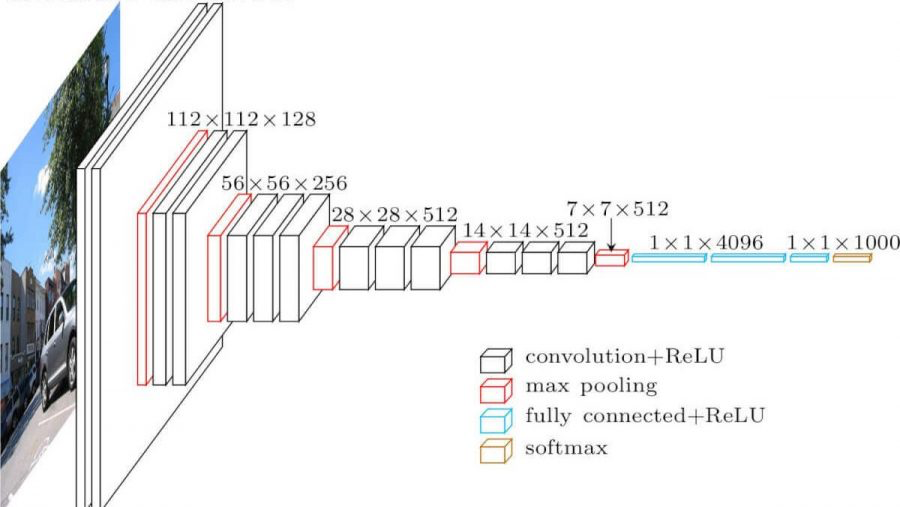


Рисунок 4 - Структура VVG16

К сожалению, сеть VVG имеет два серьезных недостатка:

* + 1. Очень медленная скорость обучения
    2. Сама архитектура сети весит слишком много

Из-за глубины и количества полносвязных узлов, данная ИНС весит более 500 МБ. Что приводи к значительным трудностям ее использования.

Формальное представление работы сети.

, где - функция активации ReLU, - функция активации softmax. - ядро свертки, - веса полносвязного слоя ИНС.

* 1. Network-in-network

Основой архитектуры послужила одна простая идея - использование сверток для увеличения комбинаторности извлекаемых характеристик предыдущего слоя для следующих слоев.

Как нам уже известно, сверточные сети работаю по принципу скалярного произведения ядра свертки на предыдущий слой.

С последующим выделением наиболее важных и выраженных характеристик к помощью применения функций активаций и maxpooling.

Данный подход приводит к тому, что мы можем потерять важные характеристики предыдущего слоя при прохождении свертки. Если они находятся рядом, один может просто преобладать над другим, при выделении с помощью maxpooling. Второй важной проблемой является то, то мы на этапах свертки не можем выделять сложные фигуры и учитывать взаимное расположение разных характеристик при свертке. Все эти проблемы решились большими полносвязными нейронными сетями на последних слоях модели.

Однако этот подход можно также применять и на сверхточных блоках. Такая идея была предложена в архитектуре нейронной сети Network in Network [5].

Рисунок 5 - Архитектура нейронной сети Network in Network

В NiN после прохождения операции свертки применяются дополнительные пространственные слои MLP. Это полносвязные нейроны, которые домножаются на полученную карту характеристик после свертки. Так производится анализ взаимного расположения качеств в окне свертки. С первого взгляда, добавление дополнительного, да и еще полносвязного слоя может раздуть нейронную сеть, что приведет к более длительному обучению и обработке изображений. Однако на практике это оказывается лучше применения голых пикселей и большого количества характеристик и сверхточных слоев. Так после свертки можно использовать меньшее количество параметров, являющихся общими для всех пикселей этих свойств.

* 1. InceptionV3

Берет свое начала от сети Алексея Крижевского AlexNet [6]. Особенностью которого было:

* + - * + Линейная ректификация ReLU в качестве нелинейности.
        + Использование методики отбрасывания для выборочного игнорирование нейронов при обучении. Метод dropout.
        + Перекрытие maxpooling - избежание усреднения average pooling

Однако, так как сеть AlexNet появилась до VGG, в ней были заложены старые принципы применения больших ядер сверток или и др.

Дальнейшее изучение архитектур нейронных сетей и желание их оптимизации привело Кристиана Жегеди к созданию архитектуры Inception [7]. Особенностью которой было применение одновременно нескольких подходов свертки:

* + - * + Использование малых сверточных ядер с большим количеством слоев, использую идею VGG;
        + Уменьшать количество каналов (измерений) последующих слоев с применением свертки. То есть выделение обобщенных характеристик для окна свертки последующей обработкой NiN сетью. Что позволяет избежать чрезмерное использование огромного объема характеристик для каждого пикселя;
        + На каждом уровне используем ядра свертки разного размера: , , , что позволяет получать характеристика для разного масштаба. Также для более больших ядер мы получаем более быстрое обучение.
        + Не используем плоских полносвязных слоев вообще, так как в них очень много параметров. Вместо этого на последнем уровне используем Global average pool - усреднение результатов

Так был получена эффективная нейронная сеть с комбинацией различных ядер [8].

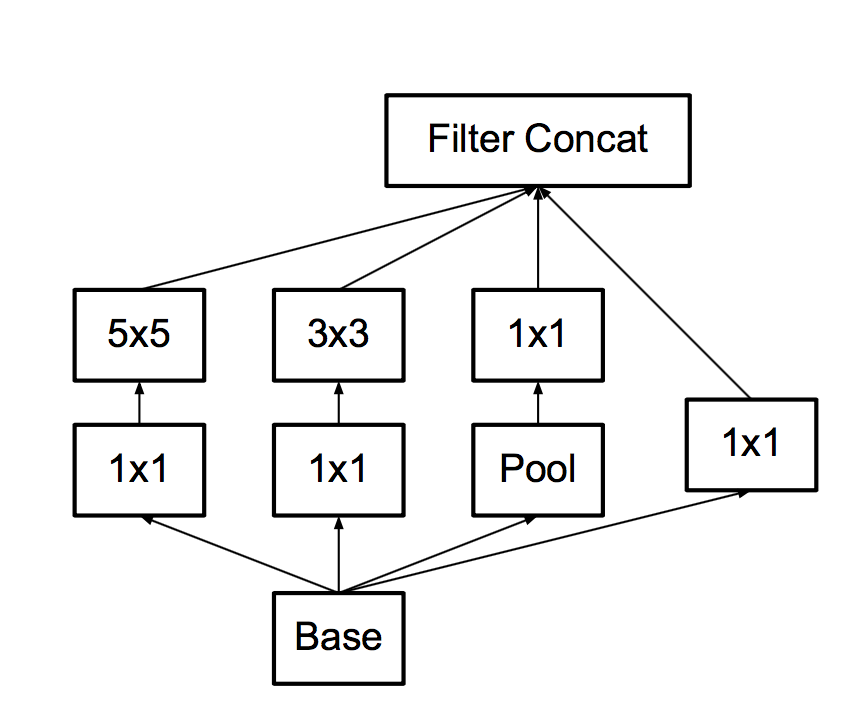


Рисунок 6 - Комбинация ядер в сверточном слое

Дальнейшее развитие идеи привело к новым идеям. Можно заменить большие ядра свертки размерности на два последовательных . Так как сигналы соседних пикселей сильно скорректированы, мы получим низкие потери при извлечении информации. Кроме того, можно разбить один сверточный слой

на два и - в эффективности потери низкие, однако получаем более легковесную нейронную сеть. Также применяются и и свертки.

Следующей проблемой решалось применения пулинга изображения, для уменьшения размеров картинки. Если делать пулинг до свертки, то мы получаем малое количество активаций после свертки, и как следствие, малое определение характеристик в окне свертки. Второй вариант, это использование пулинга после свертки. Однако, в нем основная проблема – это неэффективность вычислений. Для Inception был предложен подход производить свертку и пулинг одновременно.

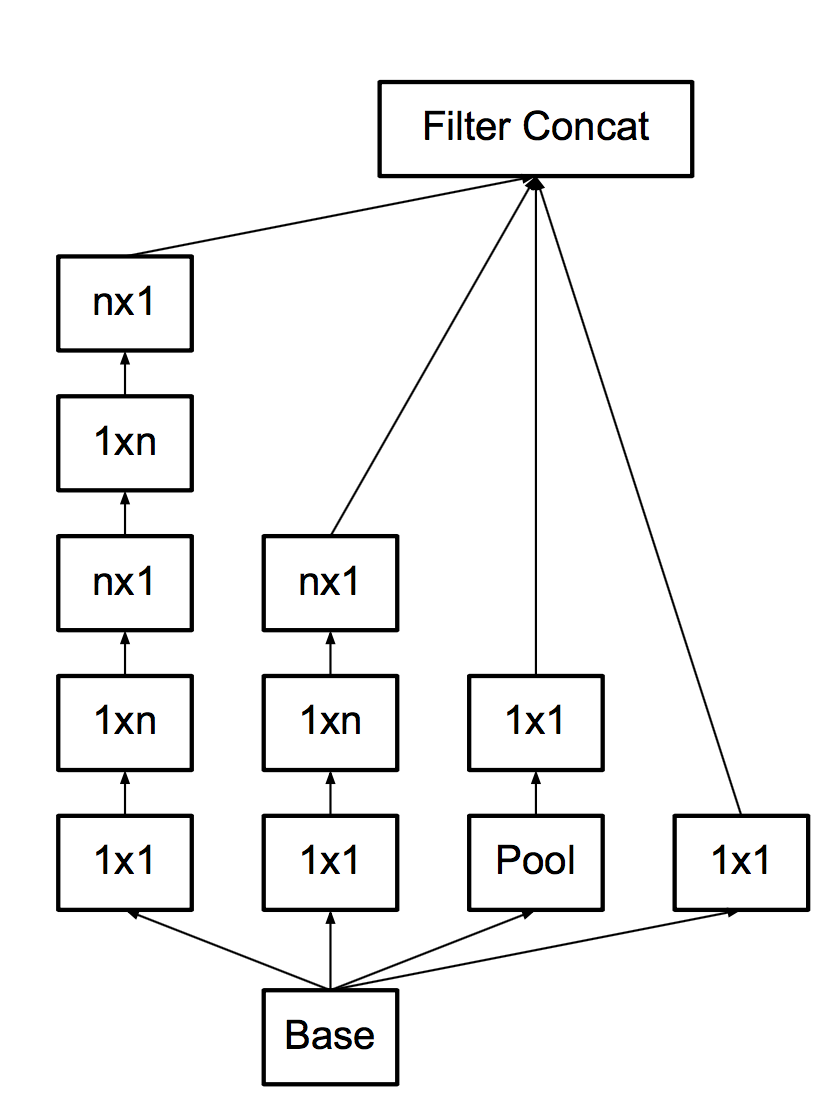


Рисунок 7 - Комбинация ядер малых размеров в сверточном слое Inception

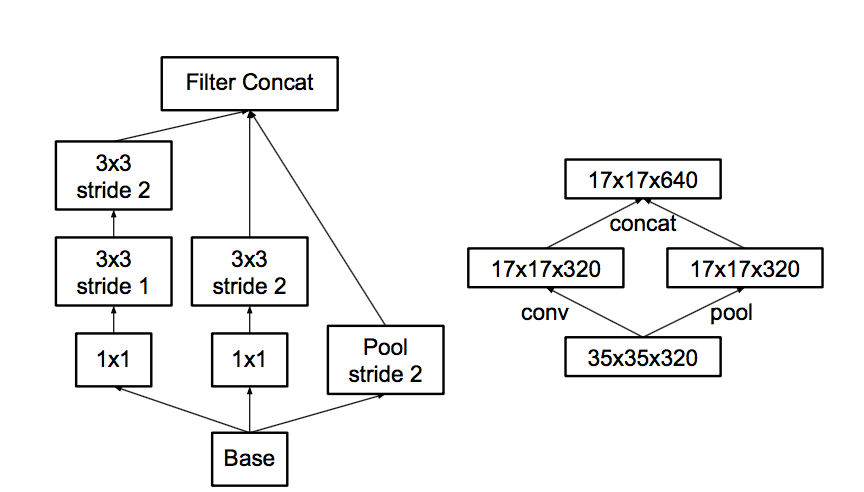


Рисунок 8 - Параллельная свертка и пулинг в слое Inception

Так как после прохождения аула количество характеристик уменьшается в два раза - перегрузки вычислений не будет. Пул сожмет предыдущий слой, не уменьшая при этом количество характеристик. Параллельно свертка будет прогнана в полном разрешении, но с набором характеристик поменьше. ИНС научится разделять, что требует полного разрешения, а для чего достаточно пула.

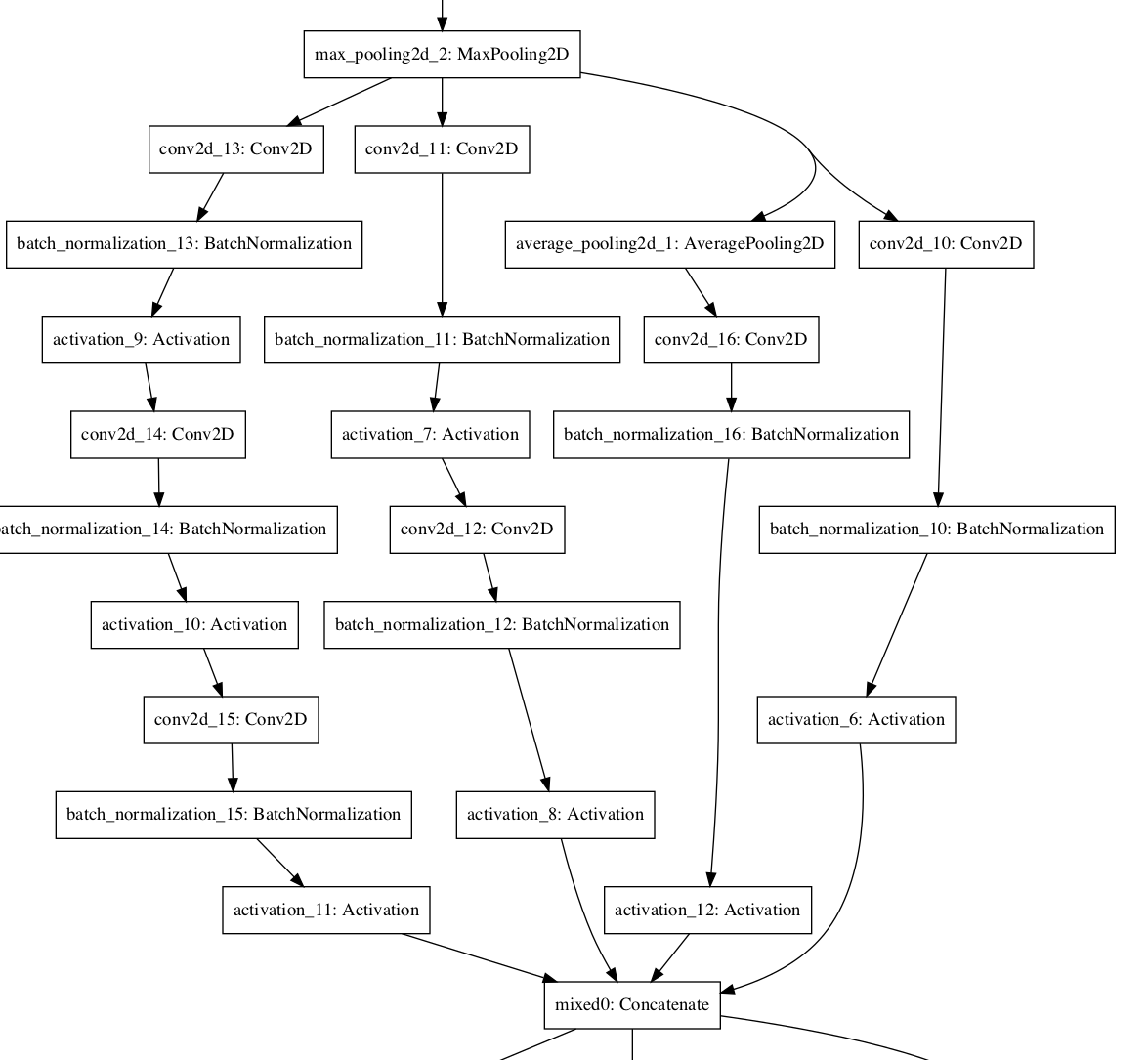


Рисунок 9 - Структура ядра Inception

* 1. Xception

Для улучшения качества распознавания своих сетей, исследователи старались добавлять больше слоев в разрабатываемые модели нейронных сетей, однако в большинстве случаев ограничения производительности, что не позволяло использовать очень глубокие нейронные сети.

В ходе развития так появилась архитектура Inception. Основой этой архитектуры является то, что при проектировании не выбираются, какие применять ядра свертки или и использовать ли max-pooling, а применяются все походы сразу. Это существенно увеличивает операции, которые необходимо для вычисления активации одного слоя. Для решения данной проблемы был предложен метод предварительного снижения входного слоя сверкой размера .

При дальнейшем развитии в 2016 году Франсуа Шолле [9] предложил использовать так называемый экстремальный Inception-модуль — Xception: Dephtwise separable convolution. К примеру обычный Inception сверточный слой выделения 32 признаков для изображения с 16 каналами с ядром размера будет иметь 16\*32\*3\*3 = 4608 весов.

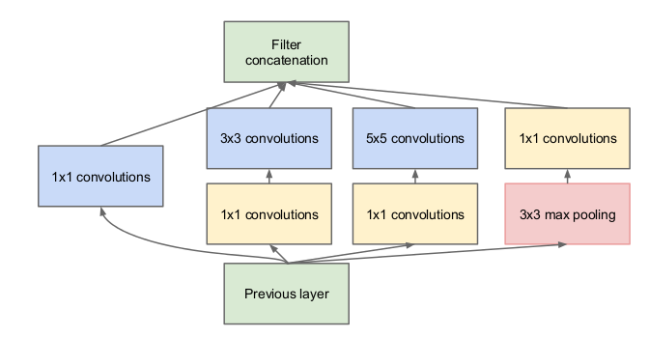


Рисунок 10 - Структура ядра Xception

**Архитектура** Xception основана на независимой свертке всех каналов изображения одним ядром. Так свертка разбивается на depthwise separale convolution блок размером и на pointwise convolution блок размером . Общее количество весов при этом уменьшается весов. Это много меньше, чем у стандартного сверхточного слоя Inception. Так, в отличие от обычного сверточного слоя, которая обрабатывает как пространственную информацию - корреляцию соседних точек внутри одного канала, так и межканальную, свертка применяется ко всем каналам сразу. Xception нейронная сеть основывается на предположении, что эти два вида информации можно обрабатывать последовательно, без потери качества. Разбивая обычную свертку на pointwise convolution - обрабатывает только межканальную связь, и на spatial convolution - обрабатывает пространственную корреляцию на каждом канале по отдельности.

