# Введение

Экспертные системы появились в 60-х годах прошлого века. Изначально они были разработаны как средства для научных исследований. Такие системы использовались для решения узкоспециализированных задач, например, постановка диагноза больному. Однако на тот момент создание такой системы было жутко дорогим и сложным делом. Только спустя десятилетия, а точнее через 20 лет, экспертные системы получили распространение во многих сферах.

Под экспертной системой понимается интеллектуальное приложение для имитирования суждения эксперта. В таких системах используется искусственный интеллект. У ИИ множество определений, например, одно из них: “Способ заставить компьютеры думать, как люди”. Также есть такое определение ИИ — это ситуация, в которой машины имитируют интеллектуальное поведение, которое присуще человеку - его когнитивные способности. Тое есть система должна действовать как человек. Однако не удалось создать такую систему, которая могла бы решать и прогнозировать все задачи. Но они особенно эффективны в ограниченных областях.

Обычно такие системы содержат базу знаний – данные или накопленный опыт в конкретной области, внутреннюю логику – применение базы знаний с с формированием экспертного ответа, интерфейс вывода – отображение этого ответа. Такие системы могут использовать возможности машинного обучения – отрасли ИИ, для обучения компьютеров решению задач. Это означает создание алгоритмов, которые могут самообучаться на входных данных и выдавать предсказание.

В данной курсовой работе, будет создана экспертная система по решению простых арифметических уравнений.

1. Анализ предметной области
   1. Цели создания

Основной целью данного курсового проекта является разработка программного модуля на языке программирования Python для расчёта арифметических выражений, представленных в графическом виде.

* 1. Формулировка задачи

Задача, которая может решать данная система. На вход системы подается изображение, несущее в себе информацию о цифрах и операторах. Системе требуется преобразовать изображение, найти контуры цифр, операторов, используя нейронный сети распознать их, составить выражение и решить его.

Входным параметром для данного курсового проекта будет растровое изображение. Растровое изображение – изображение, которое представляет собой сетку пикселей (точек). Это трехмерный массив чисел, каждое число соотвествует значению пикселя цветовой модели RGB. При накладывании всех слоев получается цветное изображение.

Выходным параметром будет число. Это ответ выражения.

1. Обоснование метода решения поставленной задачи
   1. Определение границ

Система будет распознавать арабские цифры от 0 до 9. Также арифметические операторы:

* - умножение
* / - деление
* + - сложение
* - - вычитание
  1. Представление цифр и операторов

Для человека распознать цифру на бумаге не представляет труда. Однако это очень сложный процесс. Сигналы, полученные из рецепторов в сетчатке глаза, поступают в мозг, там они обрабатывается затылочной долей мозга. Далее обработанная информация (если можно так назвать) направляется в другие участки мозга, где её можно использовать. На рисунке 2.2.1 можно увидеть число два, которое все могут узнать.



Рисунок 2.2.1 – арабская цифра 2 (два)

Но компьютер так не сделает. Не сможет увидеть изображение, как человек, и как-то распознать его. Для него изображение это всего лишь набор пикселей. На рисунке 2 можно увидеть черно-белое растровое представление цифры 2.2.2.

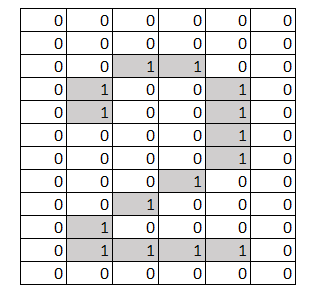


Рисунок 2.2.2 – представление цифры 2, здесь 0 означает белый цвет, 1 черный

Для черно-белого изображения объем информации одного пикселя равен 1 биту. Она может быть белой либо черной.

* 1. Обоснование алгоритмов преобразования изображений

Но все же существуют способы распознавания объектов. Однако сначала нужно выделить из входного изображения требуемые объекты, чтобы потом классифицировать их. Обычно принимают следующие шаги по преобразованию изображений:

* Первое, что нужно сделать с изображением преобразовать его в изображение с оттенками серого (grayscale image). Обычное растровое изображение содержит три канала – красный, зеленый, синий – модель RGB. Каждый пиксел — это комбинация трех интенсивностей каждого канала. Grayscale image содержит только один канал. Здесь пиксел определяется одним значением интенсивности от 0 до 255. Поэтому, для ускорения обработки изображения используется grayscale image. На рисунке 2.3.1 можно увидеть пример.



