ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«Университет «Дубна»

ИНСТИТУТ СИСТЕМНОГО АНАЛИЗА И УПРАВЛЕНИЯ

Кафедра системного анализа и управления

**КУРСОВАЯ РАБОТА**

по дисциплине

**«Теория принятия решений»**

**ТЕМА: Принятие решений в задаче распознавания образов с использованием нейронной сети**

2255

Выполнил: студент группы \_\_\_\_\_\_\_\_\_

Чернядьев Кирилл Дмитриевич

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

Руководитель:

по дисциплине ТПР

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Ст.преп. Бархатова И.А

(ученая степень, ученое звание, занимаемая должность, ФИО)

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Дубна, 2023

**Содержание**

[**Введение** 3](#_Toc135742135)

[**Постановка задачи** 4](#_Toc135742136)

[**Теоретическая часть** 5](#_Toc135742137)

[Нейронная сесть 5](#_Toc135742138)

[Для чего нужны нейронные сети 5](#_Toc135742139)

[Нейрон и нормализация 5](#_Toc135742140)

[Синапс 6](#_Toc135742141)

[Как работает нейронная сесть? 6](#_Toc135742142)

[Функция активации 7](#_Toc135742143)

[**Линейная функция** 7](#_Toc135742144)

[**Сигмоид** 8](#_Toc135742145)

[**Гиперболический тангенс** 9](#_Toc135742146)

[Тренировочный сет 9](#_Toc135742147)

[Итерация 9](#_Toc135742148)

[Эпоха 9](#_Toc135742149)

[Ошибка 9](#_Toc135742150)

[**Практическая часть** 10](#_Toc135742151)

[Формулировка задачи 10](#_Toc135742152)

[Реализация 10](#_Toc135742153)

[**Оценка полученных результатов** 19](#_Toc135742154)

[**Заключение** 20](#_Toc135742155)

[**Список литературы** 21](#_Toc135742156)

# **Введение**

В настоящее время распознавание образов играет все более важную роль в различных сферах человеческой деятельности. Людям ежедневно приходится принимать серьезные решения, основанные на анализе и классификации больших объемов данных, включая изображения, звуковые сигналы, тексты и другие типы информации. С развитием технологий и доступностью огромных объемов данных возникает потребность в автоматизации процесса принятия решений, чтобы облегчить задачу и улучшить результаты.

Одним из эффективных подходов к автоматизации распознавания образов является использование нейронных сетей. Нейронные сети - это математические модели, которые имитируют работу человеческого мозга и способны обрабатывать и анализировать сложные данные. Они состоят из множества взаимосвязанных искусственных нейронов, которые обмениваются информацией и принимают решения на основе полученных данных.

Цель данной курсовой работы заключается в исследовании и выяснении эффективности нейронных сетей в задаче принятия решений в распознавании образов.

# **Постановка задачи**

**Цель работы:** определить, являются ли нейронные сети эффективным решением задачи принятия решения в распознавании образов.

**Исходные данные:** рукописные цифры.

**Модельные представления:** нейронная сеть для распознавания рукописных цифр.

**Средства реализации:** язык программирования *Python*, среда программирования *Visual Studio*.

**Ожидаемый результат:** обученная нейронная сеть, которая распознает образы рукописных цифр.

**Критерий оценки результата:** нейронная сеть распознает образы рукописных цифр с точностью более 80%.

# **Теоретическая часть**

## Нейронная сесть

**Нейронная сеть** — это последовательность нейронов, соединенных между собой синапсами.

Структура нейронной сети пришла в мир программирования прямиком из биологии. Благодаря этой структуре, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти. Другими словами, нейронная сеть является машинной интерпретацией мозга человека, в котором находятся миллионы нейронов, передающих информацию в виде электрических импульсов.

## Для чего нужны нейронные сети

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространёнными применениями нейронных сетей являются:

1. **Классификация** — распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них дать кредит, а кому — нет. Эту работу может сделать нейронная сеть, анализируя такую информацию, как: возраст, платежеспособность, кредитная история и т.д.
2. **Предсказание** — возможность предсказать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.
3. **Распознавание** — в настоящее время самое широкое применение нейронных сетей. Используется в Google, когда пользователь ищет фото, или в камерах телефонов, когда распознавание определяет расположение лица человека.