* 1. ResNet50

С развитием глубоких и очень глубоких нейронных сетей появляются проблемы их обучения. Так при увеличении количества слоев в ИНС точность обученной модели сначала увеличивается, потом же резко ухудшается. Данное поведение прежде всего связано с тем, что нейронная сеть очень долго обучается, а при большом количестве слоев, ошибка обучения, уменьшаемая на каждом последующем слое, при обратном распространении ошибки, уже в конце концов имеет около нулевое значение, что никак не помогает обучению сети.

Одним из решений данной проблемы было предложено - добавление новых слоев ИНС по мере обучения. Так мы получаем достаточно обученную ИНС и повышаем точность с помощью добавления новых слоев. Однако, данная идея не нашла своей реализации, так как, не понятно, как инициализировать веса на новом слое. Если на новом слое инициализировать веса случайным образом, то вся обученная нейронная сеть, состоящая из предыдущих слоев будет работать неправильно, и ИНС будет обучаться заново. Если же инициализировать веса, так чтобы они пропускали просто сигнал, то никакого обучения не будет происходить, так как веса будут в большинстве между собой равны и равны 0, что приводит неэффективному обучению. Последний слой обучаться не будет.

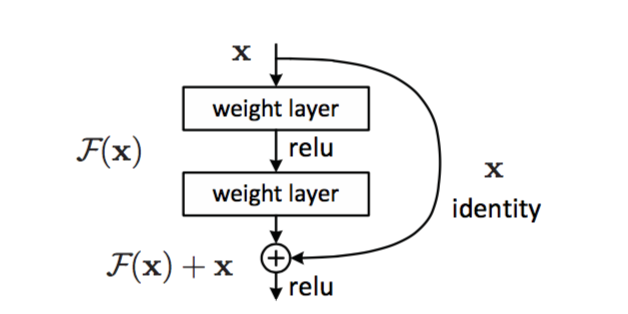


Рисунок 11 - Структура взаимодействие слоев в Resnet

Одним из решений данной проблемы занялась команда Microsoft [10], разработав архитектуру ИНС быстрого доступа.

Основная идея решения заключается в глубокой «остаточной» структуре обучения. Чтобы не ждать, что каждый слой будет непосредственно соответствовать желаемому результату, был реализован связной слой игнорирующий свертку и проходящий к следующему слою. Результаты свертки и входной сигнал свертки складываются на выходе для следующего слоя ИНС. Такой подход решает такие проблемы как:

* + - * + Быстрое обучение. Так ошибка обучения напрямую воздействуют на все слои ИНС, ускоряя при этом процесс обучения.
        + Высокая точность. Благодаря возможности обучать глубокие и очень глубокие ИНС появилась возможность беспрепятственно увеличивает глубину ИНС, тем самым увеличивая точность.
  1. MobileNet

Широкое развитие нейронный сетей и появление моделей AlexNet, VGG и др. в компьютерном зрении послужило увеличением точности. Однако эти сети являются в основе своей работы глубокими с большим количеством параметров и длительные по расчетам. Столь высокая точность в реальных задачах не так важна, как возможность работать в реальном времени на малопроизводительных устройствах. Многие реальные задачи как робототехника, системы управления робототехникой, автоматическое управление автомобилем востребованы в быстрых, работающих в реальном времени ИНС на слабых устройствах. Данные потребности способствовали появлению новой архитектуры MobileNets [11].

Особенностью архитектуры данной ИНС является отсутствие maxpooling слоев. Вместо них для снижения пространственной размерности используется свертка с шагом 2. Также для данной сети доступны два гиперпараметра - множитель ширины и - множитель глубины или множитель расширения нейронной сети. Множитель ширины задает количество каналов на каждом слое. Параметры задаются относительно нормально нейронной сети Xception. Так например при - количество каналов на каждом слое будет в 4 раза меньше, чем на нормальной нейронной сети. Множитель глубины описывает размер карт характеристик - feature map на каждом последующем слое (относительно работы Xception). Так если размер feature map будет в 2 раза меньше.

Настраивая данные характеристики, такой подход позволяет получить быструю нейронную сеть с достаточной точностью.

* 1. MobileNetV2

Дальнейшее развитие мобильных нейронных сетей привело к появлению новой нейронной сети MobileNetV2 [12]. Отличительной особенностью которого от первой версии является применение расширяющего сверточного блока - expansion convolution block или bottleneck convolution block.

**Архитектура** данной нейронной сети состоит из применения трех слоев:

* + - * 1. Сначала используется pointwise блок с большим набором параметров. Называем как расширяемый сверхточный блок. На входе он принимает тензор размерности , и генерирует на выходе тензор размера - где - новый гиперпараметр - уровень расширения. Рекомендуется задавать данный параметр в пределе . Для меньших сетей это как правило 5, а для более крупных рекомендуется 10. Этот слой создает отображение входного тенора в пространстве большей размерности. Такое отображение назвали «Цифровым многообразием»
        2. Следом идет depthwise свертка с ReLU6 [13] функцией активации. Данная реализация также схоже и с блоком MobileNetV1. На входе получаем , а на выходе тенор с размером - где - шаг свертки.
        3. В конце идет свертка с линейной функцией активации, понижающая число каналов. Использование цифрового многообразия на предыдущих слоях позволяет выделять и не терять полезную информацию при такой свертке. На выходе получаем тензор размера , где - количество каналов на выходе блока.

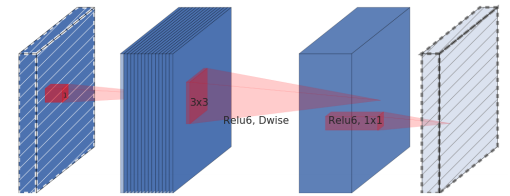


Рисунок 12 - Процесс свертки изображения нейронной сетью MobileNet

* 1. DenseNet

Нейронная сеть, предложенная Gao Huang и Zhuang Liu в 2018 году [14] Является прямым развитием ИНС ResNet.

Большой проблемой развития ИНС, в особенности глубоких ИНС является то, что путь ошибки при прохождении через слои при обучении уменьшается и может совсем исчезнуть. Глубокие и очень глубокие сети очень сложно обучать. Для того чтобы решить проблему увеличения глубины обучения без потери возможности ее обучать, в сетях ResNet использовался поход пробрасывания исходных значений предыдущих слоев на следующие, суммируя с результатом свертки.

Данный подход решал проблему обучаемости глубоких нейронных сетей, однако оставались проблемы смешения результатов свертки и исходного изображения. Это приводило к применению большого количества весов.

Подход Desnet отличается тем, что результат свертки на слое не суммируется с исходным значением характеристик, а объединяется, образуя большую feature map.

Это позволяет применять меньший набор каналов для карт характеристик изображения, и при этом не смешивать результаты сверток.

Вторым значительным изменением является использование принципа «коллективного знания», при котором каждый последующий слой знает все о предыдущих.

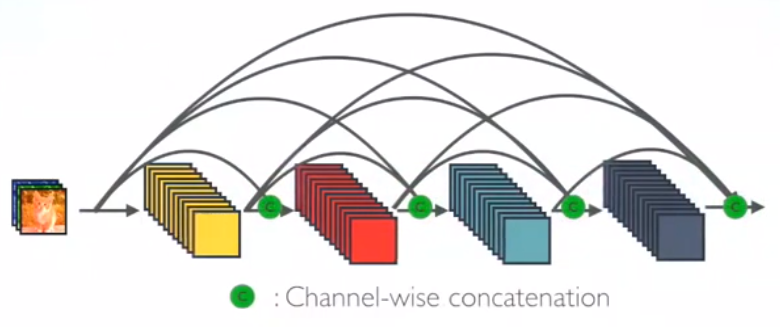


Рисунок 13 - Структура работы Desnet

При таком подходе мы передаем результаты работы сверхточного слоя сразу на все последующие слои.

* 1. NASNet

Нейронная сеть, использующая все подходы анализа изображения параллельно.

1. Распознание дорожных знаков на базе библиотеки tensorflow

Для работы с изображениями и обработки данных использовался язык разработки и инженерных расчетов python с использованием интерфейса Anaconda и PyCharm.

* 1. Общая схема работы системы

Исследование проводилось в три этапа: сбор и генерация набора данных, обучение модели нейронной сети, проверка результатов работы ИНС.

* 1. Сбор и предварительная обработка данных

Корректная работа нейросетевых моделей сильно зависит от данных. Для того, чтобы обученная нейронная сеть работала в реальных условиях, необходимо, чтобы данные, применяемые при обучении, были корректными, были корректно размечены и отображали все возможные допустимые отклонения для каждого класса объектов.

Рисунок 14 - Векторное изображение дорожного знака "Обгон запрещен"

Сбор данных производился из двух источников. GTSRB [15] - набор размеченных дорожных знаков, полученных фотографированием в реальных условиях (в наборе GTSRB - найдены ошибки. Так в тестовом наборе и тренировочном отличаются знаки под индексом 31). DorZnaki [16] - набор эталонов дорожных знаков в векторном формате.

Эталонные изображения были выбраны для поддержки всего набора дорожных знаков обученной моделью, а также для создания искажений, которых сложно добиться в реальных условиях.



Рисунок 15 - Дорожный знак "Обгон запрещен" из набора GTSRB

Предобработка данных состоит из частей:

* + - * Приведение к единому размеру - пикселей
      * Смещение дорожного знака с масштабированием в картинке
      * Изменение отношения ширины к длине знака с использованием поворота, для реализации разворота знака в пространстве
      * Затемнение и засветление
      * Наложение шумовых изображений поверх - реализация грязного окна
      * Выравнивание одинакового количества знаков в каждом наборе

Для наложения шумовых изображений был использован неразмеченный набор картинок openimages [17], и вставлялся с большой прозрачностью.

Рисунок 16 - Обработанное векторное изображения знака "Обгон запрещен"

Основным подходом в предварительной обработки изображения является применения сверки — скалярного перемножения исходного изображения я матрицу ядра свертки [18].

, где — исходное изображение, — матрица свертки изображения, — конечное изображение. Обычно преобразование изображения матрицей свертки еще можно представить, как фильтрация изображения, где ядро свертки — это цифровой фильтр.

Так для получения размытого изображения применен фильтр размытия изображения. Размер матрицы размытия влияет на силу размытия. Так перемножаясь с матрицей свертки пиксели исходного изображения усредняются со значением соседних пикселей.

Для повышения контрастности представлен в виде матрицы c выделением текущего пикселя, и усилением разницы между соседними. Следует отметить, что норма матрицы свертки должна быть равна 1.

Если же норма матрицы свертки больше 1 или меньше 1, то мы получим заселение, либо затемнение исходного изображения соответственно.

Для того, чтобы при обработки изображений не было заветов, при которых значения пикселей привыкаю допустимые значения [0,255], если у нас байт-представление цвета, применены методы обрезки максимальных значений пикселей.

**def** boxFilter(sourceImage, boxSize=3, boxScalar=**None**, padding=**"VALID"**):

*"""*

*box фильтр - размытие изображения*

**:param** *sourceImage: исходное изображение*

**:param** *boxScalar: множетель ядра фильтра - определяет яркость полученного зображения*

*если boxScalar = 1/boxSize\*\*2 - яркость не меняется*

**:param** *padding: исходное изображение*

**:return***: обработанное изображение*

*"""*

**if** (boxScalar **is None**):

boxScalar = 1. / boxSize \*\* 2

x\_image = tf.constant(np.asarray(sourceImage).astype(np.float32), dtype=tf.float32)

x\_image = tf.transpose(x\_image, [2, 0, 1])

x\_image = tf.reshape(x\_image, [n.value **for** n **in** x\_image.shape] + [1])

kernel = tf.constant(boxScalar \* np.asarray([[1] \* boxSize] \* boxSize), dtype=tf.float32)

kernel = tf.reshape(kernel, [boxSize, boxSize, 1, 1])

filtered = tf.nn.conv2d(x\_image, kernel, strides=[1, 1, 1, 1], padding=padding)

**with** tf.Session() **as** sess:

y\_image, = sess.run([filtered])

y\_image = y\_image.transpose((3, 1, 2, 0,))

y\_image = y\_image.reshape(y\_image.shape[1:])

*# обработка засветов*

y\_image = np.minimum(y\_image, 255)

y\_image = np.maximum(y\_image, 0)

newImage = Image.fromarray(y\_image.astype(np.uint8), sourceImage.mode)

**return** newImage

Набор, полученный из набора GTSRB обрабатывать не требуется, так как данные получены в реалистичных условиях и уже искажены.



Рисунок 17 - Набор изображений из набора GTSRB и их количество

* 1. Обучение нейронной сети

Для работы с изображением лучше всего подходят сверточные нейронные сети. В дипломной работе применены готовые модели сверточных сетей: Xception, ResNet, Inception, InceptionResNet, MobileNet, MobileNetv2, DenseNet.

Для взаимодействия нейронной сети с исходными данными, представление исходных картинок требуется представить в виде векторов, понятных для нейронной сети. Так, для выбранной схемы RGB изображения с размером должны иметь размерность , где 3 — это количество каналов для каждого отдельного цвета (красный, зеленый, синий). Вектора должны быть нормализованы — иметь значения в разбросе от -1 до 1. Кроме того, для ускорения процесса обучения и обработки набор изображений отправляется пачками, поэтому конечная размерность набора данных для обучения , где — размер набора данных для обучения.

model = keras.applications.xception.Xception(include\_top=**True**,

weights=**'imagenet' if** classesCount == 1000 **else None**,

input\_tensor=keras.Input(shape=inputSize),

input\_shape=**None**, pooling=**None**, classes=classesCount)

model.compile(loss=**"categorical\_crossentropy"**,

optimizer=**"adam"**, metrics=[**"accuracy"**])

Для обучения нейронных сетей был применён метод оптимизации Адам и использованиям градиентного спуска по кросс-энтропии. В коде представлен пример генерации нейронной сети на основе архитектур Xception и установка метода обучения Адам по точности и расчетом ошибки по энтропии ошибки точности.

* 1. Обучение на 28 классах

Сравним процесс обучение для нескольких моделей нейронных сетей. Основными критериями сравнения работы нейронной сети является скорость работы, качество классификации картинок по классам, ресурсоемкость.

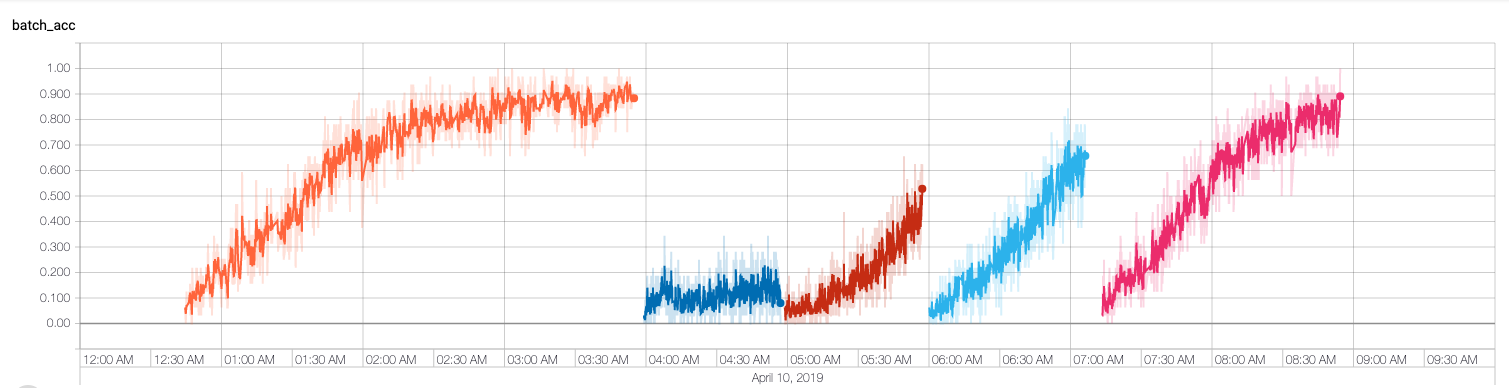


Рисунок 18 - Процесс обучения нейронных сетей

Последовательно запущенное обучение нейронных сетей на наборе 10000 пакетов по 5 итераций проводилось в общем счете около 7 часов. Так для обучения xception потребовалось 3 часа 10 минут, для inceptionv3 - 57 минут, для mobileNet и для mobileNetv2 примерно по 1 часу, а для denset - час и 40 минут.

Однако по графику показателя точности (Рис. 18) от процесса обучения нейронной сети видно, что за все время обучения для модели Inceptionv3 - этот показатель не растет. Также для моделей mobileNet и MobileNetv2, данный набор для обучения явно недостаточен — наблюдается резкий рост улучшения качества работы сети без дальнейшей стагнации.

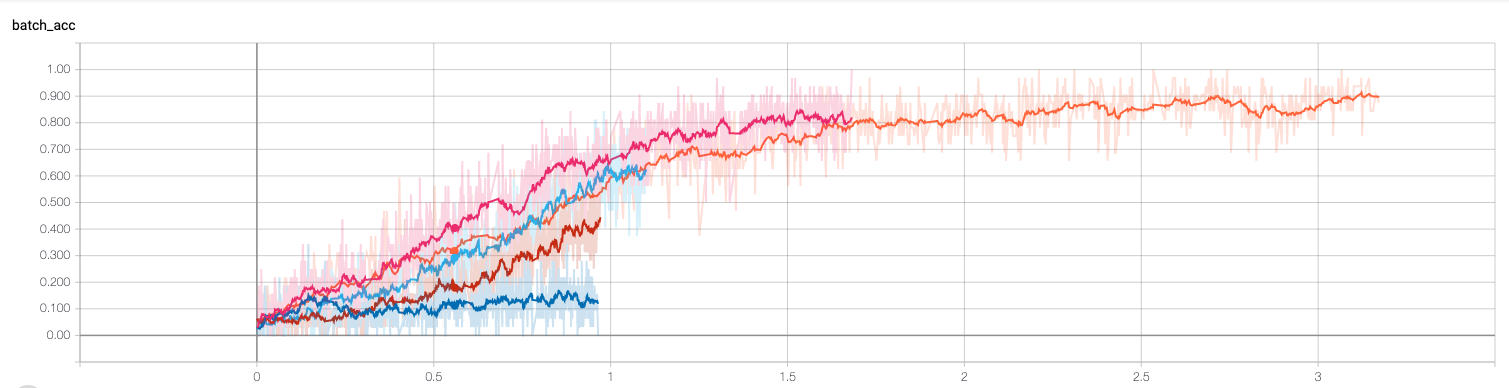


Рисунок 19 - Сравнение процессов обучения разных архитектур от времени

Сравним показатели точности работы нейронной сети между собой. Так из графика (Рис. 19) видно, что xception и desnet имеют лучшие показатели точности 0.875 и 1 соответственно. Однако для обучения xception требуется большее количество времени.