Рисунок 2.3.1 – слева изображение с одним каналом, справа изображение с тремя каналами

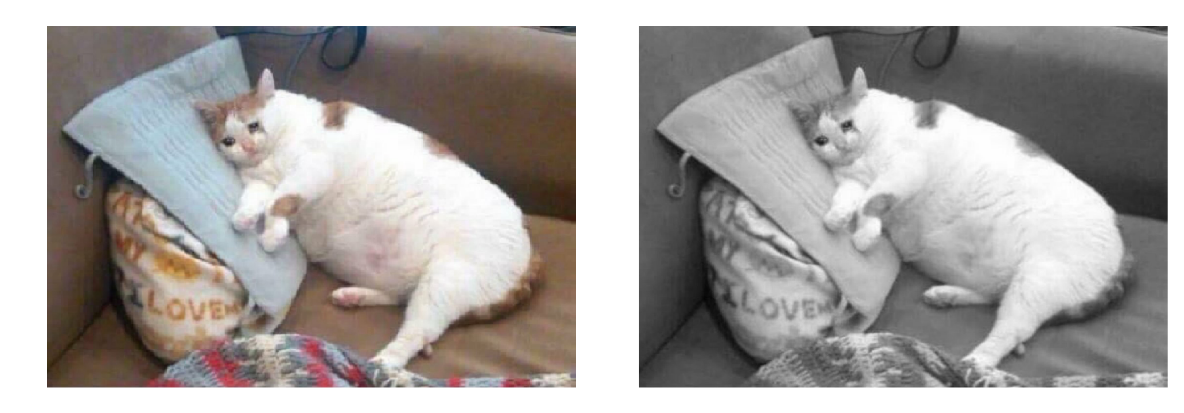
Визуальной разницы нету. Значение L:255 определяет белый цвет, когда для изображения с 3-каналами определяется так - R:255 G:255 B:255. На самом деле это не очень удачный пример. Поэтому на рисунке 2.3.2 представлен более понятный для восприятия пример

Рисунок 2.3.2 – слева находится цветное изображение, справа находится изображение с оттенками серого

* Затем следует убрать шумы. Шумы влияют на точность нахождения правильных контуров. Для того чтобы убрать шумы применяют размытие изображений или blur. Классическим примером является фильтр Гаусса. Здесь самым важным элементов является kernel(ядро) – матрица 3x3 коэффициентов фильтра размытия. На рисунке 2.3.3 представлена стандартная матрица, применяющаяся в размытии Гаусса.

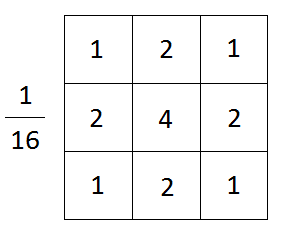


Рисунок 2.3.3 – значения матрицы kernel

Самый важный элемент в этой матрице находится в центре – 4. Именно он будет “накладываться” на значение пикселя исходного изображения. На рисунке 2.3.4 представлен пример. Слева находиться матрица интенсивности, снизу матрица kernel. Синим обозначена наложение kernel матрицы. Красным выделена ячейка, для которой подсчитывается значение.

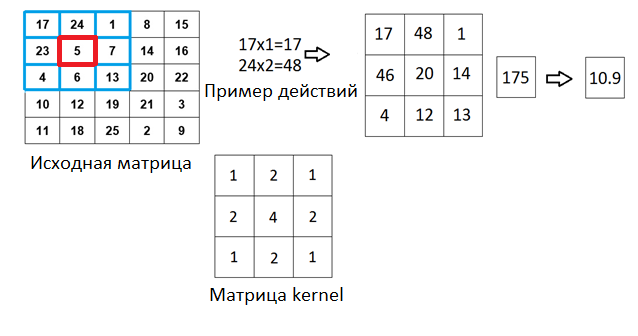


Рисунок 2.3.4 – процесс подсчета для одного значения

В матрицу 3x3 записываются результаты умножения значений интенсивности и коэффициента. Тое есть 17 умножается на 1, затем 24 умножается на 2, 1 умножается на 1 и так по всем 9 элементам. После того как матрица была заполнена, подсчитывается сумма всех её элементов. Здесь она равна 175. Затем сумма делится на 16(сумма значений матрицы kernel) Полученное число заносится в новую матрицу. Тое есть на выходе должна получиться матрица 5x5(размеры входной и выходной матрицы должны быть одинаковы), где на месте [2][2] находится значение 10.9. Рисунок 2.3.5.