## Нейрон и нормализация

Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше.

Они делятся на три основных типа: входной, скрытый и выходной. Существуют еще нейрон смещения и контекстный нейрон. В том случае, когда нейронная сеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, *n*-скрытых слоев (обычно таких слоев не больше трех), которые обрабатывают входной слой, и выходной слой, который выводит результат (см. рис. 1).



Рис. 1. Перцептрон

У каждого из нейронов есть два основных параметра: входные данные (*input data*) и выходные данные (*output data*). В случае входного нейрона: *input* = *output*, а в остальных в поле *input* попадает суммарная информация всех нейронов с предыдущего слоя, после чего она нормализуется и с помощью функции активации попадает в поле *output*.

Важно помнить, что нейроны оперируют числами в диапазоне от [0, 1] или от [-1, 1]. Но тогда возникает проблема обработки чисел, которые выходят из этого диапазона. На данном этапе такая проблема решается путь деления единицы на нужное число. Этот процесс называется **нормализацией**, и он очень часто используется в нейронных сетях.

## Синапс

**Синапс** — это связь между двумя нейронами. У синапсов есть один параметр —вес. Благодаря весу, входная информация изменяется, когда передается от одного нейрона к другому.

Допустим, есть три нейрона, которые передают информацию следующему. Получается, что у нас есть три веса, соответствующие каждому из этих нейронов. У того нейрона, у которого вес будет больше, хранящаяся в нем информация будет доминирующей в следующем нейроне. На самом деле, совокупность весов нейронной сети или матрица весов — это своеобразный мозг всей системы. Именно благодаря этим весам, входная информация обрабатывается в результат.

## Как работает нейронная сесть?

В данном примере (см. рис. 2) изображена часть нейронной сети, где буквами «I» обозначены входные нейроны, буквой «H» — скрытый нейрон, а буквой «W» — веса.

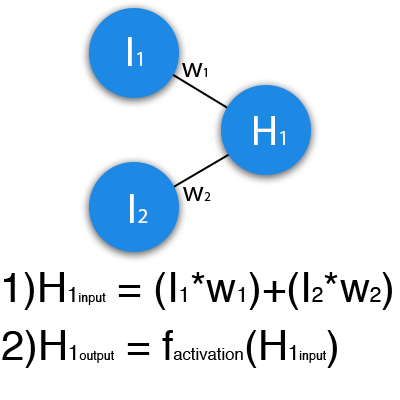


Рис. 2. Часть нейронной сети

Из формулы на рисунке видно, что входная информация — это сумма всех входных данных, умноженных на соответствующие им веса. Тогда дадим на вход 1 и 0. Пусть . Входные данные нейрона будут следующими: . Теперь, когда у нас есть входные данные, мы можем получить выходные данные, подставив входное значение в функцию активации. Имея выходные данные, мы передаем их дальше. Таким образом, мы повторяем до всех слоев, пока не дойдем до выходного нейрона.

Запустив такую сеть в первый раз, мы увидим, что ответ далек от правильного, потому сеть не натренирована. Чтобы улучшить результаты мы будем тренировать нейронную сеть.

## Функция активации

Функция активации — это способ нормализации входных. Если на входе будет большое число, пропустив его через функцию активации, вы получите вход в нужном вам диапазоне. Функций активаций достаточно много, поэтому мы рассмотрим самые основные: линейная, сигмоид (логистическая) и гиперболический тангенс. Главные их отличи — это диапазон выходных значений.

### **Линейная функция**

Линейная функция практически никогда не используется, за исключением случаев, когда нужно протестировать нейронную сеть или передать значение без преобразований (см. рис. 3).

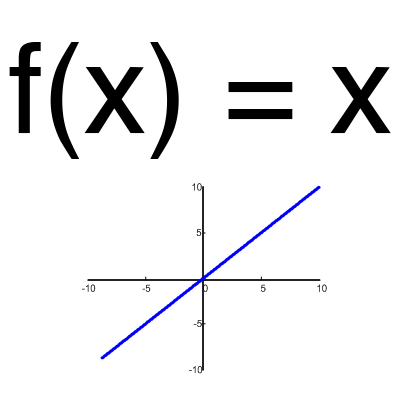


Рис. 3. Линейная функция

### **Сигмоид**

Это самая распространенная функция активации, ее диапазон значений от нуля до единицы. Иногда ее называют логистической функцией, и именно на ней показано большинство примеров в сети (см. рис. 4).