* + 1. Определение моделью xception

Total params: 2,293,852

Trainable params: 2,259,740

Non-trainable params: 34,112

Для обучения xception на 28 классах потребовалось создать модель с 2 миллионами параметров. Полученный размер данной модели, 252 Мб. Обучение заняло 3 часа 10 и по окончанию, точность работы данной модели приблизительно равна 0.89.



Рисунок 20 - Xception\_28. Обгон грузовым автомобилям запрещен

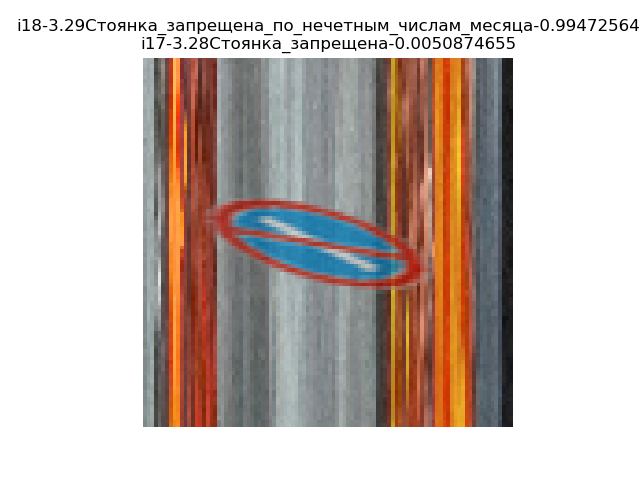
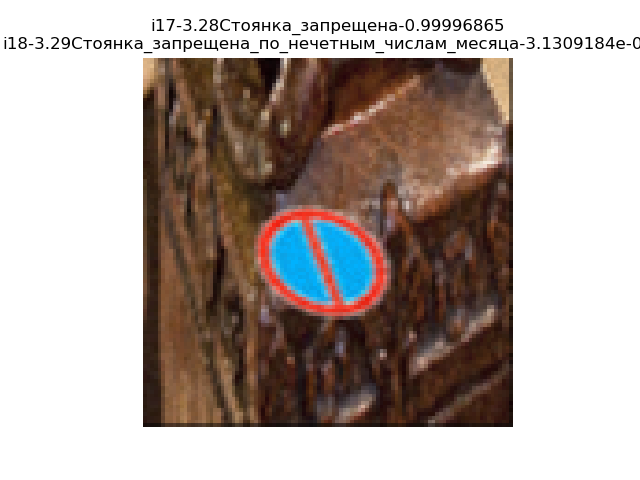


Рисунок 21 - Xception\_28. Распознавание знаков "Стояка запрещена по нечетным числам месяца" и "Стоянка запрещена"

По данным показателям можно сделать вывод, что нейронная сеть xce ption является высокоточной, однако ресурсной при расчетах. Так при обучении ИНС на наборе данных 100 тыс. картинок заняло около 3 часов, то на обработку одной может уйти примерно (3\*60\*60+10\*60+22)/(10000\*5) = 0,228 сек. или 228 мс. Учитывая, что данную модель планируется применять на мобильной платформе, на меньшем по мощности оборудовании, данные показатели не приемлемые.

Рассмотрим работу нейронной сети на примерах. На рис (000) видно, что нейронная сеть правильно и с высокой долей уверенности определила знак «Обгон грузовым автомобилям запрещен». Также следует отметить возможность xception уверено отличать схожие знаки как на рис. (000).



Рисунок 22 - Xception\_28. "Движение на велосипедах запрещено"

На следующем примере - рис (000) модель правильно определила зашумленный знак «Движения на велосипедах запрещено»

**Вывод**. В ходе обучения нейронной сети Xception на наборе картинок размером 100 на 100 пикселей, и количеством классов 28 классов показала хорошие качества распознавания знаков с точностью 0.89. Также данная нейронная сеть показывает высокие качества точности, при распознавании схожих знаков, таких как: “Стоянка запрещена” и “Стоянка по нечетным числам запрещена”. При этом скорость работы и обучения достаточно низкая, при обучении составила 3 часа и 10 мин, в расчете на распознавание знака на одном изображении потребуется 228 мс.

* + 1. Определение моделью Inceptionv3

Модели Xception и InceptionV3 являются схожими по строению и по размеру — ИНС Inceptionv3 вести 262 МБ. Однако после часового обучения нейросети InceptionV3 мы получили показатель точности равный 0.1.

Total params: 21,860,156

Trainable params: 21,825,724

Non-trainable params: 34,432

Данная модель имеет больший набор тренируемых параметров, около 21 млн. Связи с чем на обучение этой нейронной сети нужно много-больше времени.



Рисунок 23 - Inception\_28. Распознавание знака "Движение запрещено"

Сравним распознавание похожих знаков нейросети с низкой точностью.

Знак “Движение запрещено”, определено как “Движение транспортных средств с опасными грузами запрещено". Однако, уверенность предсказания всего 0.1. Также если посмотреть сравнить оригинальный с ошибочно предсказанным, можно найти схожие черты, как одинаковая красная окантовка знаков запрета.

Знак “Опасность” распознан как “Движение транспортных средств с опасными грузами запрещено”. При этом, уверенность предсказания также, как и предыдущем примере не больше 0.2. Сравнивая исходный знак и предсказанного, можно отметить общую красную окантовку, а также наличие контента, который может быть ошибочно определен, связи с низкой пикселизацией.

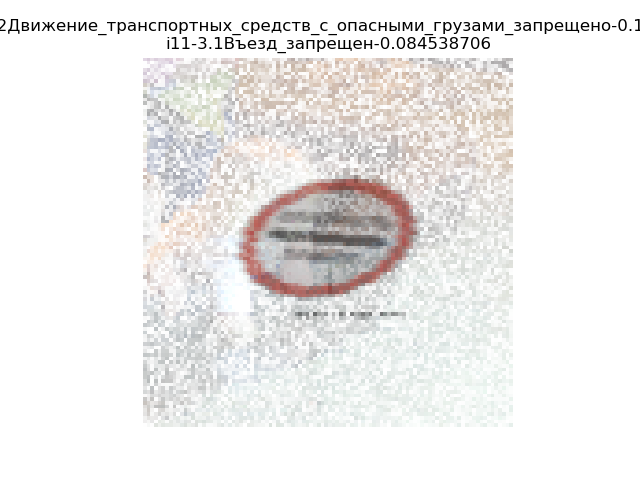


Рисунок 24 - Inception\_28. Распознавание знака "Опасность"



Рисунок 25 - Inception\_28. Распознавание знака "Ограничение массы"

Знак “Ограничение массы” определен как “Движение транспортных средств с опасными грузами запрещено”. Уверенность предсказания низкая и составляет всего 0.1. С оригинальным изображением общие черты – это красная окружность с белым фоном по центу.

Знак “Стоянка по нечетным числам запрещена” определен как “Таможня” с уверенностью равной 1. С оригинальным изображением, у предсказанного знака из общего только красная окантовка. Однако, сравнивая с предыдущими примерами распознавания знаков можно заметить, что для примеров выше предсказания нейронной сети смещались к одному значению “Движение транспортных средств с опасными грузами запрещено”. В этом же примере можно отличается фон – синий и предсказание совершенно другое. Можно понять, что для модели Inception большую роль в определениях знаков является цветовая гамма изображения.

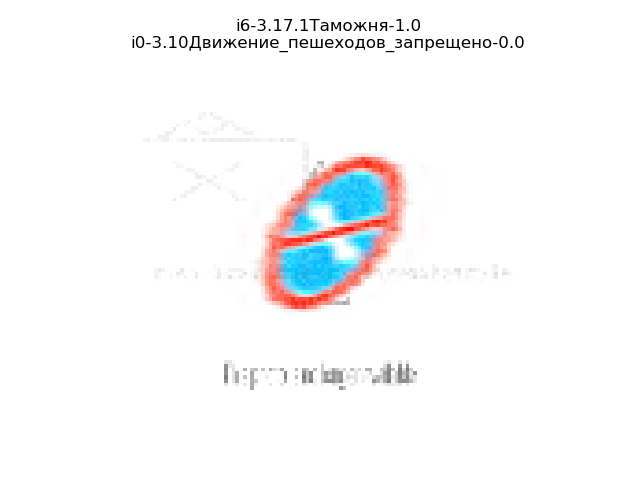


Рисунок 26 - Inception\_28. Распознавание знака "Стояка по нечетным числам запрещена"

Как видно из примеров, на начальном этапе для моделей характерно смещение результатов распознаваний к определенному классу. В нашем случае большинство изображений определилось как «Движение транспортных средств с опасными грузами запрещено». Можно сделать предположение, что нейронные сети обучаются в 3 этапа:

1. Этап получения первичных признаков изображений. Кластеризация изображений из набора по первичным признакам. Для данного этапа необязателен быстрый рост показателя точности работы НС.
2. Непосредственный этап обучения. Отличается резким возрастанием показателя точности ИНС.
3. Стагнация. Снижение скорости улучшения точности ИНС. Нейронная сеть достигает максимальной точности. Дальнейшее обучение может привести к переобучению.

**Вывод**. Обученная нейронная сеть, основанная на архитектуре InterceptiuonV3, показала низкие показатели точности определения дорожных знаков из набора 28 классов с размерами картинок 100 на 100 пикселей равные 0.1. Прежде всего это связанно с недостаточным обучением нейронной сети, в котором около 22 млн. тренируемых параметров при обучении. На первой стадии обучения ИНС наблюдается, в основном, низкая уверенность в предсказываемых результатах нейронной сети, однако для примеров, сильно отличающихся от основного набора возможен резкий всплеск роста уверенности, но при это не обязательно с правильным предсказанием. Также хотелось отметить, что большое влияние на определение изображений в interception является его цветовая гамма. С связи с этим можно предположить, что обученная нейронная сеть будет плохо работать для черно-белых изображений и затемненных изображений.

* + 1. Определение моделью mobileNet

Обучение мобильной ИНС заняло 1 час, и было преждевременно остановлено. Как показывают данные обучения, рост точности ИНС не перешел в стадию стагнации. После обучения точность составляет 0.48. Размер модели составляет 40 МБ.

Total params: 3,257,564

Trainable params: 3,235,676

Non-trainable params: 21,888

Тренируемых параметров 3 млн. Среднее время на обработку одного изображения (58\*60\*60+16\*60)/(10000\*5) = 0,06992 сек, или 69 мс. Данный показатель является достаточным для применения на мобильных устройствах. Так при максимальной скорости обновления кадров на устройстве равной 16 мс мы будем получать актуальные данные о расположении знака каждые 5 кадров.

Рассмотрим примеры работы на примерах.



Рисунок 27 - MobileNet\_28. “Остановка запрещена”



Рисунок 28 - MobileNet\_28. Распознавание знака "Контроль"

На данном примере знак “Остановка запрещена” определен с высокой уверенностью – 0.85. Причем знак 2го по уверенности предсказания – “Стоянка запрещена” также похож на оригинальный знак.

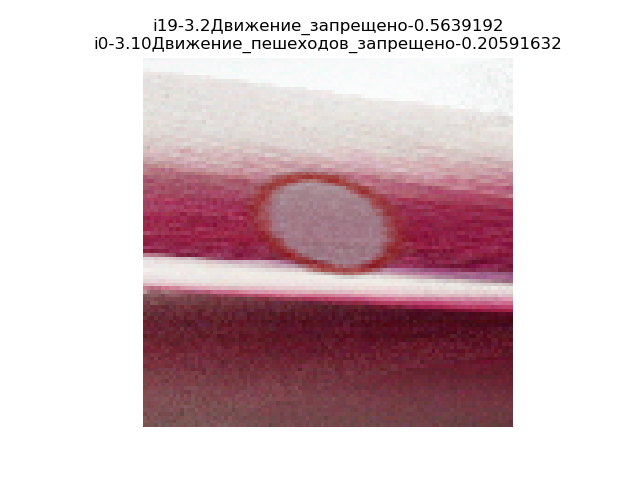
 Примере знак “Движение запрещено” определен правильно. Уверенность при этом составила 0.56. Что достаточно плохо. Со вторым предсказанным знаков, из общих черт можно отметить наличие красной окантовки. При это на второй знак показатель уверенности достигает 0.2. Та же можно отметить, что в наборе знаков запрета почти у всех есть общие черты – наличие красной окантовки и белый фон на знаке. Для при предсказании знаков будет немалое значение уверенности предсказания. Для верного предсказания понижается уверенность. Общая уверенность предсказаний нейронной сети понижается.



Рисунок 30 - MobileNet\_28. Распознавание знака "Обгон грузовым автомобилям запрещен"

Рисунок 29- MobileNet\_28. Распознавание знака "Движение запрещено"

Знак “Контроль” был ошибочно определен как “Разворот запрещен”. Уверенность составила 0.51. Второе ошибочное предсказание “Ограничение длины” – с уверенностью 0.2. Показатели уверенности низкие. Все знаки между собой похожи, однако, так как в наборе присутствует знак “Опасность”, который похож на знак контроль, ожидалось, что он тоже попадет набор из 2 лучших предсказанных знаков.

В примере знак был верно определен как “Обгон грузовым автомобилям запрещен” с показателем уверенности 0.42. Вторым предложенным знаком является “Остановка запрещена”.

**Вывод**. Основанная на архитектуре Xception нейронная сеть MobileNet показала низкие показатели точности определения дорожных знаков на наборе из 28 разных классов и равной 0.48. Однако, при рассмотрениях работы ИНС на конкретных примерах, наблюдается частое правильное определение анализируемого знака, причем с высокой уверенностью. В работе наблюдается завышение уверенности, при распознавании. За счет этого, можно сделать вывод о низкой точности распознавания. Скорость работы при распознавании одного изображения составляет 69 мс, а вес модели – 40 МБ. Данные показатели являются приемлемыми для использования на мобильной платформе.

* + 1. Определение моделью mobileNetv2

Модель modelNetv2 отличается от первой версии только наличием слоя «Цифрового разнообразия». Обучение данной модели заняло 1 час и 6 минут. Вес ИНС составляет 27 МБ. По завершении обучения, точность составляет 0.64.



Рисунок 31 - MobileNetV2\_28. Распознавание знака "Ограничение длины"

 Рассмотрим работу нейронной сети на примерах. Распознавание дорожного знака “Ограничение длины” произведено с высокой точностью и уверенностью 0.96. Знак при это имел низкие искажения, хорошо просматривается на фоне, не зашумлен.

Рисунок 32 - MobileNetV2\_28. Распознавание дорожного знака "Движение пешеходов запрещено"

Знак “Движение пешеходов запрещено” был определен неправильно как “Подача звукового сигнала запрещена”, причем с высокой уверенностью, равной 0.72. Изображение имеет небольшие искажения и имеет похожие очертания с неправильно определенным изображением – красная окантован с наклонным зачеркиванием.

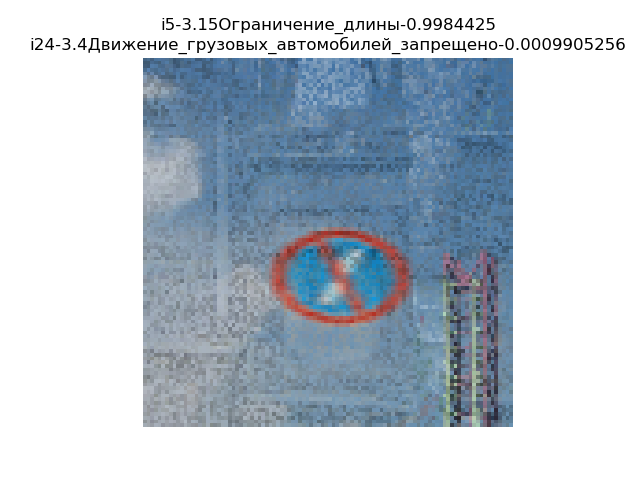


Рисунок 33 - MobileNet\_28. Распознавание знака "Ограничение длины"

Дорожный знак “Стоянка по нечетным числам запрещена” определен неправильно как “Ограничение длины”, причем с завышенной уверенностью 0.99.

Из общих черт определенного знака с исходным являются – одинаковая красная окантовка с перечеркиванием.

Знак “Обгон грузовым автомобилям запрещен” определен правильно, с высокой уверенностью 0.97. Изображение не искажено, четко выделяется на фоне.

**Вывод.** Нейронные сети MobileNet и MobileNetV2 показали достаточную низкую точность распознавания, для обоих равной 0.48 и 0.64 соответственно. Однако, при рассмотрении примеров наблюдаются определение знаков с высоким показателем уверенности, в том числе и при неправильном распознавании. Скорость распознавания одного изображения составляет 79 мс. Вес модели составляет 27 МБ. Данную нейронную сеть можно применять в мобильной платформе.



Рисунок 34 - MobileNetV2\_28. Распознавание знака "Обгон грузовым автомобилям запрещен"

* + 1. Определение моделью DenseNet

Обучения заняло 1 час и 40 минут. Размер модели 86 МБ. По завершении обучения точность составила 0.84.

Total params: 7,066,204

Trainable params: 6,982,556

Non-trainable params: 83,648

Тренируемых параметров около 7 млн. Благодаря тому, что в desnet используются прямые связи через все слои нейронной сети, обучение глубокой происходит сети происходит быстро.



Рисунок 35 - DenseNet\_28. Распознавание знака "Стоянка запрещена"

Так при распознавании знака “Стоянка запрещена” – нейронная сеть отработала правильно, с высокой уверенностью, равностью 0.78. Фоновое изображение имеет сложную структуру и может усложнить определение значения знака.

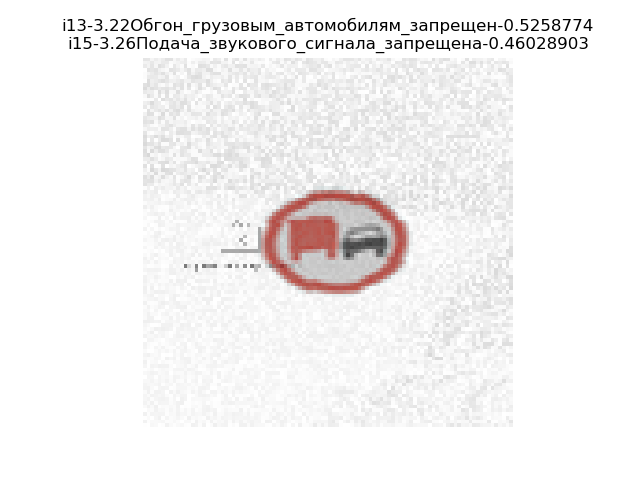


Рисунок 36 - DenseNet\_28. "Обгон грузовым автомобилям запрещен"

Знак “Обгон грузовым автомобилям запрещен” определен правильно. Показатель уверенности составляет низкие 0.52. При этом, на 2 предсказанный знак уверенность составляет 0.42.

