МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»

(СПбГУТ)

Кафедра «Информационных управляющих систем»

**ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ № 3**

ПО ДИСЦИПЛИНЕ «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

«СОЗДАНИЕ НЕЙРОСЕТИ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ ЦИФР»

|  |  | Выполнили студенты группы ИСТ-312  Аджигирей Д. С., Кандиков М. В., Фомина Я. А., Новожилова Д. А., Чистякова Д.А. | |
| --- | --- | --- | --- |
|  |  | Фамилия И. О. | |
|  |  | Руководитель | ассистент |
| оценка |  |  | уч. степень, уч. звание |
|  |  | Климов А. М. | |
| дата, подпись |  | Фамилия И. О. | |

**Санкт-Петербург**

**2024**

СОДЕРЖАНИЕ

[**Введение** 2](#_heading=h.30j0zll)

[Этап I 3](#_heading=h.1fob9te)

[Этап II 3](#_heading=h.3znysh7)

[Этап III 4](#_heading=h.2et92p0)

[Этап IV 5](#_heading=h.tyjcwt)

[Этап V 7](#_heading=h.3dy6vkm)

[Заключение 14](#_heading=h.1t3h5sf)

## **Введение**

**Цель работы** – научиться работать со нейросетями путём её машинного обучения и тестирования, сделать её результаты более точными

Нейронные сети, рассматриваемые в лабораторной работе – класс алгоритмов машинного обучения. С их помощью удается достичь точных результатов в области распознавания образов, классификации изображений, а также обработки и анализа видеоданных.

В данной лабораторной работе рассматривается нейросеть описанного выше типа, специализирующаяся на распознавании цифр на наборе данных MNIST - базе данных, в которой хранятся образцы написания рукописных цифр (рис. 1). Для этого использовались среды программирования: Google Colab, Pycharm.



Рисунок 1 - примеры цифр из обучающей выборки библиотеки MNIST.

Перед началом работы мы разделили обязанности:

1. Диана Новожилова – написание отчёта по данным, полученным другими в течение обучения нейросети;
2. Дарья Чистякова – написание отчета, создание графика для наглядности полученного результата;
3. Дмитрий Аджигирей – обучение нейронной сети. Анализ данных при следующих значениях: количество эпох – 1500, валидация – 0,1, создание кода;
4. Максим Кандиков – обучение нейронной сети. Тестирование кода и его рефакторинг;
5. Яна Фомина – обучение нейронной сети. Анализ данных, обучение двух моделей: количество эпох – 2500, 4900; валидация – 0,2;0,3.

Итак, нам удалось проанализировать работу нейронной сети в деталях.

**Этап I**

Данный код был помещен в Google Colab, его следовало разделить на блоки. Стоит обратить на используемые библиотеки (рис. 2). NumPy – библиотека для работы с многомерными массивами, matplotlib – для визуализации данных, tensorflow – для машинного обучения искусственного интеллекта.

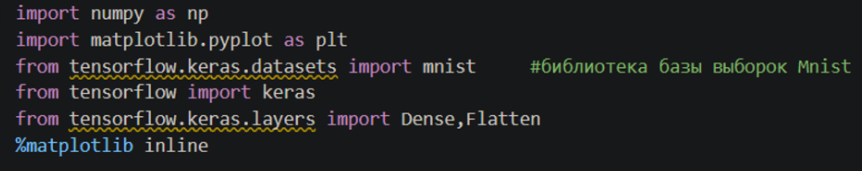
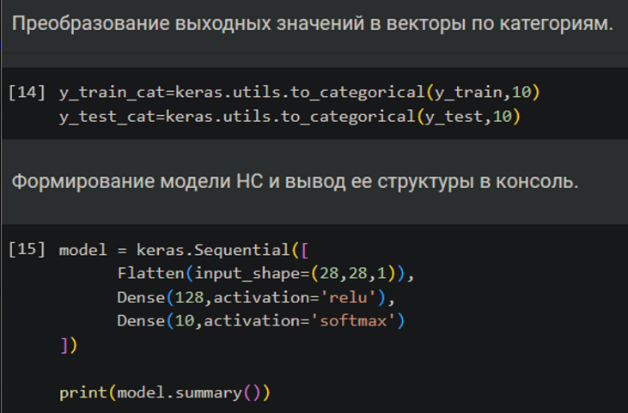


Рисунок 2 – библиотеки, участвовавшие в создании нейронной сети

Далее программа разделена на следующие блоки (рис 3, 4): обучающая и тестовая выборка, нормализация входных данных, преобразование выходных значений в векторы по категориям, формирование модели нейросети и вывод её структуры в консоль, компиляция модели (был использован алгоритм оптимизации adam).