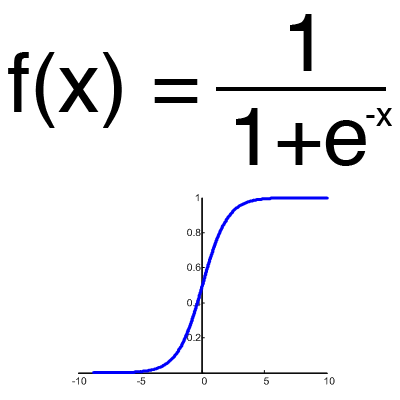


Рис. 4. Сигмоид

Если в нашем случаем будут присутствовать отрицательные значения (например, акции могут идти не только вверх, но и вниз), то вам понадобиться функция, которая захватывает и отрицательные значения.

### **Гиперболический тангенс**

Имеет смысл использовать гиперболический тангенс только тогда, когда значения должны быть и отрицательными, и положительными, так как диапазон функции от -1 до +1. Использовать эту функцию только с положительными значениями нецелесообразно, так как это значительно ухудшит результаты работы нейросети (см. рис. 5).

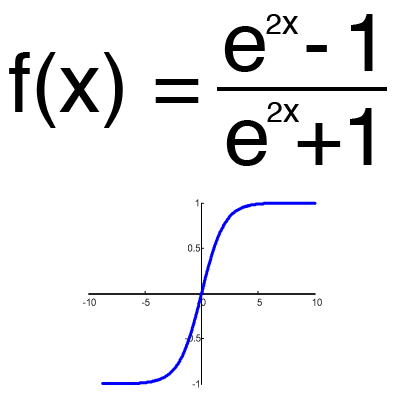


Рис. 5. Гиперболический тангенс

## Тренировочный сет

Тренировочный сет — это последовательность данных, которыми оперирует нейронная сеть.

## Итерация

Итерация — это своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Другими словами, это общее количество тренировочных сетов, пройденных нейронной сетью.

## Эпоха

При инициализации нейронной сети эта величина устанавливается в нуле и имеет потолок, задаваемый вручную. Чем больше эпоха, тем лучше натренирована сеть и, соответственно, лучше ее результат. Эпоха увеличивается каждый раз, когда мы проходим весь набор тренировочных сетов.

## Ошибка

Ошибка — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами.

# **Практическая часть**

## Формулировка задачи

Нам необходимо реализовать и обучить нейронную сеть для распознавания рукописных цифр, а также проверить ее работоспособность: найти возможные проблемы работы нашего приложения, дать оценку и вынести вердикт, является ли данная реализация эффективным решением нашей задачи.

## Реализация

Для данной курсовой работы приложение будет реализовано на языке программирования *Python* с использованием таких библиотек, как *matplotlib* и *keras* (см. рис. 9).

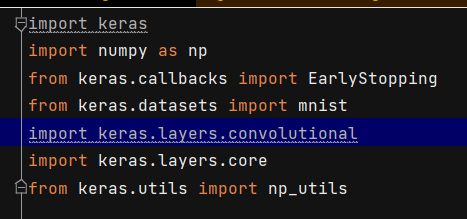


Рис. 9. Используемые библиотеки

Структура нейронной сети выглядит следующим образом: шесть слоев, из которых первые два слоя будут иметь тип *Conv2D*, третий слой — *Flatten*, а оставшиеся три слоя будут типа *Dense*.

*Conv2D* создает ядро свертки, которое помогает получить тензор выходных данных. Данный слой работает с входными изображениями, которые рассматриваются как двумерные матрицы. При обработке изображений ядро представляет собой сверточную матрицу или маски, которые можно использовать для размытия, повышения резкости, тиснения, обнаружения краев и т.п., путем выявления свертки между ядром и изображением.