Сам же знак “Подача звукового сигнала запрещена” определяется правильно и с высокой уверенностью в 0.96.

 Знак “Контроль” определен правильно, с уверенностью 0.99. Изображение при этом зашумлено, фон имеет сложную структуру.

Рисунок 37 - DenseNet\_28. Распознавание знака "Подача звукового сигнала запрещена"



Рисунок 38 - DenseNet\_28. Распознавание знака "Контроль"

**Вывод**. Нейронная сеть Desnet, обученная на наборе 28000 изображений с 28 классами, показал высокую точность распознавания, равную 0.84. Однако по проведенным сравнениям реальных результатов работы ИНС можно предположить о переобучении ИНС. В работе не использовался подход проверки работы обученной модели на данных, не используемых в обучении. Однако, по результатам тестирования наблюдается низкая уверенность в определении знака “Обгон грузовым автомобилям запрещен” с смещением (как далее выяснилось в большую по количеству примеров) на предсказание знака “Подача звукового сигнала запрещена”. Вес нейронной сети составляет 84 МБ, а скорость распознавания одного знака – 121 мс. При данных показателях, применения данной модели на мобильной платформе может быть проблематичным.

* 1. Обучение на 51 классе

Связи с тем, что по результатам, проведенных экспериментов выше, для некоторых архитектурных решений мы получили недообученную, следующий эксперимент проводился в 2 этапа. Общее количество примеров обучения для каждой модели составило 200 тыс. шт. А общее время обучение заняло 9 часов и 51 мин на первом этапе, 80 часов и 14 мин на втором.

Так на первом этапе можно заметить, что наиболее быстро развивающимся сетями являются desnet и xception. Только для xception на единицу прохождения итерации обучения требуется больше времени.

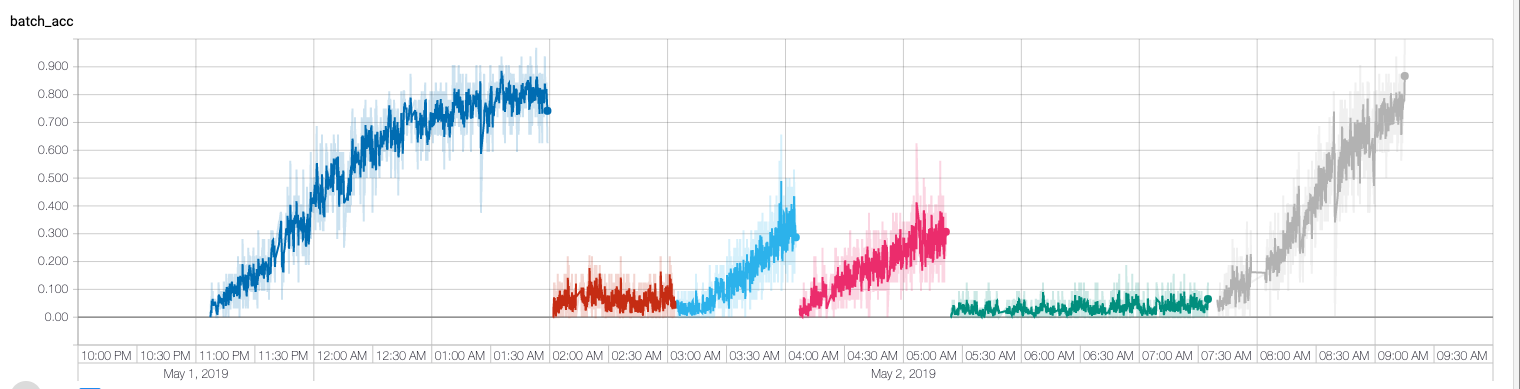


Рисунок 39 - Процесс обучения нейронных сетей на первом этапе

Также на модели, Xception наблюдается процесс стагнации, за счет чего, можно делать вывод о завершении обучения данной модели еще на первом этапе.

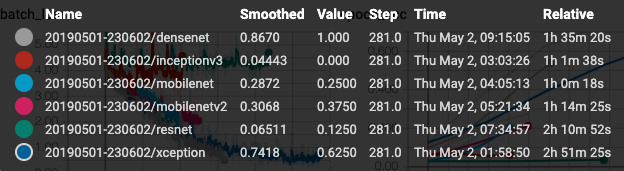
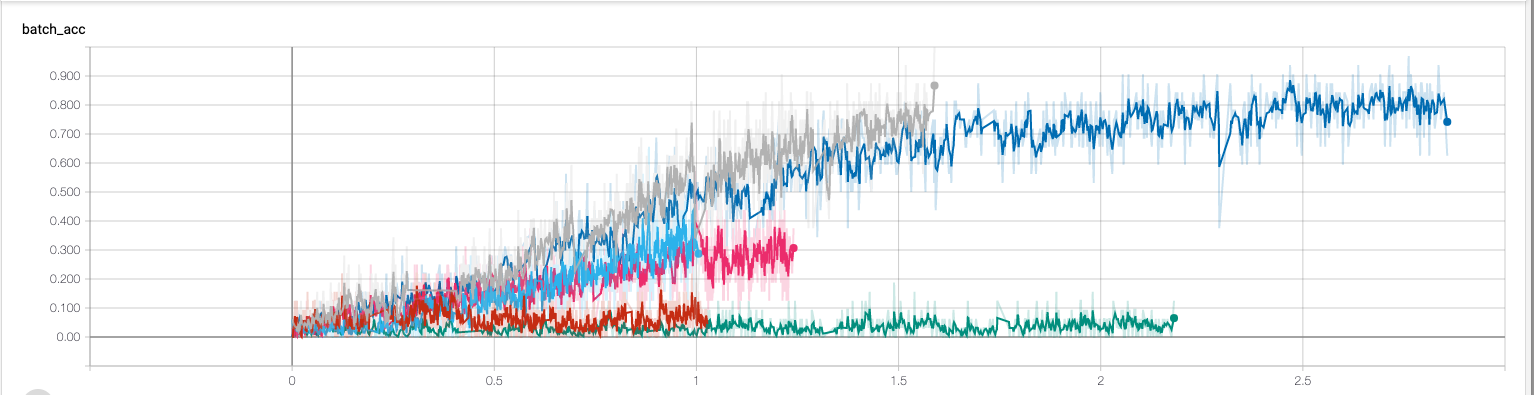


Рисунок 40 - Сравнение процессов обучения нейронный сетей на первом этапе

На втором же этапе мы можем заметить резкое улучшение качества работы для mobileNet и mobileNetV2.

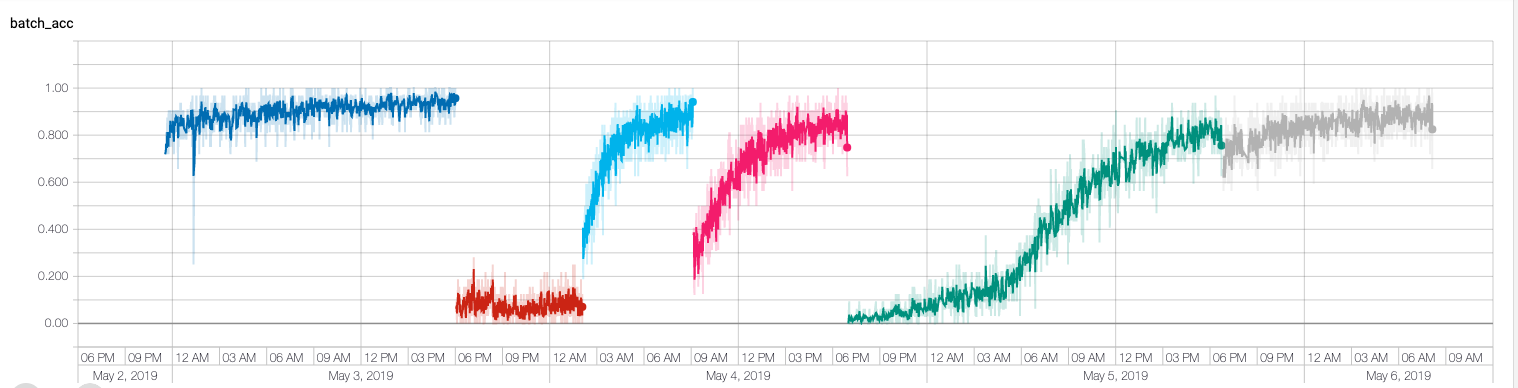


Рисунок 41 - Процесс обучения разных архитектур на 2 этапе

Также можно заметить, что точность mobileNetV2 достигла 1, а Desnet уменьшилась, по сравнению с первым этапом. Поэтому будем сравнивать усреднение значения точности для каждой из моделей.

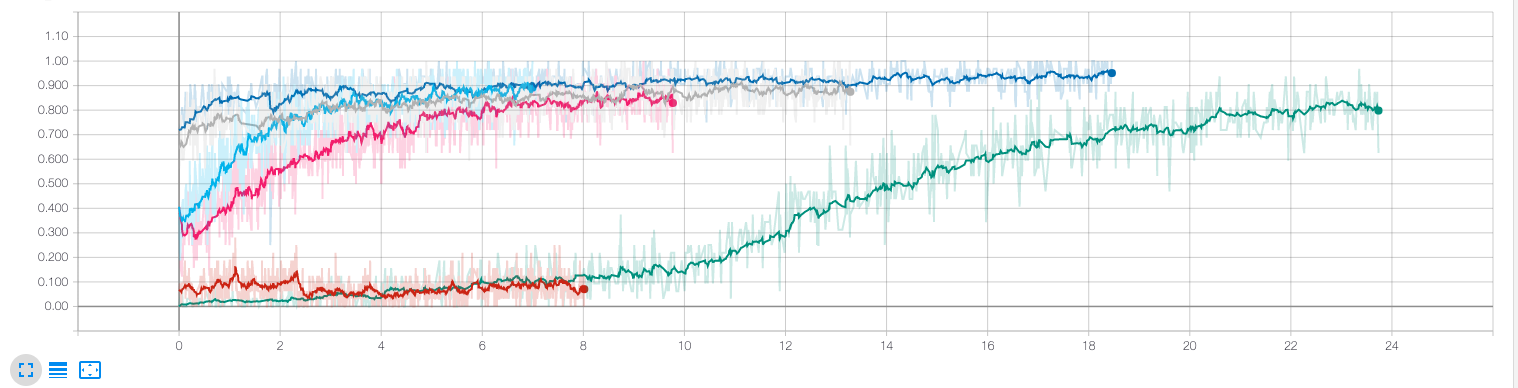


Рисунок 42 - Сравнение процессов обучения нейронных сетей на 2 этапе

Так наилучшими моделями за время обучения Xception, Densenet, MobileNetV2.

* + 1. Распознавание Xception

Так как архитектура xception применяет все возможные ядра сверки с размерами от , и обучение данной модели должно быть быстрым. В тоже время, так как в архитектуре применяйся большие сверточные матрицы — процесс обучения будет длительным. В нашем случае общая длительность обучения модели заняла около 21 часа. Причем основная часть обучения прошла на первом этапе в первые 3 часа.

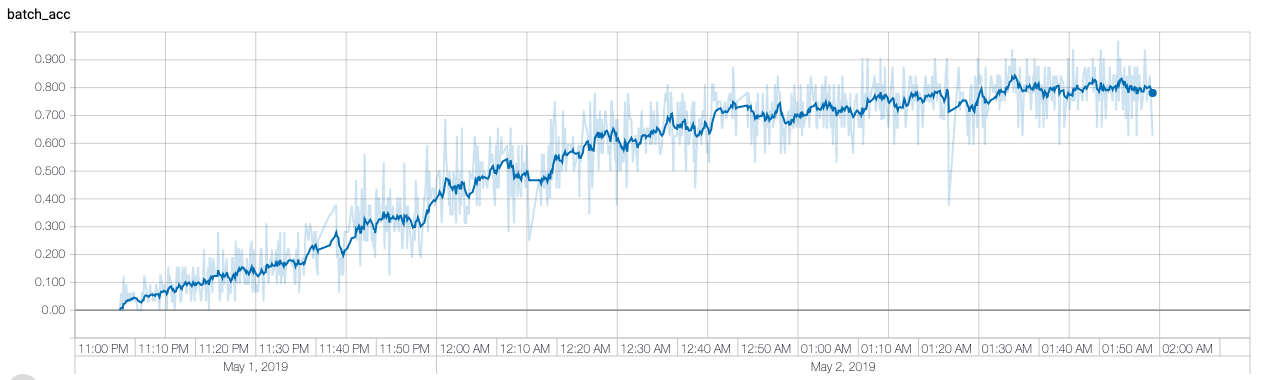


Рисунок 43 - Процесс обучения нейронной сети Xception

Обученная нейронная сеть распознает с точностью 0.9511. Так на примере распознавания дорожного знака “Конец ограничения минимальной скорости” обученная нейронная сеть распознала верно с точностью 0.99. Знак при этом был уменьшен на изображении.

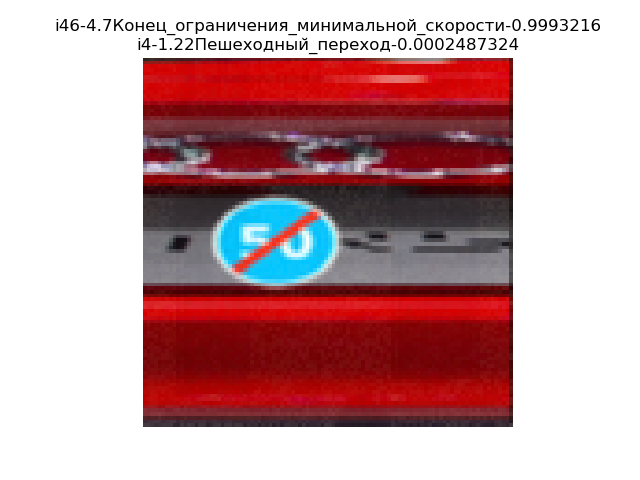


Рисунок 44 - Xception\_51. Распознавание знака "Конец ограничениея минимальной скорости"

Знак “Движение с прицепом запрещено” модель распознала с показателем уверенности 0.99. Знак уменьшен на фоне, мало искажен.



Рисунок 45 - Xception\_51. Распознавание знака "Движение с прицепом запрещено"

“Сужение дороги” на изображении сильно искажен. К изображению применен дополнительный слой зашумления. Нейронная сеть верно распознала знак, уверенность при распознавании составила 0.99.



Рисунок 46 - Xception\_51. Распознавание знака "Сужение дороги"

Знак “Въезд запрещен” определен правильно с показателем уверенности 0.99.

**Вывод.** В ходе обучения нейронной сети Xception на 200 тыс. наборе из 51 знаков нейронная сеть показала точность, равную 0.95. Что является очень хорошим показателем. Однако в процесс обучения потребовал 21 часа. Так, при распознавании одного знака на мобильном устройстве потребуется 383 мс. При максимальной скорости обновлении кадров 16мс, актуальные данные о распознанных знаках будут предоставляется каждые 24 кадра. Вес обученной ИНС составляет 240 МБ. Данные показатели не приемлемы при развертывании на мобильном устройстве.



Рисунок 47 - Xception\_51. Распознавание знака "Въезд запрещен"

* + 1. Распознавание resnet

Так как resnet глубокая нейронная сеть с применением сквозных связей через слои. Обучение сопровождается достаточно длительным начальным процессом формирования связей между нейронами.

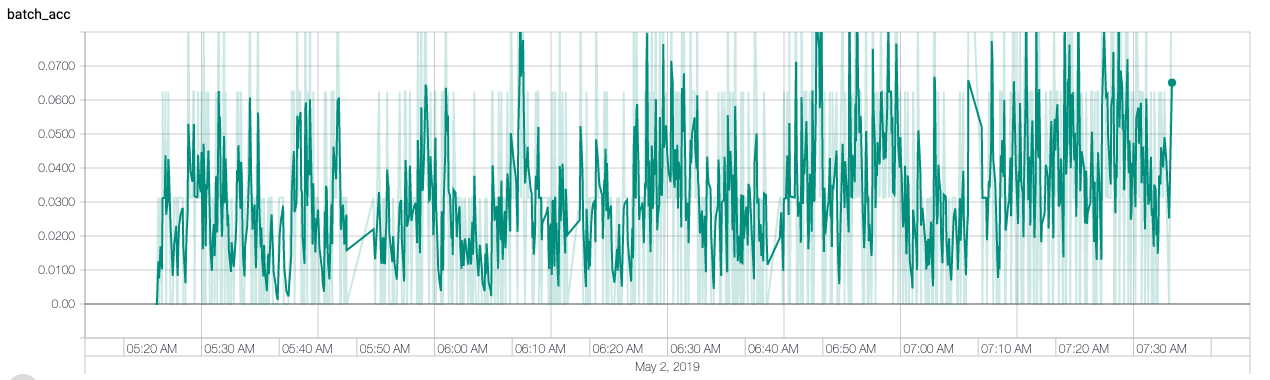


Рисунок 48- Процесс обучения нейронной сети Resnet на 1 этапе

Так для обучения resnet первой стадии длительностью 2 часа не хватило, и основной процесс пришелся на 2 часть и занял по времени 24 часа.

На данном рисунке (00) хорошо видно, что весь процесс обучения resnet состоит из 3 условных частей. Причем первая часть отличается очень вялым развитием, что может говорить о длительном выходе нейронной сети из локального минимума, либо касательно сети resnet это может быть плохая инициализация весов сквозных связей. Для большей скорости обучения всех слоев ИНС, требуется, чтобы эти связи были инициализированы ближе к единице.

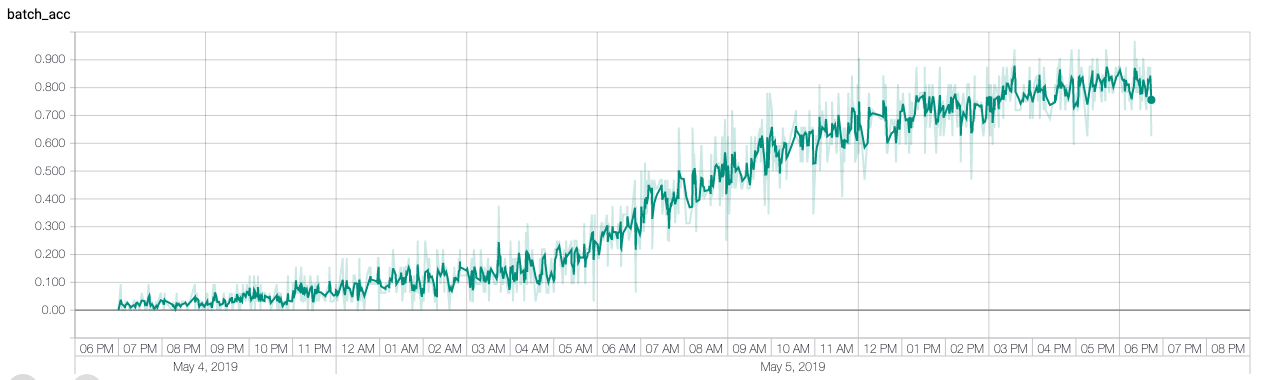


Рисунок 49 - Процесс обучения нейронной сети resnet на 2 этапе

После всего процесса модель resnet показывает низкую точность распознавания 0.8.