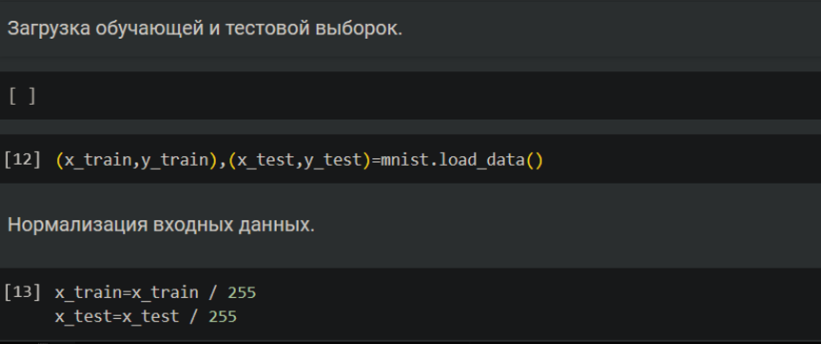


Рисунок 3, 4 – основной код, создающий основу нейросети

**Этап II**

Когда основной код написан, следующим шагом является обучение модели нейросети. Для этого была прописана строка, запускающая этот процесс (рис. 5). Количество эпох и кросс-валидация могут быть изменены для эффективной тренировки нейронной сети.

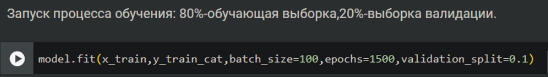
****

Рисунок 5 – код для обучения нейросети (с эпохами в количестве 1500, валидацией, равной 0,1)

После запуска этого блока можно наблюдать за процессом тренировки сети и за изменением значений на каждом этапе (рис. 6).

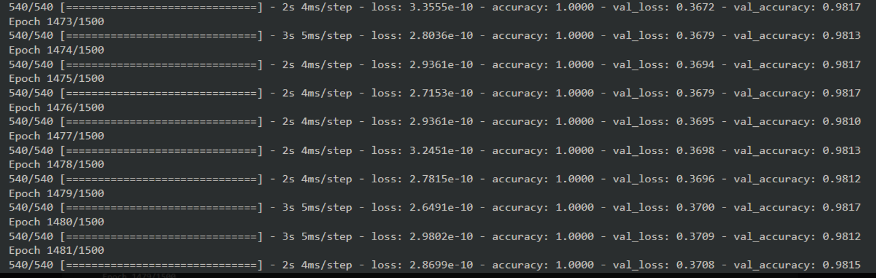


Рисунок 6 – запуск процесса обучения

**Этап III**

После завершения этапа обучения необходимо сохранить полученную нейронную сеть (рис. 7). Скачиваем файл с названием «16\_model.h5».

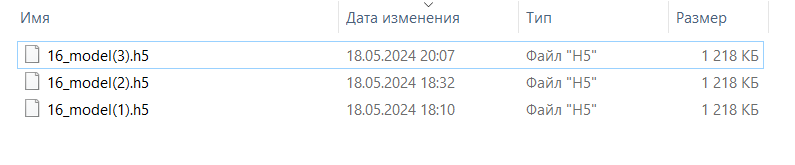


Рисунок 7 – результат сохранения нейронной сети

**Этап IV**

Далее переходим к созданию интерфейса. Используем среду разработки PyCharm. Создаем новый файл с расширением «.py» и названием «gui». Для начала подключаем необходимые библиотеки (рис. 8). «os» – для работы с операционной системой, «tkinter» – для графического интерфейса, «keras» – для взаимодействия с нейронными сетями, «win32gui» – для автоматизации gui процессов, «numpy» – для работы с многомерными массивами и матрицами.

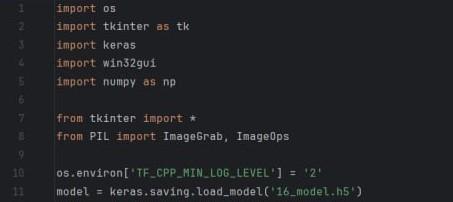


Рисунок 8 – подключение библиотек

Отметим, что первый запуск кода был неудачным: некорректно установились библиотеки win32gui и PIL в Python. Столкнулись с ошибкой «Import error: No module named tensorflow [duplicate]».

Для успешной установки win32gui была установлена библиотека pywin32 (с помощью команды pip install pywin32 в командной строке Windows). В ней содержались компоненты для запуска нужной директивы.

Аналогично была установлена еще одна библиотека pillow, позволившая решить проблему с PIL.

Для устранения последней ошибки воспользовались командой pip install tensorflow==1.2.0 –ingore-installed. Альтернативным решением могла являться загрузка пакет новейшей версии библиотеки в PyCharm.