Слой имеет следующие характеристики:

* 32 — это число ядер свертки;
* *kernel\_size* = 3 — размер ядра 3х3;
* функция активации — *relu* (*Rectified Linear Activation*)

Также в реализации используется функция *MaxPolling2D* [1], эта функция сжимает картинку или слой свертки, при этом группа пикселей (обычно размера 2х2) уплотняется до одного пикселя (см. рис. 10). Это позволяет намного упростить вычислительный процесс и тем самым увеличивает скорость работы сети. К тому же, это позволяет увеличить область, которую захватывает ядро свертки, в два раза, переходя от маленьких деталей изображения к более крупным, и объединить карты признаков, полученных сверткой, в более абстрактные признаки (не в пиксели, а в черточки).



Рис. 10. Визуализация работы метода *MaxPooling2D*

Далее в коде описывается ещё один свёрточный слой *Conv2D*. После идёт слой flatten, этот слой преобразует *2D*-данные в *1D*-данные. Следом идёт три полносвязных слоя *dense*. Последний из них имеет 10 выходных сигналов (вероятностей), индексы которых соответствуют распознанным цифрам.

В итоге мы получаем такую архитектуру нейронной сети (см. рис. 11)

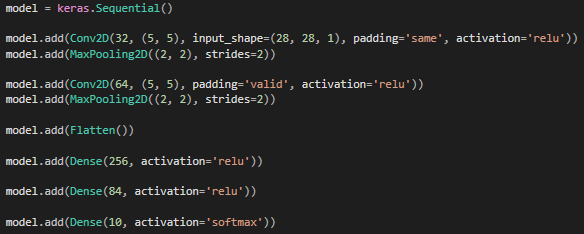


Рис. 11. Архитектура нейронной сети

На этом описание архитектуры нейронной сети заканчивается. Далее мы будем обучать нашу нейронную сеть.

Первый этап обучения это – подбор данных. Для этого в библиотеке *keras* есть готовый набор данных *mnist*, он содержит 60000 картинок с цифрами.

Сначала мы загрузим в память набор данных *mnist*, после создадим два массива (*X\_train*, *X\_test*). Элементами этих массивов будут числа от 0 до 255 (значения яркости пиксела). Переведём эти массивы в тип *float32*, разделим на 255, таким образом отнормируем данные в диапазоне от 0 до 1. Далее сформируем еще два массива (*Y\_train*, *Y\_test*), так называемые метки, грубо говоря, массивы правильных ответов, для сверки с результатами работы нейронной сети. Они представляют из себя некое количество массивов, в которых находится по десять значений (вероятностей), по одному на каждую цифру.

В итоге получим такой блок кода (см. рис. 12).

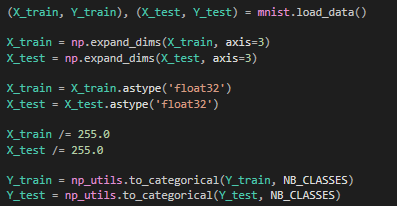


Рис. 12. Архитектура нейронной сети

Далее мы будем собирать и обучать написанную нами сеть. Для этого мы будем использовать следующие методы:

1. *model.summary*() – этот метод, при обучении сети, выводит в терминале ее структуру (см. рис.13)

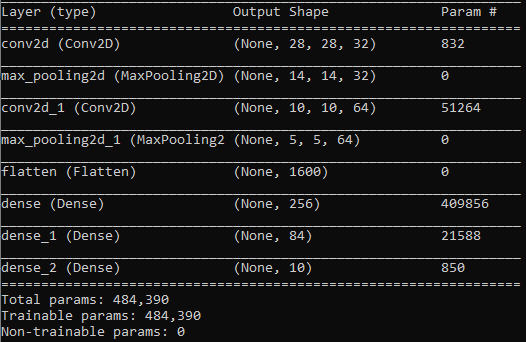
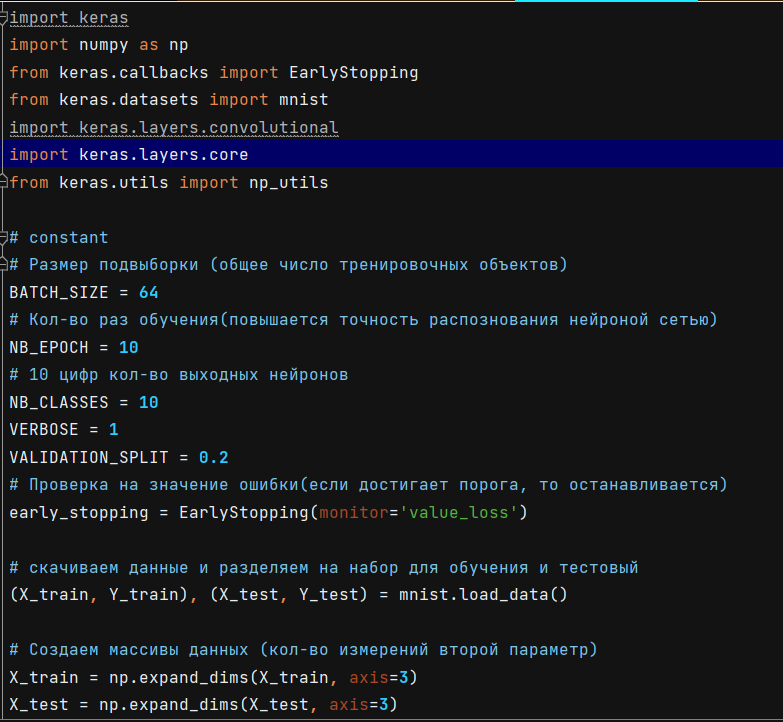