Так дорожный знак “Ограничение длины” был определен, как “Выезд на дорогу с полосой для маршрутных транспортных средств”. Уверенность при этом составила примерно 0.26. Следует отметить, что для данного примера применено достаточное большое искажение исходного изображения, что может влиять на конечных результат распознавания.



Рисунок 50 - Resnet\_51. Распознавание дорожного знака "Ограничение длины"

Дорожный знак “Движение мотоциклов запрещено” определился, как “Таможня”. Причем уверенность довольно высокая и равна 0.82. Изображение отчетливое, с небольшими искажениями.



Рисунок 51 - Resnet\_51. Распознавание дорожного знака "Движение мотоциклов запрещено"

При этом, основываясь на предыдущих тестированиях нейронных сетей, на данном наборе, можно сделать вывод, что мы стабильно получаем ошибку смещения, которую можно получить при неправильном формировании обучающий выборки. Так, в нашем случае, мы стабильно получаем смещение обучения на классы: “Таможня”, “Выезд на дорогу с полосой для маршрутных транспортных средств”

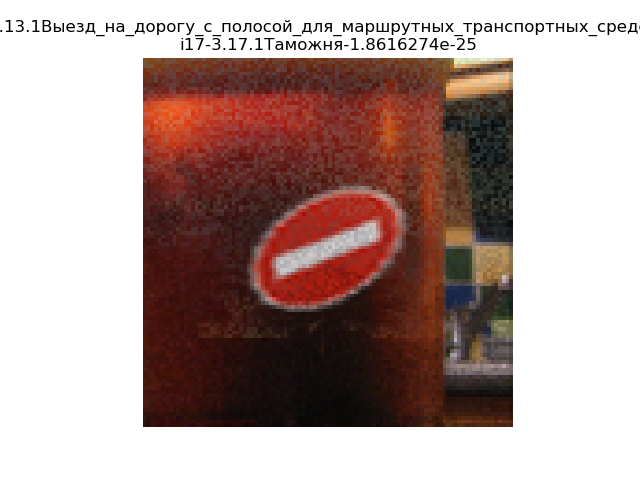


Рисунок 52 - Resnet\_51. Распознавание дорожного знака "Выезд запрещен"

Знак “Выезд запрещен”, был определен как “Выезд на дорогу с полосой для маршрутных транспортных средств”. Уверенность при этом составила 0.36. Такие показатели являются неожиданными, учитывая тот момент, что цветовая гамма для знаков разная.

Однако, это не удивительно, так как resnet использует в основе архитектуру Xception, по факту может анализировать текст в условно черно-белом спектре.

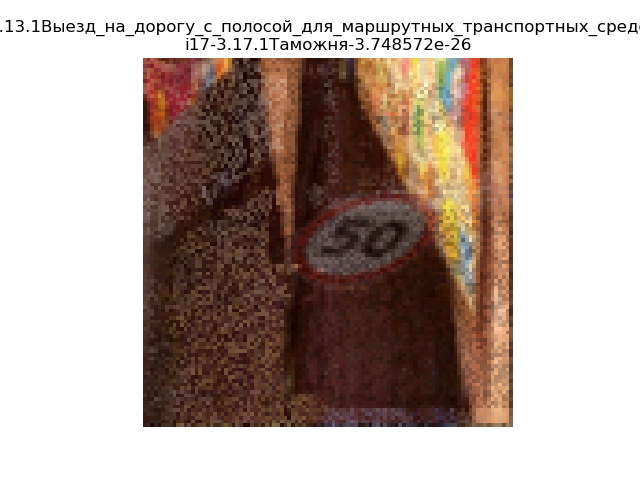


Рисунок 53 - Resnet\_51. Распознавание знака "Ограничение максимальной скорости"

Знак “Ограничение максимальной скорости” определен, как Выезд на дорогу с полосой для маршрутных транспортных средств. Уверенность определения составляет 0.2. При этом, следует отметить, что изображение сильно зашумлено, а сам знак сильно уменьшен.

В **выводе** хотел бы отметить, что по окончании обучения, занявшего по времени 24 часа. Нейронная сеть показала достаточно высокий показатель точности, равный 0.65. Однако, рассмотрев конкретные примеры работы нейронной сети, можно заключить, что данная модель выигрывает только за счет того, что выдает низкие показатели точности для ошибочных результатов, а не за счет того, что результаты в большинстве правильные. В ходе обучения наблюдается частое смещение результатов предсказания к одному классу. Прежде всего это связано с большим преобладанием количества примеров из набора конкретного класса над другими. Для решения данной проблемы существуют подходы для равномерного обучения: снижения коэффициента при обучении на конкретном классе, уравнивание наборов обучения по классам, при подготовке тренируемых наборов данных. Размер обученной нейронной сети составил 284.5 МБ. Время обработки одного изображения составляет 446 мс.

* + 1. Работа mobileNet

Общая длительность обучения составила 8 часов.

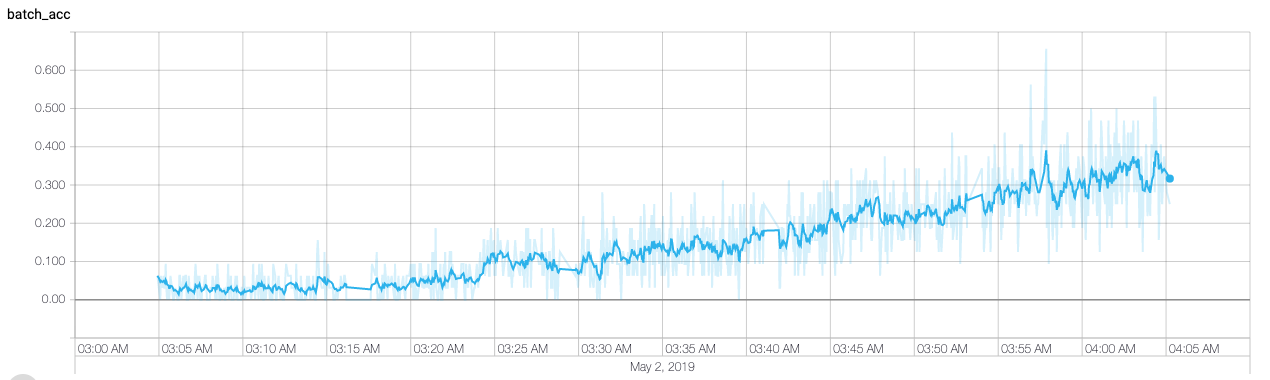


Рисунок 54 - Процесс обучения нейронной сети MobileNet на 1 этапе

На первом этапе наблюдается длительный процесс первичной инициализации весов. Далее происходит ускоряющийся процесс обучения с большими локальными скачками. Это может отображать процесс выхода «выпрыгиванием» из локальных минимумов.

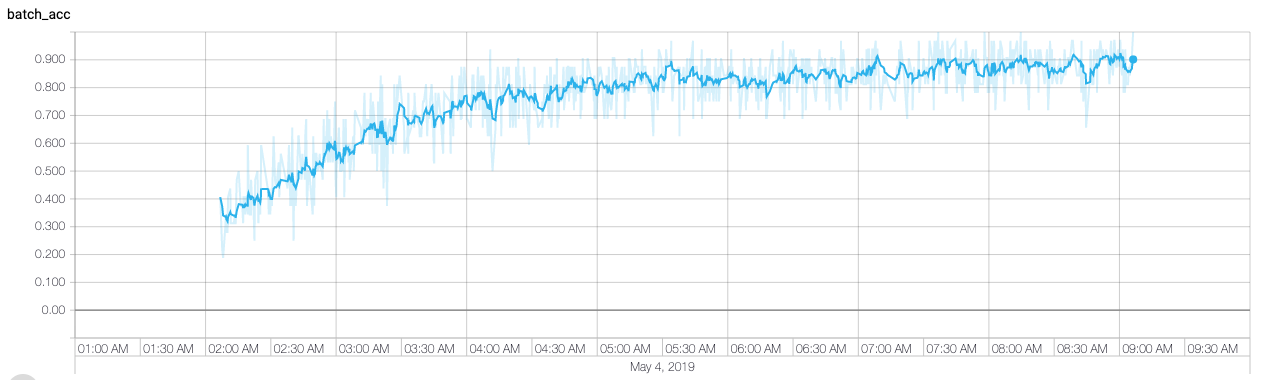


Рисунок 55 - Процесс обучения нейронной сети MobileNet на 2 этапе

Второй же этап отображает продолжение обучение и продолжительную стагнацию, по которой можно заключить, что процесс обучения достиг максимума. Точность по завершении обучения равна 0.9.

Так на рисунке выше можно отметить неправильное определение засверленного знака «Конец запрещения обгона». Однако ИНС спутала со схожим знаком «Конец запрета обгона грузовым автомобилем», и при этом выдала низкую уверенность. Следует отметить, что также похожий знак «Конец зоны всех ограничений» не попал в этот список.



Рисунок 56 - MobileNet\_51. Распознавание дорожного знака "Конец запрещения обгона"



Рисунок 57 - MobileNet\_51. Распознавание знака "Въезд запрещен"

Дорожный знак “Въезд запрещен”, распознан правильно, с уверенностью, равной 0.99. Знак при этом быль сильно искажен.

Также дорожный знак “Поворот направо запрещен” определен правильно. Уверенность при распознавании составила 0.99.

 Дорожный знак “Обгон запрещен” распознан правильно. Уверенность составляет при этом 0.99. Изображение сильно искажено и зашумлено. Фон имеет сложную структуру.

Рисунок 58 - MobileNet\_51. Распознавание дорожного знака "Поворот направо запрещен"



Рисунок 59 - MobileNet\_51. "Обгон запрещен"

**Вывод.** Нейронная сеть mobileNet, обученная на наборе из 51 различных дорожных знаков, показала точность распознавания, равную 0.89. При проверке работы на отдельных примерах, наблюдается высокая уверенность определения значения знака и высокая частота угадываний. Скорость работы при этом, составляет 144 мс. А вес 40 МБ. Что является приемлемым при применении на мобильной платформе.

* + 1. Распознавание mobileNetV2

Так как mobileNetV2 схож с mobileNet, процесс обучения такой же как и у первой версии.

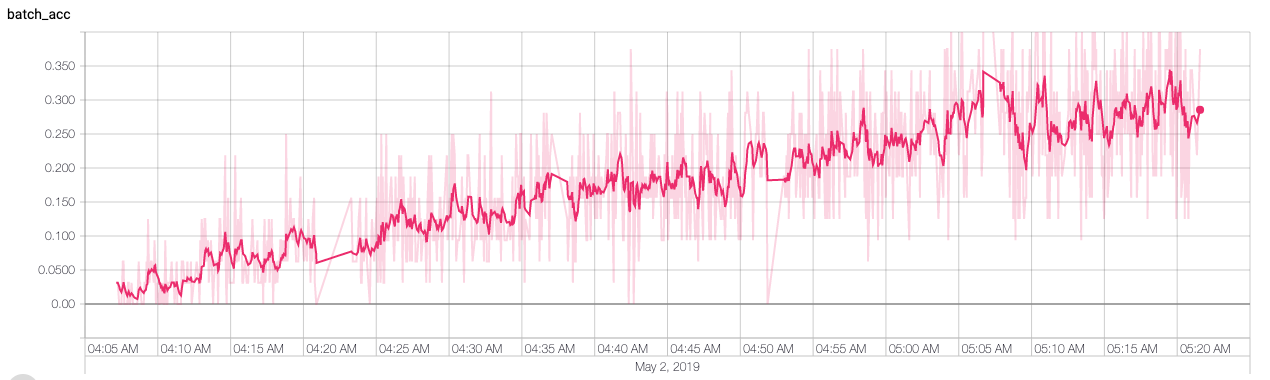


Рисунок 60 - Процесс обучения нейронной сети MobileNetV2 на 1 этапе

Однако, в отличие от первой версии, длительного процесса выхода из локального минимума на первом этапе обучения не наблюдается.

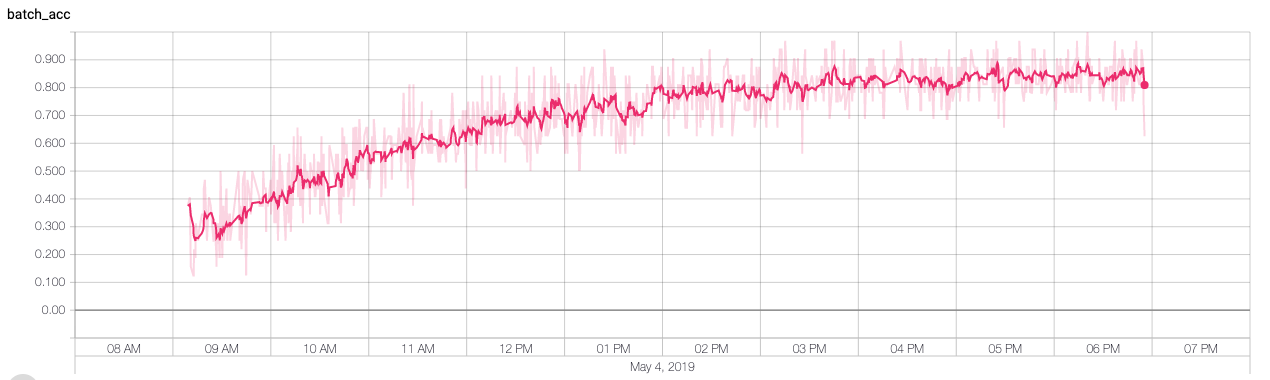
 Весь процесс обучения занял 11 часов. Точность распознавания достигла 0.83.

Рисунок 61- Процесс обучения нейронной сети MobileNetV2 на 2 этапе



Рисунок 62 - MobileNetV2\_51. Распознание знака "Конец дороги для автомобилей"

Рассмотрим процесс распознавания на примерах.

Знак “Движение транспортных средств с опасными грузами запрещено”, с уверенностью 0.8. Изображение искажено.



Рисунок 63 - MobileNetV2\_51. "Движение транспортных средст с опасными грузами запрещено"

Знак “Конец дороги для автомобилей” распознан с показателем уверенности 0.73. Фон имеет сложную структуру, изображение зашумлено.



Рисунок 64 - MobileNetV2\_51. Распознание дорожного знака "Ограничение минимальной дистанции"

“Ограничение минимальной дистанции” распознан с показателем уверенности 0.99. Фон имеет сложную структуру, знак искажен.

Дорожный знак “Искусственная неровность” определен неправильно, как знак “Движение транспортных средств с взрывчатыми и легковоспламеняющимися грузами запрещено”, с показателем уверенности 0.5

 Если сравнить процессы обучения этих двух сетей. То вторая версия отличается более пологим подъемом, прямое следствие появление дополнительного слоя.

Рисунок 65 - MobileNetV2\_51. Распознавание дорожного знака "Искусственная неровность"

Если сравнивать процесс обучения mobileNet с mobileNetV2, процесс 2й модели происходит быстрее, а общее время занимает большее время. Но окончанию завершения обучения точность mobileNetV2 несколько меньше в сравнении с mobileNet – 0.83 против 0.89.

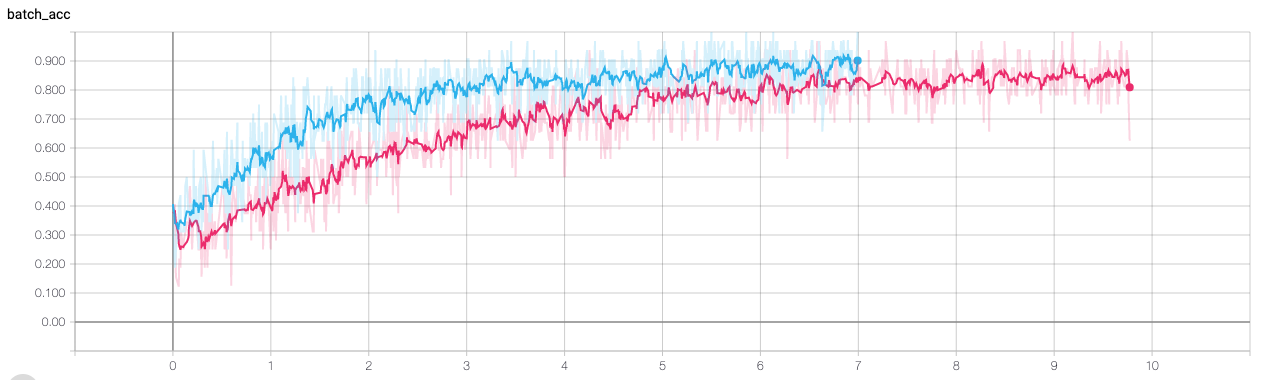


Рисунок 66 - Сравнение процессов обучения MobileNet и MobileNetV2

**Вывод.** Обученная нейронная сеть mobileNet2 показала высокую точность распознавания знаков. Для обучения модели на наборе из 200 тысяч картинок потребовалось примерно 11 часов. Так для обработки одного изображения требуется 198 мс. Весь модели 27 МБ. Применение на мобильном устройстве не рекомендуется.

* + 1. Распознавание сети DenseNet

Обучение нейронной сети, основанной на архитектуре desnet, заняло 14 часов.

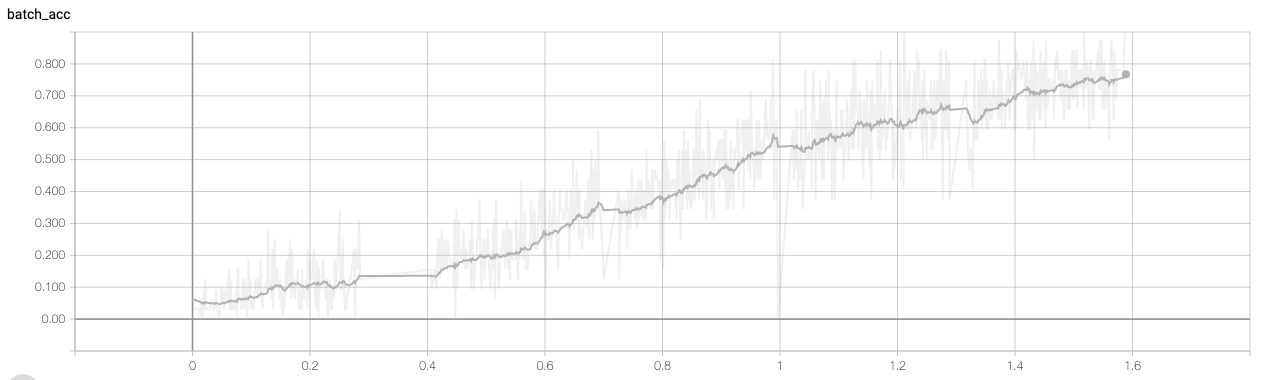


Рисунок 67 - Процесс обучения нейронной сети desenet на 1 этапе

Обученная нейронная сеть показывает точность 0.87. При чем, точность в 0.86 была достигнута еще на первом этапе, в первые 2 часа обучения.