Рисунок 2.3.5 – выходная матрица

И так проделывается для остальных значений входной матрицы.

На рисунке 2.3.6 представлено размытое изображение.



Рисунок 2.3.6 – применение размытия с kernel 3x3

* Затем следует threshold или пороговое значение. Это простейший метод сегментации – выделение областей изображения с разными интенсивностями. Каждое значение пикселя сравнивается с пороговым значением. Если значение пикселя больше порогового значения, то значение пикселя меняется на определенное (белое, значение 255), если нет, то на другое (черное, значение 0).

|  |  |
| --- | --- |
|  | (2.3.1) |

На рисунке 2.3.7 представлен результат применения операции threshold.



Рисунок 2.3.7 – применение threshold

* И наконец идет распознавание контура. Контуром называется кривая, которая соединяет точки, имеющий одинаковую интенсивность. Одним из способов нахождения контура является применение оператора Собеля. Он вычисляет приближенные значения интенсивности пикселя по двум направлениям – x и y.

Применение оператора позволяет определить значение производной изменения интенсивности в горизонтальном направлении, а в вертикальном. На рисунке 2.3.8 представлены значения матриц и .

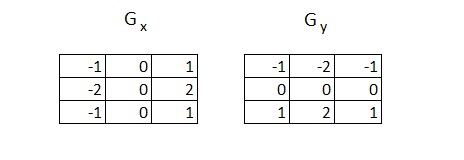


Рисунок 2.3.8 – значение матриц и

Затем следует вычислить градиент для пикселя с координатами по формуле 2.4.1.

|  |  |
| --- | --- |
|  | (2.4.1) |

На рисунке 2.3.9 представлено изображение применение оператора Собеля.



Рисунок 2.3.9 – применение оператора Собеля

2.4 Формирование базы данных

Одним из основных признаков экспертной системы является база фактов, содержащая данные. Так как разрабатываемая система должна оперировать с образами цифр и операторов, то нужно составить базу фактов, содержащая изображения их. Всего получается 14 классов.

Будет использован набор данных handwritenarifmeticoperators взятый на сайте Kaggle.com – сайт на котором в свободном доступе представлены разные наборы данных для классификации и машинного обучения. Этот набор данных включает большое количество операторов и знаков (sin, cos, ≠), но будут использоваться только те классы, которые были описаны в главе 2.1 Определение границ.

Но нужно разделить этот набор данных на две части. Первая часть называется обучающие данные или обучающий набор – используется для построения модели машинного обучения. Второй частью будет тестовым набором – для оценки качества модели. Обычно используют 75% для обучающего и 25% для тестового.

В наборе обучения входят 16 папок. Общее количество файлов 15540. В тестовый набор также входят 16 папок и общее количество файлов 4675.

1. Разработка системы

## 3.1 Разработка алгоритма

В этом подпункте будет описан алгоритм работы системы. На рисунке 3.1.1 представлен алгоритм.



Рисунок 3.1.1 – алгоритм работы

## 3.2 Построение диаграммы пользовательского интерфейса

На рисунке 3.2.1 представлена диаграмма пользовательского интерфейса.



Рисунок 3.2.1 – диаграмма пользовательского интерфейса

## 3.3 Реализация системы

Система была реализована как веб-приложение используя фреймворк Flask. В качестве базы данных для хранения проделанных операций использовалась MongoDB. Для обучения модели использовалась библиотека Keras. Для определения контуров цифр и оператора была использована библиотека OpenCV.

Сначала нужно было обучить модель на собранном наборе данных. Для этого был создан отдельный файл.