Рис. 13. Архитектура нейронной сети

1. *model.compil* [3] – этот метод компилирует нейронную сеть (собирает её в одно целое). Для работы ему необходимы следующие аргументы:
   1. *loss*: это функция, которая минимизируется в процессе обучения. В нашем случае используется *categorical\_crossentropy*.
   2. *optimizer*: Этот объект определяет процедуру обучения. Алгоритмы оптимизации минимизируют функцию потерь. Существуют такие оптимизаторы как: [*SGD*](https://keras.io/api/optimizers/sgd), [*RMSprop*](https://keras.io/api/optimizers/rmsprop), [*Adam*](https://keras.io/api/optimizers/adam), [*Adadelta*](https://keras.io/api/optimizers/adadelta), [*Adagrad*](https://keras.io/api/optimizers/adagrad), [*Adamax*](https://keras.io/api/optimizers/adamax), [*Nadam*](https://keras.io/api/optimizers/Nadam), [*Ftrl*](https://keras.io/api/optimizers/ftrl) [4]. В нашем случае используется оптимизатор [*Adam*](https://keras.io/api/optimizers/adam).
   3. *metrics*: используется для мониторинга обучения. Она оценивает качество работы алгоритма. Мы выбрали метрику *accuracy* – доля правильных ответов алгоритма.
2. *model.fit* [3] – этот метод как раз и обучает саму нейронную сеть. Ее параметрами являются:
   1. *X\_train* и *Y\_train* (данные для обучения и ответы соответственно)
   2. *batch\_size* – размер подвыборки. При обучении все количество картинок делится на мелкие группы размером равным значению *batch\_size*.
   3. *epochs* – количество эпох обучения (одна эпоха – это одна итерация по всем входным данным).
   4. *validation\_split* – доля обучающей выборки, которая будет отдана под валидацию.
   5. *verbose* - устанавливая значение 0, 1 или 2, вы просто говорите, как вы хотите «увидеть» прогресс в обучении для каждой эпохи.
   6. *callbacks* – имеет большой функционал, но чаще всего используется для остановки обучения, в случае если ошибка на конкретном наборе данных перестала уменьшаться. Для этого используется аргумент *early stopping* (ранняя остановка) [5].
3. *model.evaluate* [3] – этот метод используется для тестирования нейронной сети после ее обучения. Этот метод получает на вход тестовую выборку вместе с метками для нее. Метрика была задана еще при подготовке к работе, так что больше ничего не нужно.
4. *model.save* – этот метод просто сохраняет обученную модель нейронной сети.

На этом написание нейронной сети заканчивается, и мы можем посмотреть полный код (см. рис. 14).