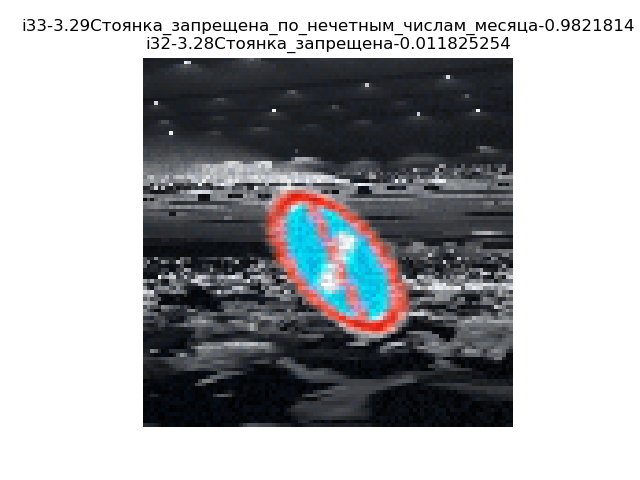


Рисунок 68 - DenseNet\_51. Распознавание знака "Стоянка запрещена по нечетным числам месяца"

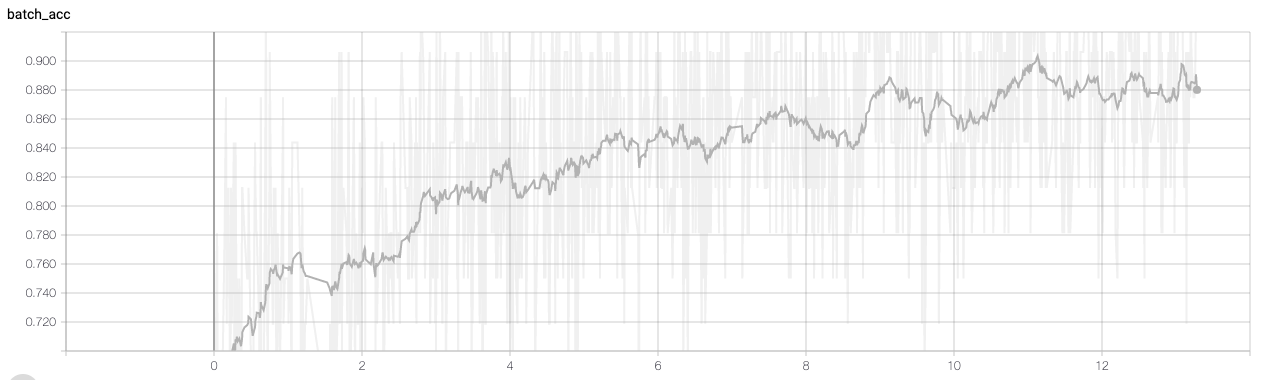


Рисунок 69 - Процесс обучения нейронной сети denseNet на 2 этап

Так знак “Стоянка запрещена по нечетным числам месяца” определен правильно с уверенностью 0.98.

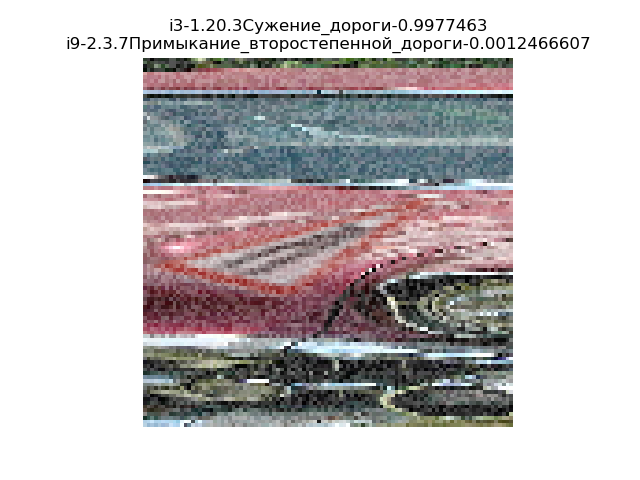


Рисунок 70- DenseNet\_51. Распознавание знака "Сужение дороги"



Рисунок 71 - DenseNet\_51. Распознавание дорожного знака "Остановка запрещена"

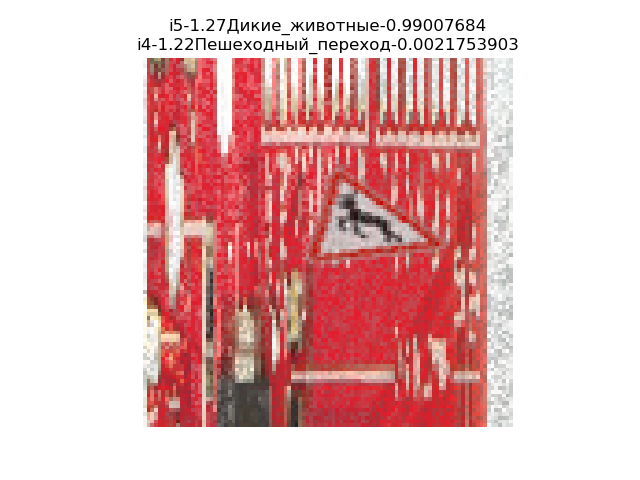
Знак “Сужение дороги” распознан правильно с уверенностью 0.99. При этом знак сильно искажен и зашумлен. Фон имеет сложную структуру, по цветовой гамме переливается с цветами знака.

Рисунок 72 - DenseNet\_51. Распознавание дорожного знака "Дикие животные"

Знак “Остановка запрещена” распознан правильно с уверенностью 0.99. Изображение также сильно искажено и зашумлено.

Дорожный знак “Дикие животные” определен правильно с показателем уверенности 0.99. Изображение сильно искажено и зашумлено.

**Вывод.** Обученная нейронная сеть desnet показала очень высокие показатели точности, при этом скорость данной сети является самой быстрой из рассматриваемых архитектур. Точность составляет 0.87. При этом время обработки одного изображения составила 268мс. А вес модели составляет 82 МБ. Применение на мобильной платформе невозможно.

* 1. Вывод к главе

В ходе изучения обучения и тестирования нейронных сетей различных архитектур были получены данные о точности работы нейронные сетей.

Таблица 1 - Точность работы нейронных сетей разных архитектур

|  |  |  |
| --- | --- | --- |
| Архитектура | 1 часть на 28 классах | 2 часть на 51 классе |
| Xception | 0.8922 | 0.9511 |
| InceptionV3 | 0.107 | 0.0705 |
| MobileNet | 0.479 | 0.8917 |
| MobileNetV2 | 0.6411 | 0.8292 |
| ResNet | - | 0.7983 |
| DenseNet | 0.8373 | 0.8747 |

Так, наиболее точными в обоих случаях являются нейронные сети Xception и DenseNet.

Таблица 2 - Скорость обучения нейронных сетей различных архитектур в 1 части, на 28 классах

| Архитектура | Время обучения на 28 классах | Обработка одной картики. мс. |
| --- | --- | --- |
| Xception | 1:40:54 | 121,08 |
| Inception | 57:52 | 69,44 |
| MobileNet | 58:16 | 69,92 |
| MobileNetV2 | 1:6:8 | 79,36 |
| ResNet | - | - |
| DenseNet | 3:10:22 | 228,44 |

По итогу обучения на 28 классах, наиболее подходящие к применению на мобильной платформе, с учетом точности и скорости работы, являются сети: MobileBetV2 и Xception.

Таблица 3 - Скорость работ архитектур на 51 классе

| Архитектора | 1 этап | 2 этап | На один знак, мс |
| --- | --- | --- | --- |
| Xception | 1:35:20 | 13:17:1 | 267,705 |
| InceptionV2 | 1:1:38 | 8:1:7 | 162,825 |
| MobileNet | 1:0:18 | 6:59:34 | 145,96 |
| MobileNetV2 | 1:14:25 | 9:46:16 | 198,205 |
| Resnet | 2:10:52 | 23:44:2 | 466,47 |
| DenseNet | 2:51:25 | 18:27:32 | 383,685 |

При обучении нейронной сети на 51 классе, ухудшается скорость работы нейронных сетей. Для использования на мобильной платформе, больше всего подходит нейронные сети: MobileNet и MobileNetV2.

1. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

Для реализации целей, поставленных в данной дипломной работе – создания инструментария для обучения и тестирования нейронных сетей, потребовались затраты, для создания программного продукта, написанного на языке python, а также косвенные затраты, связанные с расходами на обслуживание и эксплуатацию оборудования.

В данном разделе проводится расчет затрат на создание программного продукта и его тестирование. Доказывается целесообразность создания программы.

* 1. Себестоимость проекта

В ходе ВКР был предложен демонстрационный стенд применения ИНС на мобильной платформе для распознавания дорожных знаков. А также предложено теоретическое и аналитическое обоснование выбранных методов реализации демоверсии решения, а также предложены методы улучшения работы для реальных систем, с оценкой рисков научной деятельности, на основе знаний в данной области.

Для разработки программного компонента потребовалось привлечь специалистов для составления технического задания, проектирования решения, а также непосредственной разработки.

Приведем в таблице продолжительность работ каждым участником проекта, на каждом этапе, для дальнейшего расчета трудозатрат на проекте.

Таблица 4 - Расчет трудоемкости проекта

| Наименование работ | Трудоемкость чел./дни | |
| --- | --- | --- |
|  | Руководитель | Дипломант |
| Разработка технического задания | 1 | 2 |
| Разработка методов решения задачи | 1 | 5 |
| Изучение материалов | - | 24 |
| Проектирование решения | - | 14 |
| Программная реализация | - | 27 |
| Тестирование и откладка | - | 14 |
| Оформление пояснительной записки | 2 | 14 |
| Итого | 4 | 100 |

Суммируя все необходимые для выполнения проекта работы, получим, что руководителю надо затратить 4 дня, а разработчику 100 дней. Суммарное время разработки двух исполнителей займет 104 дня.

Используя полученные данные о необходимых человеко/дней на проекте, проведем расчет на основную и дополнительную заработную плату исполнителей разработки:

* Трудоёмкость выполнения работ (Таблица 13) Трук = 4 чел/дней, Тдип = 100 чел/дней.
* Дневная ставка научного руководителя Друк = 3500 руб.
* Дневная ставка разработчика Ддип = 1200 руб.
* Процент дополнительной заработной платы – 12%.
* Процент отчислений на социальные нужды – 30,2%.
* Процент накладных расходов – 33%.

Основная заработная плата исполнителей рассчитывается по формуле:

Сзо = Трук \* Друк + Тдип \* Ддип

Сзо = 4 \* 3500 + 100 \* 1200 = 14000 + 120000 = 134000 руб.

Дополнительная заработная плата составляет 12% от основной заработной платы:

Сзд = Сзо \* 0,12

Сзд = 134000 \* 0,12 = 16080 руб.

Отчисления на социальные нужды составляют 30,2% от основной и дополнительной заработной платы:

Ссн = (Сзо +Сзд) \* 0,302

Ссн = (134000 + 16080) \* 0,302 = 45324,16 руб.

Расчёт количества и стоимости материалов с учётом транспортно-заготовительных расходов занесены в таблицу 14.

Таблица 5 - Расходы на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Материалы | Кол-во | Цена, руб. | Сумма, руб. |
| Бумага для оргтехники, пачка | 2 | 260 | 520 |
| Картридж для принтера HP CE285A | 1 | 2100 | 2100 |
| Компакт диск (CD-RW) | 1 | 75 | 75 |
| Канцелярские товары |  |  | 243 |
| **ИТОГО:** | | | **2938** |
| Транспортные расходы (15%) | | | 440,7 |
| **ВСЕГО:** | | | **3378,7** |

Рассчитаем издержки на амортизацию ПК и оргтехники, которая определяется по формуле:

Аг = Коб \* Нам/100,

где Коб – балансовая стоимость оборудования,

Нам – норма амортизации. Равна 20%.

За один рабочий день сумма амортизации составит:

Ад = Аг / N,

где N – количество рабочих дней в 2019 году, равное 250 дням.

Балансовая стоимость ПК на базе ASUS K501U – 76000 руб., принтера Samsung – 6700 руб.

Амортизация за год – Аг = 16540 руб.

Амортизация за день – Ад = 66,16 руб.

Амортизация оборудования за время проектирования дипломного проекта:

Сам = 66,16\* 104 = 6880,64 руб.

К статье «Накладные расходы» относятся расходы на управление и хозяйственное обслуживание. Накладные расходы рассчитываются по формуле:

Снр = αнк \* (Сзо + Сзд)

Снр = 0,33 \* (134000 + 16080) = 0,33 \* 150080 = 49526,4 руб.

Из полученных данных по статьям расхода составляется калькуляция себестоимости разработки проекта. Себестоимость проекта представлена в таблице 15.

Таблица 6 - Калькуляция себестоимости проекта

|  |  |  |
| --- | --- | --- |
| Статья затрат | Сумма, руб. | Уд.вес., % |
| Основная заработная плата | 134000 | 52,5 |
| Дополнительная заработная плата | 16080 | 6,3 |
| Отчисление на социальные нужды | 45324,16 | 17,76 |
| Накладные расходы | 49526,4 | 19,4 |
| Материалы | 3378,7 | 1,32 |
| Издержки на амортизацию ПК и оргтехники | 6880,64 | 2,7 |
| **ИТОГО:** | **255189,9** | **100** |

## 

* 1. Вывод

В главе, которая посвящена технико-экономическому обоснованию, были рассмотрены и реализованы основные затраты на реализацию ВКР, которые связаны с разработкой программного модуля. О суммарных затратах можно сделать выводы по рассчитанным показателям экономической эффективности. В случае данной ВКР они составили **255189,9 руб**. Оплата труда исполнителя составляет основную часть данных суммарных затрат.

Заключение

В проведенной работе были изучен инструментарий разработки и тестирования моделей нейронных сетей различных архитектур TensorFlow. Данный инструментарий предоставляет больший функционал для проверки и создания нейронных сетей (в сравнении с библиотеками вроде scikit-learn).

В ходе работы были изучены принципы работы сверточных нейронных сетей таких архитектур, как: VVG16, Xceptrion, Inception, MobileNet, MobileNetV2, ResNet, DenseNet, NasNet. Для всех архитектур, кроме VVG16 и NasNet проведены практические эксперименты с изучением скорости обучения, работы и точности работы нейронных сетей. Эксперименты проводились в двух этапах, для набора из 28 дорожных знаков и 51 дорожного знака.

В ходе экспериментов было выявлено, что лучшими архитектурами, наиболее подходящими для применения на мобильных платформах, и отвечающие хорошим показателям скорости, точности и размером модели нейронной сети являются архитектуры MobileNet и MobileNet2. Точность работы данных сетей достигает показателей 0.89 и 0.82, что вполне приемлемо для применения для нетребовательных (технологически неопасных) реальных задачах. Однако скорость обработки одного изображения превышает 100 мс. Что для систем реального времени не допустимо.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

|  |  |
| --- | --- |
| [1] | Саймон и Хайкин, Нейронные сети: Полный курс, Москва: Вильямс, 2006. |
| [2] | А. А.А., И. Д.Х., К. Н.Н. и Я. В. Б., Теория управления, СПб: ЛЭТИ, 1999. |
| [3] | K. Simonyan и A. Zisserman, «Very Deep Convolutional Networks For Lange-scale Image Recognition,» в *ICLR*, 2015. |
| [4] | «ImageNet,» Stanfrod Vision Lab, 2016. [Электронный ресурс]. Available: http://www.image-net.org/. [Дата обращения: 20 Январь 2019]. |
| [5] | Min Lin, Qiang Chen и Shuaicheng Yan, «Network In Netwok,» National University of Singapore, Singapore, 2014. |
| [6] | A. Krizhevsky, I. Sutskever и G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» University of Toronto, Toronto, 2012. |
| [7] | C. Szegedy, W. Liu и Y. Jia, «Going deeper with convolution,» Google Inc., 2014. |
| [8] | К. Семен, «Эволюция нейросетей для распознавания изображений в Google: Incption-v3,» *Habr,* 31 май 2016. |
| [9] | F. Chollet, «Xception: Deep Learning with Depthwise Separable Convolutions,» Google Inc, 2017. |
| [10] | K. He, Z. Xiangyu, S. Ren и J. Sun, «Deep Resigual Learning for Image Recognition,» Microsoft Research, 2015. |
| [11] | A. G. Howard, M. Zhu, B. Chen и D. Kalenichenko, «MobileNets: Efficient Convolutional Networks for Mobile Vision Applications,» Google Inc, 2017. |
| [12] | M. Sandler и A. Howard, «MobileNetV2: The Next Generation of On-Device Computer Vision Networks,» Google Research, 2018. |
| [13] | A. Krizhevsky, «Convolutional Deep Belief Networks on CIFAR-10,» University of Toronto, Toronto, 2010. |
| [14] | G. Huang и Z. Liu, «Densely Connected Convolutional Networks,» 2018. |
| [15] | «GTSRB DataSet,» Nisys GmbH, 2010. [Электронный ресурс]. Available: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset. [Дата обращения: 20 январь 2019]. |
| [16] | Г. Алексей, «Дорожные знаки в векторе,» [Электронный ресурс]. Available: http://www.artpatch.ru/dorznaki.html. [Дата обращения: 20 Январь 2019]. |
| [17] | «Open Images Dataset V5 + Extensions,» Google LLC, [Электронный ресурс]. Available: https://storage.googleapis.com/openimages/web/index.html. [Дата обращения: Январь 2019]. |
| [18] | К. Андрей, «Обработка изображений,» Январь 2019. [Электронный ресурс]. Available: https://github.com/AndreKuzubov/TensorFlowDecisions/tree/master/image\_processing. [Дата обращения: Март 2019]. |

Приложение А

* AutoPilot-master\RoadSign\explorationDataAnalys.py

from matplotlib import gridspec as gridspec

import matplotlib.pyplot as plt

import random

from PIL import Image

import glob

import os

from RoadSign.utils import imageProccessing

from RoadSign.utils import utils

import numpy as np

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

DEBUG = False

#

# Аналитика обучающих данных

#

# @AndreyKuzubov 2018

def dataAnalitics(datasetFolder, tag):

dataSetImages = []

for path in list(next(os.walk(datasetFolder)))[1]:

dataSetImages += [

{

"images": [

fileName for fileName in glob.glob(datasetFolder + path + "/\*.ppm")