Сразу возьмем нужные модули. Рисунок 3.3.1

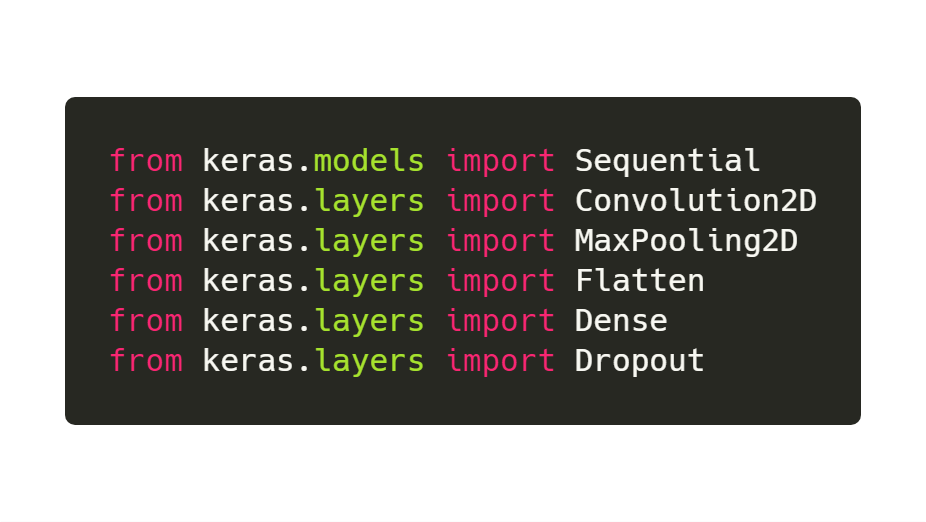


Рисунок 3.3.1 – импортированные модули

Для повышения точности распознавания была взята такая последовательность модулей. Рисунок 3.3.2.



Рисунок 3.3.2 – последовательность модулей

Во 2-ой строчке определяется сверточный слой с 32-мя фильтрами размер которых равен 3x3. Тое есть фильтры пройдут по входящему изображению размером 45x45 пикселей с 3-мя каналами и найдут специфичные значения пикселей. Используется функция активации Relu.

В 4-ой применяется max-пулинг. Он выберет самые большие числа из массива фильтров. Используется для уменьшения количества точек.

В 5-ой строчке используется Dropout. Для предотвращения переобучения.

Затем идут ещё 2 слоя и в строке 10 используется Flatten. Это вектор, который содержит все значения их фильтров. Так как последующие вычисления будут происходить с векторами.

В строке 11 используется Dense или full-connected layer. Т.е. создается скрытый слой нейронов с 512-ю точками. В 13-ой строчке будет нейрон выхода с 16 выходными точками. Используется функция активации softmax, так как число параметров выхода больше 1.

На рисунке 3.3.3 представлено продолжение скрипта для обучения модели.



Рисунок 3.3.3 - продолжение скрипта

В строчке 1 вызываем метод compile. Параметр optimizer – используемый оптимизатор, loss – функция ошибок, metrics – параметр, по которому считается качество модели.

Затем идет метод ImageDataGenerator – нормирует пиксели и немного трансформирует изображение.

Далее метод flow\_from\_directory. Нужно указать путь к папке, содержащие изображения, target\_size – размер изображений, batch – количество изображений, которые будут обработаны, прежде чем синоптические веса будут обновлены, class mode - указывает тип возвращаемых меток.

И наконец на рисунке 3.3.4 представлен последний фрагмент скрипта обучения.



Рисунок 3.3.4 – последний фрагмент скрипта обучения

Вызывается метод fit – для обучения модели. Указывается обучающая выборка, количество изображений в наборе данных, epochs – количество эпох или одна пройдённая обучающая выборка, validation\_data – указывается тестовый набор, validation\_steps – указывается количество изображений в тестовом наборе данных.

И последняя строчка сохранение модели в файл, чтобы потом эту модель можно использовать. В этот файл сохраняются веса модели, архитектура модели и конфигурация обучения.

Теперь перейдем к программе. На рисунке 3.3.6 представлен код декоратора /file\_on. Он является основным для определения и классификации изображений. Он представляет из себя последовательность действий, описанных в подпункте 3.1 Разработка алгоритма.



Рисунок 3.3.6 – основной метод для классификации

На рисунке 3.3.7 представлен интерфейс для взаимодействия пользователя и системы.