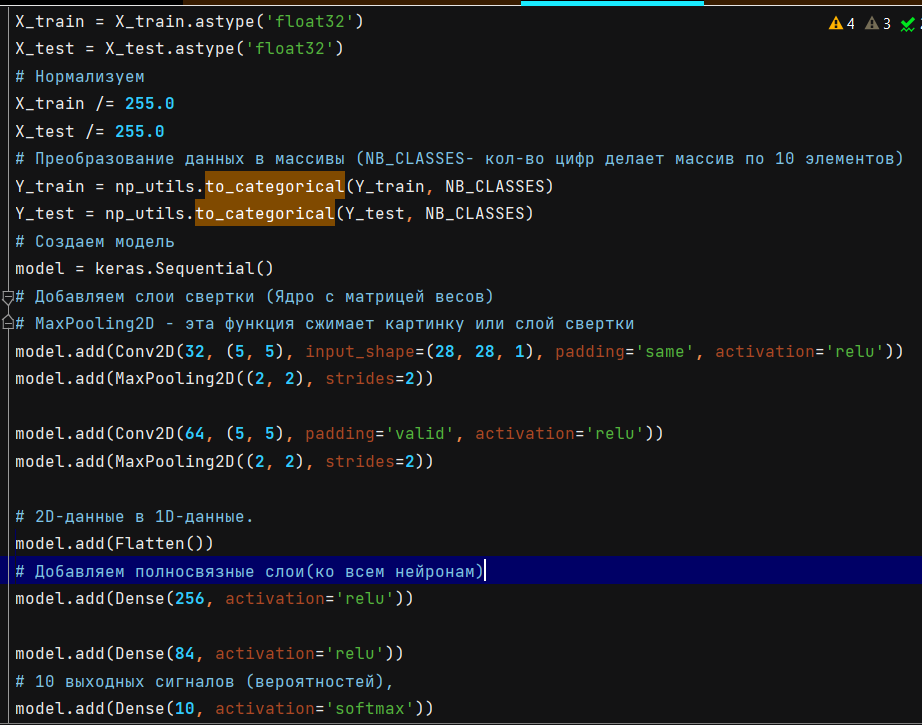




Рис. 14. Полный код нейронной сети

Далее мы запускаем этот код и ждём пока нейронная сеть обучится.

После обучения программа выдала вычисленную точность и сообщение, что наша обученная модель успешно сохранена (см. рис. 15).



Рис. 15. Точность нейронной сети

Далее посмотрим, как наше приложение распознает рукописные цифры (см. рис. 16 – 19)