] + [

fileName for fileName in glob.glob(datasetFolder + path + "/\*.png")

],

"name": path,

}

]

\_\_datasetStatistic(dataSetImages, tag)

def \_\_datasetStatistic(picdataset, tag):

def autolabelH(rects, ax):

"""

Attach a text label above each bar displaying its height

"""

for rect in rects:

offset = max([rect.get\_width() for rect in rects]) \* 0.1

ax.text(rect.get\_width() + offset, rect.get\_y() + rect.get\_height() / 2,

str(rect.get\_width()),

ha='center', va='center', color="black")

for item in picdataset:

item["shortList"] = [random.choice(item['images']) for i in range(0, 5)]

item['glueList'] = imageProccessing.glueImagesHorisontal(

images=[Image.open(imageFile) for imageFile in item['shortList']],

size=(50, 50)

)

\_\_logDataSetStatisticValues(picdataset=picdataset, tag=tag)

gs = gridspec.GridSpec(len(picdataset), 4, wspace=0.1, hspace=0.1)

fig = plt.figure(figsize=(7, 20))

# bars1 - кол-во в наборе

axBars = plt.subplot(gs[:, 2:4])

axBars.axis('off')

rect = axBars.barh(

[d['name'] for d in picdataset],

[len(d['images']) for d in picdataset], color="blue", height=0.8)

axBars.set\_title('Count of pictures in DataSet')

axBars.set\_xlabel("count")

axBars.set\_ylim([-0.45, len(picdataset) - 0.55])

axBars.set\_yticks([])

# axBars.grid(True)

autolabelH(rect, axBars)

# визуализация картинок

axPics = [plt.subplot(gs[i, 1]) for i in range(len(picdataset))]

for i, axPic in enumerate(axPics):

axPic.axis('off')

axPic.imshow(picdataset[i]['glueList'])

# подпись набора

axTexts = [plt.subplot(gs[i, 0]) for i in range(len(picdataset))]

for i, axText in enumerate(axTexts):

axText.axis('off')

axText.text(0.5, 0.5, picdataset[i]['name'],

horizontalalignment='center',

verticalalignment='center')

utils.createNoExistsFolders("RoadSign/dataanalalysis/{tag}".format(tag=tag))

fig.savefig("RoadSign/dataanalalysis/{tag}/datasetStatistic1.png".format(tag=tag))

fig.show()

def \_\_logDataSetStatisticValues(picdataset, tag):

utils.createNoExistsFolders("RoadSign/dataanalalysis/{tag}".format(tag=tag))

with open("RoadSign/dataanalalysis/{tag}/datasetStatistics.txt".format(tag=tag), "w+") as file:

for item in picdataset:

count = len(item['images'])

images = [Image.open(imageFile) for imageFile in item['images']]

stdSize = np.std([max(image.size) for image in images])

maxSize = max([max(image.size) for image in images])

minSize = min([min(image.size) for image in images])

name = item['name']

if (DEBUG):

print("{tag} {name} count: {count} stdSize: {stdSize} maxSize: {maxSize} minSize {minSize}".format(

tag=tag,

name=name,

count=str(count),

stdSize=str(stdSize),

maxSize=str(maxSize),

minSize=str(minSize),

))

file.write(

"{tag} {name} count: {count} stdSize: {stdSize} maxSize: {maxSize} minSize {minSize}".format(

tag=tag,

name=name,

count=str(count),

stdSize=str(stdSize),

maxSize=str(maxSize),

minSize=str(minSize),

) + '\n')

file.close()

if \_\_name\_\_ == '\_\_main\_\_':

DEBUG = True

# dataAnalitics(

# datasetFolder="RoadSign/datasets/GTSRB/Training/",

# tag="GTSRB\_original"

# )

dataAnalitics(

datasetFolder="RoadSign/datasets/fromPattern\_original/Запрещающие знаки/",

tag="fromPattern\_original/Запрещающие знаки"

)

* AutoPilot-master\RoadSign\start\_createDataSetOnPatters.py

import os

import shutil

import sys

from RoadSign.utils.specificFixs import \*

from RoadSign.utils.inetUtils import \*

import requests

import lxml.html as lxmlHtml

from urllib.parse import urljoin

from RoadSign.utils import utils

import zipfile

import glob

from PIL import Image

import glob

from RoadSign.utils import imageProccessing

from RoadSign.utils import utils

import random

import numpy as np

import csv

import uuid

#

# Загрузка шаблонов с сайта

#

#

# @Andreykuzubov 2018

IMAGE\_SIZE = 100

BASE\_FOLDER = os.getcwd()

GTSRB\_DIR = "datasets/GTSRB/"

DORZNAKI\_DIR = "datasets/DorZnaki/"

BACKGROUND\_DIR = "datasets/background/"

DORZNAKI\_PREPARED\_DIR = "datasets/DorZnaki\_Prepared/"

BACKGROUND\_IMAGES = glob.glob(BACKGROUND\_DIR + "\*.jpg") + glob.glob(BACKGROUND\_DIR + "\*.png")

BACKGROUND\_IMAGES\_BUFFER = 300

backgroundImages = []

def loadGTSRB():

if (not os.path.exists(GTSRB\_DIR + "GTSRB\_Final\_Training\_Images")):

print("loading GTSRB\_Final\_Training\_Images...")

download\_file("http://benchmark.ini.rub.de/Datasets/GTSRB\_Final\_Training\_Images.zip",

local\_filename=GTSRB\_DIR + "GTSRB\_Final\_Training\_Images.zip")

print("unzipping GTSRB\_Final\_Training\_Images...")

zip\_ref = zipfile.ZipFile(GTSRB\_DIR + "GTSRB\_Final\_Training\_Images.zip", 'r')

zip\_ref.extractall(GTSRB\_DIR + "GTSRB\_Final\_Training\_Images/")

zip\_ref.close()

os.remove(GTSRB\_DIR + "GTSRB\_Final\_Training\_Images.zip")

print("success GTSRB\_Final\_Training\_Images")

if (not os.path.exists(GTSRB\_DIR + "GTSRB\_Final\_Test\_Images")):

print("loading GTSRB\_Final\_Test\_Images...")

download\_file("http://benchmark.ini.rub.de/Datasets/GTSRB\_Final\_Test\_Images.zip",

local\_filename=GTSRB\_DIR + "GTSRB\_Final\_Test\_Images.zip")

print("unzipping GTSRB\_Final\_Test\_Images...")

zip\_ref = zipfile.ZipFile(GTSRB\_DIR + "GTSRB\_Final\_Test\_Images.zip", 'r')

zip\_ref.extractall(GTSRB\_DIR + "GTSRB\_Final\_Test\_Images/")

zip\_ref.close()

os.remove(GTSRB\_DIR + "GTSRB\_Final\_Test\_Images.zip")

print("success GTSRB\_Final\_Test\_Images")

def loadDorZnaki():

baseUrl = "http://www.artpatch.ru/dorznaki.html"

html = requests.get(baseUrl).text.replace("<br>", '\n <br>')

s = lxmlHtml.fromstring(html)

def loadToLocal(category, name, url):

response = requests.get(url, stream=True)

response.raise\_for\_status()

utils.createNoExistsFolders(DORZNAKI\_DIR + "{category}".format(category=category, name=name))

with open(DORZNAKI\_DIR + "{category}/{name}.zip".format(category=category, name=name),

'wb') as handle:

for block in response.iter\_content(1024):

handle.write(block)

zip\_ref = zipfile.ZipFile(DORZNAKI\_DIR + "{category}/{name}.zip".format(category=category, name=name), 'r')

utils.createNoExistsFolders(DORZNAKI\_DIR + "{category}/zip".format(category=category, name=name))

zip\_ref.extractall(DORZNAKI\_DIR + "{category}/zip".format(category=category, name=name))

zip\_ref.close()

filename = glob.glob(DORZNAKI\_DIR + "{category}/zip/\*".format(category=category))[0]

os.rename(filename, DORZNAKI\_DIR + "{category}/{name}.ai".format(category=category, name=name))

shutil.rmtree(DORZNAKI\_DIR + "{category}/zip".format(category=category))

os.remove(DORZNAKI\_DIR + "{category}/{name}.zip".format(category=category, name=name))

print("loaded {name}".format(name=name))

content = s.xpath('//\*[@id="content"]')[0]

category = ""

url = ""

for child in content.getchildren()[7:]:

if (child.tag.lower() == "h2"):

category = child.text

if category is None:

continue

category = category.replace(" ", "\_")

elif (child.tag.lower() == "br"):

continue

elif (list(child.classes)[0] == 'znSootLeft'):

url = urljoin(base=baseUrl, url=child.getchildren()[0].attrib['href'])

elif (list(child.classes)[0] == 'znSootRight1'):

name = child.text\_content().replace('\n', '')

name = name.replace(' ', '\_')

if not os.path.exists(

DORZNAKI\_DIR + "{category}/{name}.ai".format(category=category, name=name)):

loadToLocal(category=category, url=url, name=name)

if not os.path.exists(

DORZNAKI\_DIR + "{category}/{name}.png".format(category=category, name=name)):

bash\_script = """magick convert ai:'{basefolder}/{filesource}' -resize '{size}x' -density {density} '{basefolder}/{filedestination}'"""

os.system(

bash\_script.format(

basefolder=BASE\_FOLDER,

filesource=DORZNAKI\_DIR + "{category}/{name}.ai".format(category=category,

name=name),

filedestination=DORZNAKI\_DIR + "{category}/{name}.png".format(category=category,

name=name),

size=str(IMAGE\_SIZE),

density=str(100))

)

print("files created: " + "{name}.png".format(category=category, name=name))

def loadBackGroundImages():

print("loading backgrounds...")

download\_file("https://storage.googleapis.com/openimages/2018\_04/image\_ids\_and\_rotation.csv",

local\_filename=BACKGROUND\_DIR + "image\_ids\_and\_rotation.csv")

with open(BACKGROUND\_DIR + "image\_ids\_and\_rotation.csv", "r") as csvfile:

datareader = csv.reader(csvfile)

next(datareader)

for row in datareader:

url = row[2]

name = url[url.rfind("/") + 1:]

print("loading backgrounds... %s" % (name,))

download\_file(url, local\_filename=BACKGROUND\_DIR + name)

def generateBackgroundImage(imageSize=IMAGE\_SIZE, transparent=None):

"""

:param imageSize:

:param transparent: None - не делать получпрозрачным

иначе максимальное значение прозрачности (0-1)

:return:

"""

global backgroundImages

def popFromBuffer():

img = backgroundImages[0]

del backgroundImages[0]

if (not transparent is None):

pixeldata = list(img.getdata())

for i, pixel in enumerate(pixeldata):

pixeldata[i] = pixel[:3] + (int(random.random() \* transparent \* 255),)

img.putdata(pixeldata)

return img

if len(backgroundImages) > 0:

return popFromBuffer()

while len(backgroundImages) < BACKGROUND\_IMAGES\_BUFFER:

file = random.choice(glob.glob(BACKGROUND\_DIR + "\*.jpg") + glob.glob(BACKGROUND\_DIR + "\*.png"))

try:

im = Image.open(file).convert("RGBA")

backgroundImages += [im]

except:

print("remove %s" % (file))

os.remove(file)

for i in range(len(backgroundImages)):

im = backgroundImages[i]

cropsize = (int(random.random() \* im.size[0]), int(random.random() \* im.size[1]))

cropPoint = (

int(random.random() \* (im.size[0] - cropsize[0])), int(random.random() \* (im.size[1] - cropsize[1])))

im = im.crop((cropPoint[0], cropPoint[1], cropPoint[0] + cropsize[0], cropPoint[1] + cropsize[1]))

im = im.resize((imageSize, imageSize))

backgroundImages[i] = im

return popFromBuffer()

def generateDorZnakiBanches(class\_image\_count=50000):

MIN\_SIZE\_SCALE = 0.7

classes\_files = glob.glob(DORZNAKI\_DIR + "Запрещающие\_знаки/\*.png") \

+ glob.glob(DORZNAKI\_DIR + "Предупреждающие\_знаки/\*.png") \

+ glob.glob(DORZNAKI\_DIR + "Предписывающие\_знаки/\*.png") \

+ glob.glob(DORZNAKI\_DIR + "Знаки\_приоритета/\*.png") \

+ glob.glob(DORZNAKI\_DIR + "Знаки\_особых\_предписаний/\*.png")

random.shuffle(classes\_files)

for class\_file in classes\_files:

class\_name = class\_file[class\_file.rfind("/") + 1:-4]

print("generate for class %s" % (class\_name))

if (os.path.exists(DORZNAKI\_PREPARED\_DIR + class\_name)):

continue

utils.createNoExistsFolders(DORZNAKI\_PREPARED\_DIR + class\_name)

imagesBuffer = []

sourceImage = Image.open(class\_file).convert("RGBA")

for i in range(class\_image\_count):

if i % 100 == 0:

print("i = %s" % (i))

base\_img = generateBackgroundImage()

image = sourceImage.copy().convert("RGBA")

ratio = random.uniform(0.7, 2)

image = image.resize(

(int(IMAGE\_SIZE / ratio), int(IMAGE\_SIZE \* ratio)),

Image.ANTIALIAS)

image = image.rotate(random.random() \* 90. - 45.)

ratio = random.uniform(0.5, 4)

image = image.resize(

(int(IMAGE\_SIZE / ratio), int(IMAGE\_SIZE \* ratio)),

Image.ANTIALIAS)

# darken, lighten

image = imageProccessing.boxFilter(image, boxSize=1, boxScalar=random.uniform(0.5, 1.5), padding="SAME")

sign\_size = int(random.uniform(MIN\_SIZE\_SCALE, 1.) \* IMAGE\_SIZE)

image = image.resize((sign\_size, sign\_size), Image.ANTIALIAS)

paste\_point = (

int(random.random() \* (IMAGE\_SIZE - sign\_size)),

int(random.random() \* (IMAGE\_SIZE - sign\_size)))

base\_img.paste(image, box=(paste\_point), mask=image)

blurBoxSize = int(random.random() \* 3)

if blurBoxSize > 0:

base\_img = imageProccessing.boxFilter(base\_img, boxSize=blurBoxSize, padding="SAME")

# color correction, dirty window

base\_img.alpha\_composite(generateBackgroundImage(transparent=random.random() \* 0.4))

imagesBuffer += [base\_img]

# sourceImage.show()

# image.show()

# base\_img.show()

if (len(imagesBuffer) > BACKGROUND\_IMAGES\_BUFFER):

for im in imagesBuffer:

im.save(DORZNAKI\_PREPARED\_DIR + class\_name + "/" + str(uuid.uuid4()) + ".png")

imagesBuffer = []

for im in imagesBuffer:

im.save(DORZNAKI\_PREPARED\_DIR + class\_name + "/" + str(uuid.uuid4()) + ".png")

imagesBuffer = []

if \_\_name\_\_ == "\_\_main\_\_":

# loadBackGroundImages()

# loadGTSRB()

# loadDorZnaki()

generateDorZnakiBanches(1000)

* AutoPilot-master\RoadSign\start\_to\_test\_on\_gtsrb\_indexed.py

# обучение основных архитектур компьютерного зрения на базе дорожных знаков

#

# AndreyKuzubov

#

import datetime

import os

import sys

import time

from keras.callbacks import ModelCheckpoint

from keras.preprocessing import image

import glob

import random

from PIL import Image

from tensorflow.python.keras.callbacks import TensorBoard

import tensorflow as tf

from transliterate import translit, get\_available\_language\_codes

from RoadSign.utils import imageProccessing

from RoadSign.utils import keras\_image\_processing

import numpy as np

import matplotlib.pyplot as plt

from keras.models import load\_model

import ImageNetModels

import ImageNetModels.resnet as resnet

import ImageNetModels.mobilenet as mobilenet

import ImageNetModels.mobilenetv2 as mobilenetv2

import ImageNetModels.xception as xception

import ImageNetModels.inceptionv3 as inceptionv3

import ImageNetModels.densenet as densenet

import ImageNetModels.nasnet as nasnet

os.environ['KMP\_DUPLICATE\_LIB\_OK'] = 'True'

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

TIME\_TAG = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

SOURCE\_IMAGES\_GROP\_MASK = "datasets/GTSRB\_indexed/GTSRB\_Final\_Training\_Images/GTSRB/Final\_Training/Images/\*/\*.ppm".replace("/",os.sep)

MODEL\_IMAGES\_CLASSES = sorted([f[f.rfind("/") + 1:] for f in glob.glob("datasets/DorZnaki\_Prepared/\*".replace("/",os.sep))])

PROCESSED\_IMAGES\_FOLDER = "log/images/{date}/{model}".replace("/",os.sep)

TRAINED\_MODEL\_FOLDER = "log/models/{model}\_{inputsize}\_{classes}.h5".replace("/",os.sep)

TENSOR\_BOARD\_FOLDER = "log/tensorboard/{date}/{model}".replace("/",os.sep)

IMAGE\_INPUT\_SIZE = [100, 100, 3]

TEST\_MODELS = [

["xception", xception],

["inceptionv3", inceptionv3],

["mobilenet", mobilenet],

["mobilenetv2", mobilenetv2],

["resnet", resnet],

["densenet", densenet],

# ["nasnet", nasnet],

]

def next\_batch(batch\_size, yAsNames=False):

"""

:param batch\_size: размер батча

:param yAsNames: если True, тогда labels будут == строковыми значениями (названиеми знаков)

если False - то в виде массива с одно единицей (значищей класс)

:return:

images - картики знаков

labels - названия или массив классов картинки (зависит от yAsNames)