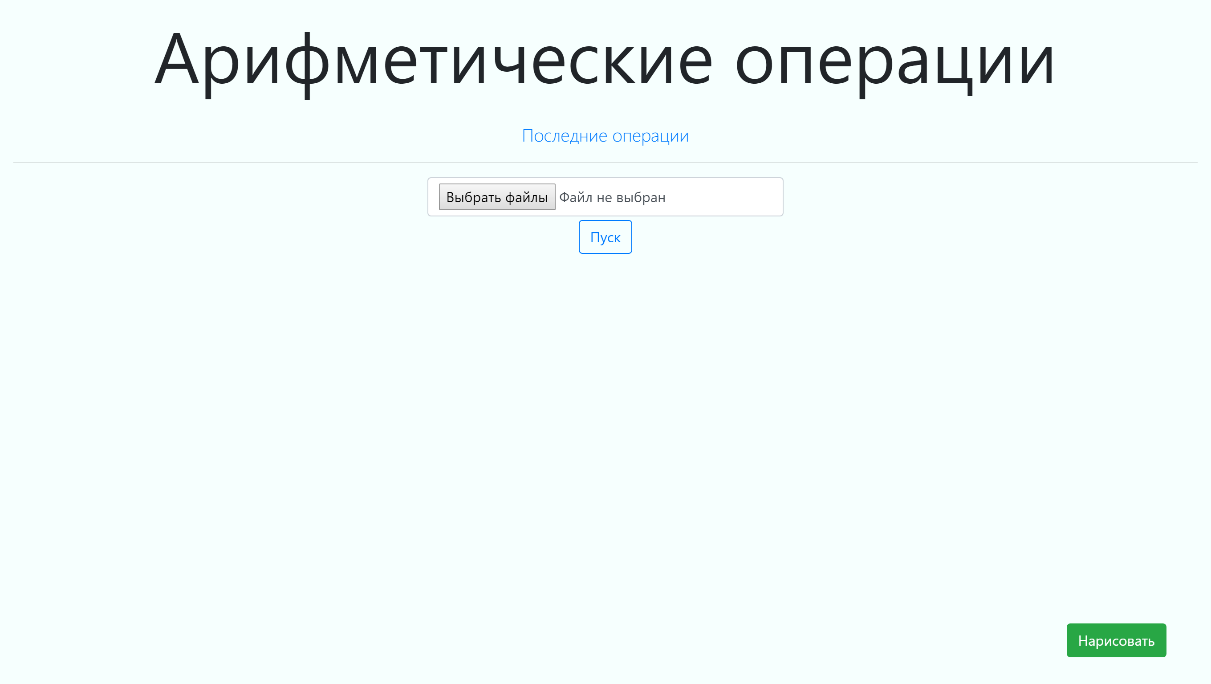


Рисунок 3.3.7 – интерфейс приложения

1. Тестирование разработанной системы
   1. Тестирование рукописных арифметических действий

На рисунке 4.2.1 представлен результат работы системы.

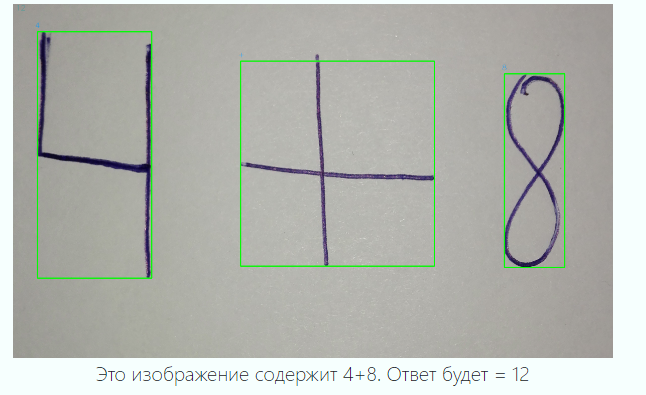


Рисунок 4.2.1 – результат работы

* 1. Тестирование арифметических действий, сделанных цифровым способом

На рисунке 4.2.2 представлен результат работы системы.



Рисунок 4.2.2 – результат работы

# Заключение

В данном курсовой работе была реализована система для расчёта арифметических выражений, представленных в графическом виде. Были использованы библиотеки: Keras, OpenCV, Numpy, Flask.

Также можно сказать и про точность распознавания. При тестировании на рукописных картинках, система со 2-ого раза дала правильный результат. При создании рисунка через область для рисования в браузере, система смогла дать приемлемое распознавание только с 6 раза.

# Список использованных источников

* Антонио Д. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2018. – 294 c.: ил.
* Handwritten math symbols dataset [Электронный ресурс] Режим доступа: <https://www.kaggle.com/xainano/handwrittenmathsymbols> - Дата доступа: 08.12.2018
* Flask documentation [Электронный ресурс] Режим доступа: <http://flask.pocoo.org/docs/1.0/> - Дата доступа 08.12.2018
* Deep learning for humans [Электронный ресурс] Режим доступа: <https://github.com/keras-team/keras> - Дата доступа 08.12.2018