Далее загружаем ранее сохранённую в файл модель нейронной сети (рис.9).



Рисунок 9 – загрузка файла с моделью

После этого переходим к функции преобразования и обработки изображения (рис. 10). Приводим всё к одному размеру, настраиваем отображение цвета и сохраняем в переменную res результат предсказания.

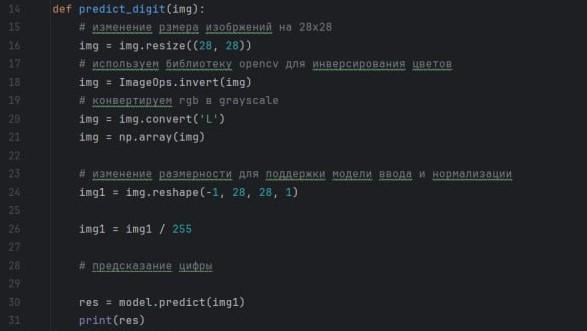
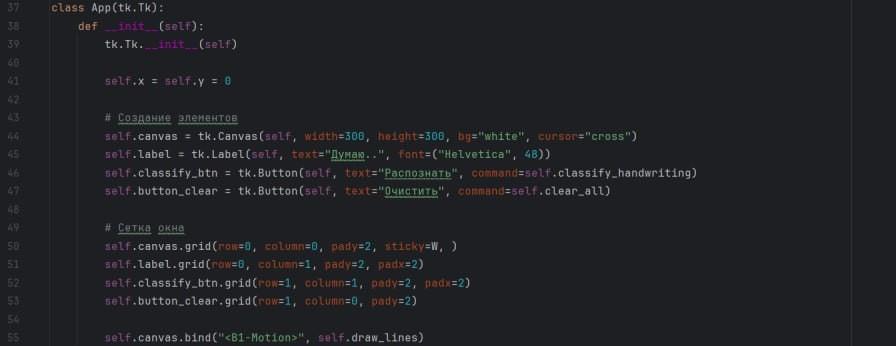
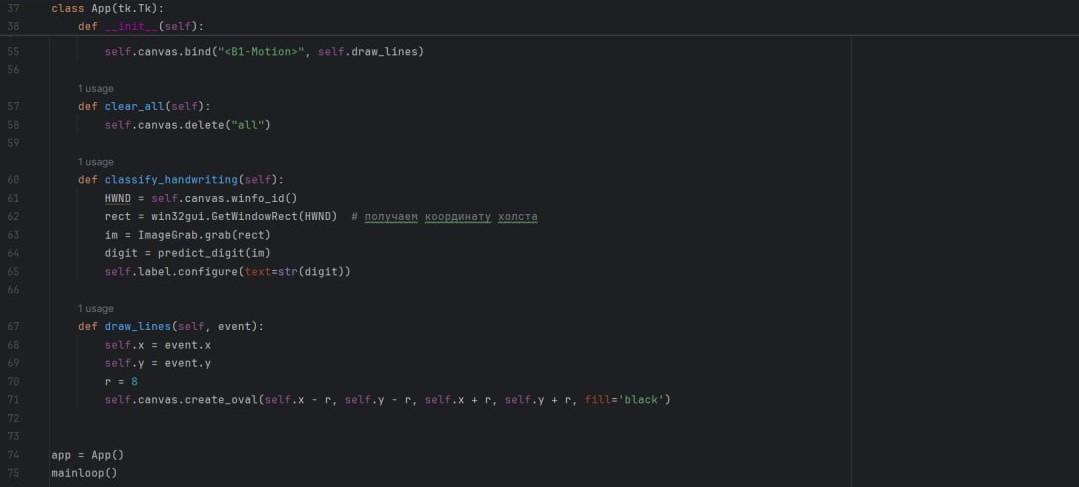


Рисунок 10 – функция преобразования и обработки изображения

Теперь, используя библиотеку tkinter, создаем графический интерфейс (рис. 11 и рис. 12). Создаём элементы отображения и сетку, настраиваем вывод результатов в окно.





Рисунки 11 и 12 – код для создания графического интерфейса

**Этап V**

После выполнения всех вышеперечисленных этапов переходим к тестированию моделей. Компилируем полученный код, в результате чего видим созданный интерфейс (рис. 13).