"""

imgFiles = glob.glob(SOURCE\_IMAGES\_GROP\_MASK)

random.shuffle(imgFiles)

images = []

labels = []

for imgFile in imgFiles[:batch\_size]:

label = os.path.dirname(imgFile)

label = label[label.rfind(os.sep) + 1:]

images += [Image.open(imgFile).resize(IMAGE\_INPUT\_SIZE[:2], Image.ANTIALIAS).convert("RGB")]

y = [0] \* len(MODEL\_IMAGES\_CLASSES)

y[MODEL\_IMAGES\_CLASSES.index(label)] = 1

labels += [label] if yAsNames else [y]

return images, labels

if \_\_name\_\_ == "\_\_main\_\_":

for modelSetting in TEST\_MODELS:

print("modeltesting on GTSRB\_indexed: %s..." % (modelSetting[0]))

testBatchSize = 100

processedImagesByModelFolder = PROCESSED\_IMAGES\_FOLDER.format(date=TIME\_TAG, model=modelSetting[0])

logTrainPath = TENSOR\_BOARD\_FOLDER.format(date=TIME\_TAG, model=modelSetting[0])

loadModelPath = TRAINED\_MODEL\_FOLDER.format(model=modelSetting[0], inputsize=str(IMAGE\_INPUT\_SIZE),

classes=str(len(MODEL\_IMAGES\_CLASSES)))

if not os.path.exists(logTrainPath):

os.makedirs(logTrainPath)

if not os.path.exists(processedImagesByModelFolder):

os.makedirs(processedImagesByModelFolder)

model = load\_model(loadModelPath)

print(model.summary())

batch\_images, batch\_labels = next\_batch(batch\_size=testBatchSize)

batch\_x = modelSetting[1].preprocess\_images(batch\_images)

batch\_y = np.array(batch\_labels)

startTime = time.time()

score = model.evaluate(batch\_x, batch\_y, verbose=0)

endTime = time.time()

print("Model: %s Test score: %.4f Test accuracy: %.4f SprendTime %.4f for batch %d" % (

modelSetting[0], score[0], score[1], endTime - startTime, testBatchSize))

imgs, img\_names = next\_batch(batch\_size=20, yAsNames=True)

batch\_imgs\_tensor = modelSetting[1].preprocess\_images(imgs)

for imageIndex, img in enumerate(imgs):

imageName = img\_names[imageIndex]

imageNameEn = translit(imageName, "ru", reversed=True)

pred = modelSetting[1].predict(model, img)

predClasses = list(modelSetting[1].decodeClasses(pred, top=2, customClasses=MODEL\_IMAGES\_CLASSES))

print("imageName = %s" % str(['-'.join(str(p) for p in c) for c in predClasses[0]]))

predictedImageNet = translit(str(['-'.join(str(p) for p in c) for c in predClasses[0]]), "ru",

reversed=True)

tf.summary.image(imageNameEn + "----->" + predictedImageNet,

np.expand\_dims(batch\_imgs\_tensor[imageIndex], axis=0))

fig = plt.gca()

fig.axis('off')

fig.set\_title("\n".join(['-'.join(str(p) for p in c) for c in predClasses[0]]))

fig.imshow(img)

plt.savefig(processedImagesByModelFolder + "/{index}\_{imagename}.png".replace("/",os.sep).format(imagename=imageName,

index=str(imageIndex)))

sess = tf.Session()

summary\_writer = tf.summary.FileWriter(logdir=logTrainPath)

all\_summaries = tf.summary.merge\_all()

summary = sess.run(all\_summaries)

summary\_writer.add\_summary(summary=summary)

summary\_writer.close()

* AutoPilot-master\RoadSign\start\_to\_train\_sign\_recognation.py

# обучение основных архитектур компьютерного зрения на базе дорожных знаков

#

# AndreyKuzubov

#

import datetime

import os

import sys

import time

from keras.callbacks import ModelCheckpoint

from keras.preprocessing import image

import glob

import random

from PIL import Image

from tensorflow.python.keras.callbacks import TensorBoard

import tensorflow as tf

from transliterate import translit, get\_available\_language\_codes

from RoadSign.utils import imageProccessing

from RoadSign.utils import keras\_image\_processing

import numpy as np

import matplotlib.pyplot as plt

from keras.models import load\_model

import json

import ImageNetModels

import ImageNetModels.resnet as resnet

import ImageNetModels.mobilenet as mobilenet

import ImageNetModels.mobilenetv2 as mobilenetv2

import ImageNetModels.xception as xception

import ImageNetModels.inceptionv3 as inceptionv3

import ImageNetModels.densenet as densenet

import ImageNetModels.nasnet as nasnet

os.environ['KMP\_DUPLICATE\_LIB\_OK'] = 'True'

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

TIME\_TAG = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

SOURCE\_IMAGES\_GROP\_MASK = "datasets/DorZnaki\_Prepared/\*/\*.png".replace("/",os.sep)

SOURCE\_IMAGES\_CLASSES = sorted([f[f.rfind(os.sep) + 1:] for f in glob.glob("datasets/DorZnaki\_Prepared/\*".replace("/",os.sep))])

PROCESSED\_IMAGES\_FOLDER = "log/images/{date}/{model}".replace("/",os.sep)

TRAINED\_MODEL\_FOLDER = "log/models/{model}\_{inputsize}\_{classes}.h5".replace("/",os.sep)

TENSOR\_BOARD\_FOLDER = "log/tensorboard/{date}/{model}".replace("/",os.sep)

IMAGE\_INPUT\_SIZE = [100, 100, 3]

TEST\_MODELS = [

["xception", xception],

["inceptionv3", inceptionv3],

["mobilenet", mobilenet],

["mobilenetv2", mobilenetv2],

["resnet", resnet],

["densenet", densenet],

# ["nasnet", nasnet],

]

def next\_batch(batch\_size, yAsNames=False):

"""

:param batch\_size: размер батча

:param yAsNames: если True, тогда labels будут == строковыми значениями (названиеми знаков)

если False - то в виде массива с одно единицей (значищей класс)

:return:

images - картики знаков

labels - названия или массив классов картинки (зависит от yAsNames)

"""

imgFiles = glob.glob(SOURCE\_IMAGES\_GROP\_MASK)

random.shuffle(imgFiles)

images = []

labels = []

for imgFile in imgFiles[:batch\_size]:

label = os.path.dirname(imgFile)

label = label[label.rfind(os.sep) + 1:]

images += [Image.open(imgFile).convert("RGB")]

y = [0] \* len(SOURCE\_IMAGES\_CLASSES)

y[SOURCE\_IMAGES\_CLASSES.index(label)] = 1

labels += [label] if yAsNames else [y]

return images, labels

if \_\_name\_\_ == "\_\_main\_\_":

with open("log/models/classes\_{count}.json".replace("/",os.sep).format(count=str(len(SOURCE\_IMAGES\_CLASSES))), "w+") as fl:

fl.write(json.dumps(SOURCE\_IMAGES\_CLASSES, ensure\_ascii=False))

fl.close()

for modelSetting in TEST\_MODELS:

print("modeltraining: %s..." % (modelSetting[0]))

saveModelPath = TRAINED\_MODEL\_FOLDER.format(model=modelSetting[0], inputsize=str(IMAGE\_INPUT\_SIZE),

classes=str(len(SOURCE\_IMAGES\_CLASSES)))

model = modelSetting[1].getModel(inputSize=IMAGE\_INPUT\_SIZE,

classesCount=len(SOURCE\_IMAGES\_CLASSES),

autoSave=False,

path=saveModelPath)

print(model.summary())

logTrainPath = TENSOR\_BOARD\_FOLDER.format(date=TIME\_TAG, model=modelSetting[0])

if not os.path.exists(logTrainPath):

os.makedirs(logTrainPath)

for i in range(10):

batch\_images, batch\_labels = next\_batch(batch\_size=10000)

batch\_x = modelSetting[1].preprocess\_images(batch\_images)

batch\_y = np.array(batch\_labels)

session = model.fit(batch\_x, batch\_y,

verbose=1,

epochs=1,

# validation\_data=(test\_xs, test\_ys),

validation\_split=0.1,

callbacks=[

ModelCheckpoint(filepath=saveModelPath,

monitor="val\_acc",

save\_best\_only=True,

save\_weights\_only=False,

mode="auto"),

TensorBoard(log\_dir=logTrainPath,

histogram\_freq=0,

write\_graph=True,

write\_grads=True,

write\_images=True)

])

model.save(saveModelPath)

for modelSetting in TEST\_MODELS:

print("modeltesting: %s..." % (modelSetting[0]))

testBatchSize = 100

processedImagesByModelFolder = PROCESSED\_IMAGES\_FOLDER.format(date=TIME\_TAG, model=modelSetting[0])

logTrainPath = TENSOR\_BOARD\_FOLDER.format(date=TIME\_TAG, model=modelSetting[0])

loadModelPath = TRAINED\_MODEL\_FOLDER.format(model=modelSetting[0], inputsize=str(IMAGE\_INPUT\_SIZE),

classes=str(len(SOURCE\_IMAGES\_CLASSES)))

if not os.path.exists(logTrainPath):

os.makedirs(logTrainPath)

if not os.path.exists(processedImagesByModelFolder):

os.makedirs(processedImagesByModelFolder)

model = load\_model(loadModelPath)

batch\_images, batch\_labels = next\_batch(batch\_size=testBatchSize)

batch\_x = modelSetting[1].preprocess\_images(batch\_images)

batch\_y = np.array(batch\_labels)

startTime = time.time()

score = model.evaluate(batch\_x, batch\_y, verbose=0)

endTime = time.time()

print("Model: %s Test score: %.4f Test accuracy: %.4f SprendTime %.4f for batch %d" % (

modelSetting[0], score[0], score[1], endTime - startTime, testBatchSize))

imgs, img\_names = next\_batch(batch\_size=20, yAsNames=True)

batch\_imgs\_tensor = modelSetting[1].preprocess\_images(imgs)

for imageIndex, img in enumerate(imgs):

imageName = img\_names[imageIndex]

imageNameEn = translit(imageName, "ru", reversed=True)

pred = modelSetting[1].predict(model, img)

predClasses = list(modelSetting[1].decodeClasses(pred, top=2, customClasses=SOURCE\_IMAGES\_CLASSES))

print("imageName = %s" % str(['-'.join(str(p) for p in c) for c in predClasses[0]]))

predictedImageNet = translit(str(['-'.join(str(p) for p in c) for c in predClasses[0]]), "ru",

reversed=True)

tf.summary.image(imageNameEn + "----->" + predictedImageNet,

np.expand\_dims(batch\_imgs\_tensor[imageIndex], axis=0))

fig = plt.gca()

fig.axis('off')

fig.set\_title("\n".join(['-'.join(str(p) for p in c) for c in predClasses[0]]))

fig.imshow(img)

plt.savefig(processedImagesByModelFolder + "/{index}\_{imagename}.png".replace("/",os.sep).format(imagename=imageName,

index=str(imageIndex)))

sess = tf.Session()

summary\_writer = tf.summary.FileWriter(logdir=logTrainPath)

all\_summaries = tf.summary.merge\_all()

summary = sess.run(all\_summaries)

summary\_writer.add\_summary(summary=summary)

summary\_writer.close()

* AutoPilot-master\RoadSign\utils\imageProccessing.py

from PIL import Image, ImageDraw

import numpy as np

import tensorflow as tf

import os

def wbImage(image):

x = tf.constant(np.asarray(image), dtype=tf.float32)

if (x.shape[2] == 4):

y = tf.constant(np.asarray(

[

[1 / 3, 1 / 3, 1 / 3, 0],

[1 / 3, 1 / 3, 1 / 3, 0],

[1 / 3, 1 / 3, 1 / 3, 0],

[0, 0, 0, 1],

]

), dtype=tf.float32)

else:

y = tf.constant(np.asarray(

[

[1 / 3, 1 / 3, 1 / 3],

[1 / 3, 1 / 3, 1 / 3],

[1 / 3, 1 / 3, 1 / 3],

]

), dtype=tf.float32)

# print('x shape:', x.shape)

# print('y shape:', y.shape)

out = tf.tensordot(x, y, axes=[[2], [0]])

with tf.Session() as sess:

result = sess.run(out)

sess.close()

# print('result shape:', result.shape)

return Image.fromarray(np.array(result, dtype=np.uint8))

def lightingImage(image, ratio):

x = tf.constant(np.asarray(image), dtype=tf.float32)

y = tf.constant(np.asarray(

[ratio, ratio, ratio, 1],

), dtype=tf.float32)

print('x shape:', x.shape)

print('y shape:', y.shape)

out = tf.einsum("ijk,k->ijk", x, y)

with tf.Session() as sess:

result = sess.run(out)

sess.close()

# print('result shape:', result.shape)

return Image.fromarray(np.array(result, dtype=np.uint8))

def contrastImage(image, ratio):

x = tf.constant(np.asarray(image), dtype=tf.float32)

y = tf.constant(1 / 8 \* np.asarray(

[

[1, 1, 1],

[1, 2, 1],

[1, 1, 1]

]

), dtype=tf.float32)

print('x shape:', x.shape)

print('y shape:', y.shape)

out = tf.nn.conv2d(x, y, padding="VALID", strides=[1, 1, 1, 1])

with tf.Session() as sess:

result = sess.run(out)

sess.close()

# print('result shape:', result.shape)

return Image.fromarray(np.array(result, dtype=np.uint8))

def blur(sourceFile, destinationFile, radius=100, sigma=1):

os.system("magick convert -blur {radius}x{sigma} {source} {destination}"

.format(

source=sourceFile,

destination=destinationFile,

radius=str(radius),

sigma=str(sigma)

))

def boxFilter(sourceImage, boxSize=3, boxScalar=None, padding="VALID"):

"""

box фильтр - размытие изображения

:param sourceImage: исходное изображение

:param boxScalar: множетель ядра фильтра - определяет яркость полученного зображения

если boxScalar = 1/boxSize\*\*2 - яркость не меняется

:param padding: исходное изображение

:return: обработанное изображение

"""

if (boxScalar is None):

boxScalar = 1. / boxSize \*\* 2

x\_image = tf.constant(np.asarray(sourceImage).astype(np.float32), dtype=tf.float32)

x\_image = tf.transpose(x\_image, [2, 0, 1])

x\_image = tf.reshape(x\_image, [n.value for n in x\_image.shape] + [1])

kernel = tf.constant(boxScalar \* np.asarray([[1] \* boxSize] \* boxSize), dtype=tf.float32)

kernel = tf.reshape(kernel, [boxSize, boxSize, 1, 1])

filtered = tf.nn.conv2d(x\_image, kernel, strides=[1, 1, 1, 1], padding=padding)

with tf.Session() as sess:

y\_image, = sess.run([filtered])

y\_image = y\_image.transpose((3, 1, 2, 0,))

y\_image = y\_image.reshape(y\_image.shape[1:])

# обработка засветов

y\_image = np.minimum(y\_image, 255)

y\_image = np.maximum(y\_image, 0)

newImage = Image.fromarray(y\_image.astype(np.uint8), sourceImage.mode)

return newImage

def glueImagesHorisontal(images, size=(30, 30)):

widths, heights = size

total\_width = int(widths \* len(images))

total\_height = heights

new\_im = Image.new('RGBA', (total\_width, total\_height))

x\_offset = 0

for im in images:

im = im.resize(size)

new\_im.paste(im, (x\_offset, 0))

x\_offset += im.size[0]

return new\_im

* AutoPilot-master\RoadSign\utils\keras\_image\_processing.py

"""

source: https://github.com/AndreKuzubov/TensorFlowDecisions

"""

import sys

import os

sys.path.append('../')

import tensorflow as tf

import numpy as np

from PIL import Image

from keras.models import Sequential

from keras.layers import Cropping2D, Lambda

def scroppingImage(sourceImage, vertical\_cropping=(2, 2), horizontal\_croping=(4, 4)):

"""

:param sourceImage:

:param vertical\_cropping: кол-во пикселей обрезать сверху и снизу

:param horizontal\_croping: кол-во пикселей обрезать слева и справа

:return:

"""

model = Sequential()

model.add(Cropping2D(cropping=(vertical\_cropping, horizontal\_croping)))

xImage = np.asarray(sourceImage)

inputShape = xImage.shape

xImage = xImage.reshape([1, inputShape[0], inputShape[1], inputShape[2]])

yImage = model.predict(xImage)

outputShape = yImage.shape

yImage = yImage.reshape([outputShape[1], outputShape[2], outputShape[3]])

return Image.fromarray(yImage.astype(np.uint8))

def scallingImage(sourceImage, size=(256, 256)):

model = Sequential()

model.add(Lambda(lambda image: tf.image.resize\_images(image, size)))

xImage = np.asarray(sourceImage)

inputShape = xImage.shape

xImage = xImage.reshape([1, inputShape[0], inputShape[1], inputShape[2]])

yImage = model.predict(xImage)

outputShape = yImage.shape

yImage = yImage.reshape([outputShape[1], outputShape[2], outputShape[3]])

return Image.fromarray(yImage.astype(np.uint8))

if \_\_name\_\_ == "\_\_main\_\_":

source = Image.open("log/source\_image.jpg")

copped\_image = scroppingImage(sourceImage=source, vertical\_cropping=(100, 100), horizontal\_croping=(200, 400))

copped\_image.show()

copped\_image.save("log/scroppingImage.jpg")

scalled\_image = scallingImage(sourceImage=source)

scalled\_image.show()

scalled\_image.save("log/scallingImage.jpg")

* AutoPilot-master\RoadSign\utils\specificFixs.py

import os

os.environ['KMP\_DUPLICATE\_LIB\_OK']='True'

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

* AutoPilot-master\ImageNetModels\xception.py

import os

import sys

from keras import Input

from keras.preprocessing import image

import numpy as np

import keras

import RoadSign.utils.utils as utils

from keras.models import load\_model

import tensorflow as tf

XCEPTION\_PATH = "log/models/XCEPTION\_{input\_size}\_{classes}.h5"

def getModel(inputSize, classesCount=1000, autoSave=True, path=None):

"""

Получение модели ИНС для обучения.

:param inputSize: size for inputing images. min valid value = 71

ожидается в виде массива [sizeW,sizeH,channelsCount]

:param classesCount: кол-во классов на выходе. Если не равно 1000, то модель не будет предобучена на наборе imagenet

:return:

"""

if (path is None):

path = XCEPTION\_PATH.format(input\_size=str(inputSize), classes=str(classesCount))

if (os.path.exists(path)):

model = load\_model(path)

model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy"])

return model

if not os.path.exists(os.path.dirname(path)):

os.makedirs(os.path.dirname(path))

model = keras.applications.xception.Xception(include\_top=True,

weights='imagenet' if classesCount == 1000 else None,

input\_tensor=keras.Input(shape=inputSize),

input\_shape=None, pooling=None, classes=classesCount)

model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy"])

if not os.path.exists(os.path.dirname(path)):

os.makedirs(os.path.dirname(path))

if (autoSave):

model.save(path)

return model

def decodeClasses(predictedClasses, top=3, customClasses=None):

"""

декодирование полученных классов по предсказанным этой моделью.

:param predictedClasses:

:param customClasses: должен всегда использоватся, если модель обучена не на imagenet

:return:

"""

if (not customClasses is None):

decodedClasses = []

for predictClass in predictedClasses:

mapClasses = [{

"className": ("i" + str(i), customClasses[i], pred),

"pred": pred

} for i, pred in enumerate(predictClass)]

mapClassesBest = sorted(mapClasses, key=lambda k: k['pred'], reverse=True)

decodedClasses += [[cl["className"] for cl in mapClassesBest[:top]]]

return decodedClasses

return keras.applications.xception.decode\_predictions(predictedClasses, top=top)

def preprocess\_images(imgs):

x = np.array([image.img\_to\_array(img) for img in imgs])

return keras.applications.xception.preprocess\_input(x)

def predict(model, img):

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

x = keras.applications.xception.preprocess\_input(x)

preds = model.predict(x)

return preds