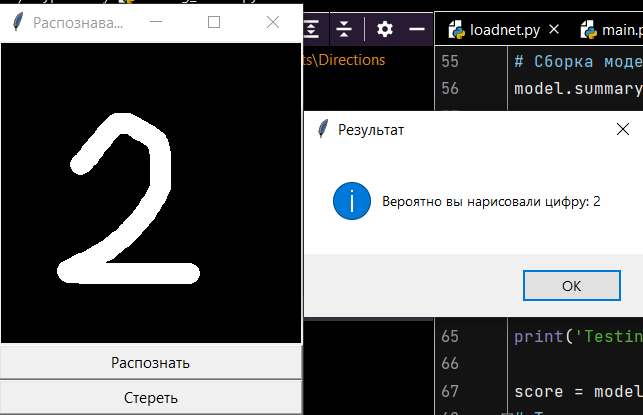


Рис. 16. Тест нейронной сети

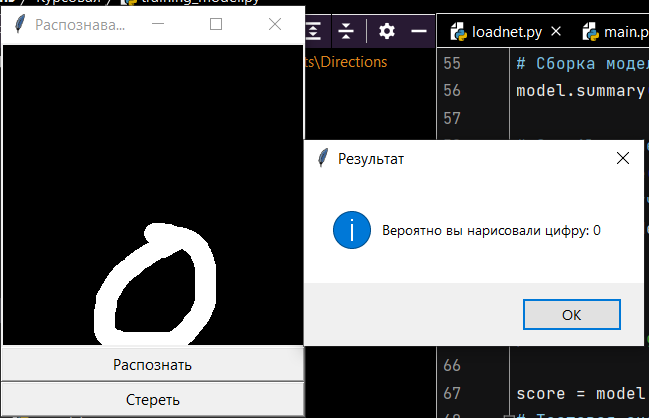


Рис. 17. Тест нейронной сети

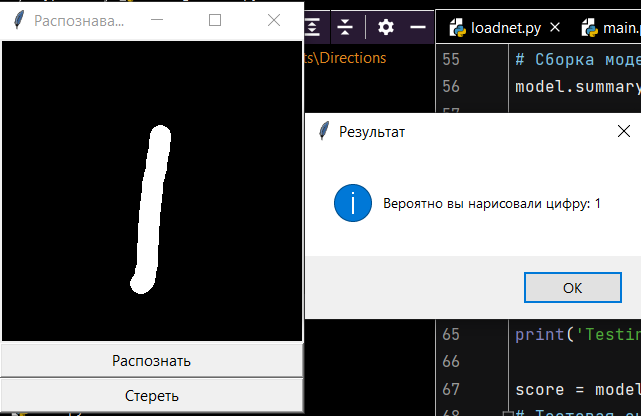


Рис. 18. Тест нейронной сети

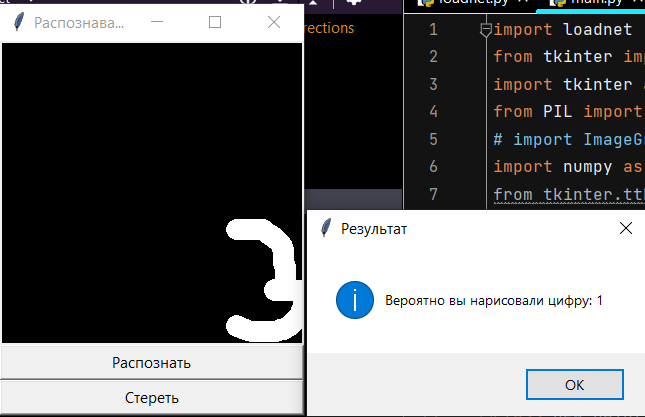


Рис. 19. Тест нейронной сети

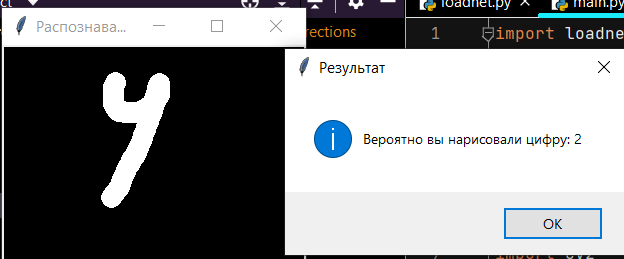


Рис. 20. Тест нейронной сети

# **Оценка полученных результатов**

Данная реализация нейронной сети имеет хорошую точность и скорость распознавания, но также имеет несколько недостатков:

1. Проблема распознавания цифр, нарисованных под большим наклоном. (рис.19)
2. Проблема распознавания цифр, нарисованных не «Идеально» (рис.20)
3. Данная реализация даёт возможность распознавать только по одному образу за раз.

Если учесть или исправить данные проблемы, то можно сказать, что данная реализация является эффективным решением нашей задачи.

# **Оценка дальнейшего совершенствования программы**

Данная программа имеет достаточно дальнейших перспектив, среди которых:

1. Возможность добавления голосового ассистента, который будет озвучивать цифры, которые нарисованы.
2. Возможность добавления распознавания картинок, добавленных пользователем.
3. Возможность добавления взаимодействия с распознанными объектами, например складывать цифры, которые были нарисованы.

# **Заключение**

В данной работе были разобраны основные понятия в сфере нейронных сетей, изучены принципы их работы, рассмотрены примеры распознавания нейронной сетью образов рукописных цифр, используя разработанное нами приложение. Был сделан вывод, что нейронные сети – это эффективное решение задачи распознавания образов в теории принятия решений.

# **Список литературы**

1. Хабр, Применение нейросетей в распознавании изображений//Habr URL: <https://habr.com/ru/articles/74326/> (дата обращения: 20.05.2023)
2. TensorFlow // Helenkapatsa URL: <https://www.helenkapatsa.ru/tensorflow/> (дата обращения: 20.05.2023).
3. Keras // Habr URL: <https://habr.com/ru/articles/482126/> (дата обращения: 20.05.2023).
4. Машинное обучение для начинающих, URL: <https://python-scripts.com/intro-to-neural-networks> (дата обращения: 20.05.2023)
5. Нейронная сеть на практике с Python и Keras // URL: <https://pythonru.com/primery/nejronnaja-set-na-praktike-s-python-i-keras> (дата обращения 20.05.2023)
6. Ссылка на GitHub // URL: <https://github.com/52herzz/TPR>