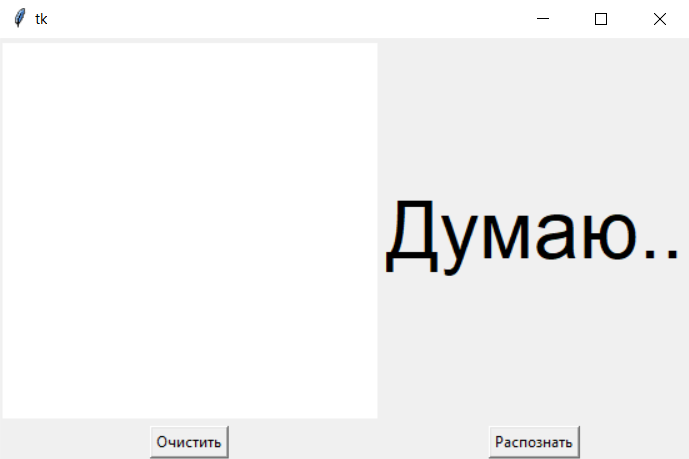


Рисунок 13 – графический интерфейс

Переходим непосредственно к тестированию работоспособности моделей. Используем тестовый набор «1, 3, 5, 7, 2», так как распознавание этих цифр наиболее затруднительно для данной нейронной сети. Введём числа четырьмя последовательными блоками.

Первая модель

Показатели: Epochs = 1500, validation\_split = 0,1. Результаты проверки приведены ниже (рис. 14, 15, 16, 17).

Первый блок:

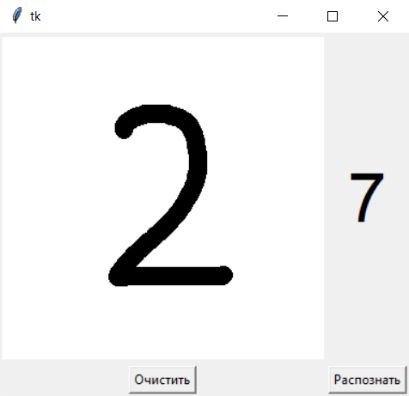
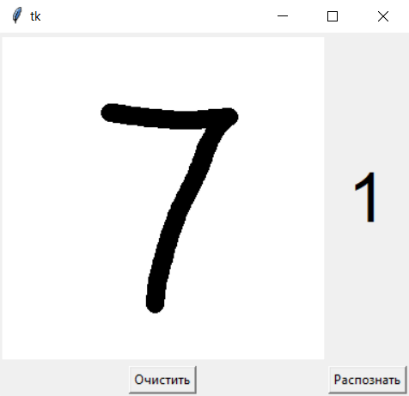
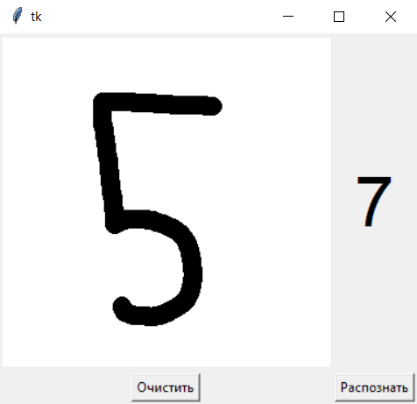
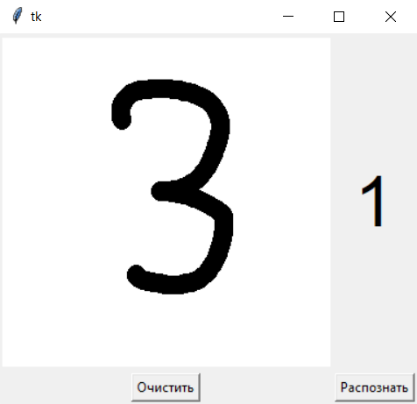
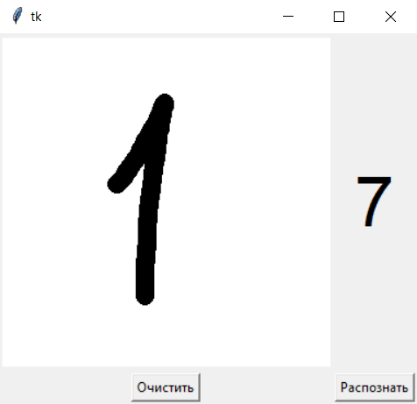


Рисунок 14 – первый блок тестирования первой модели

Второй блок:

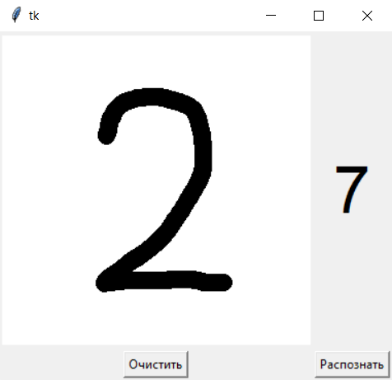
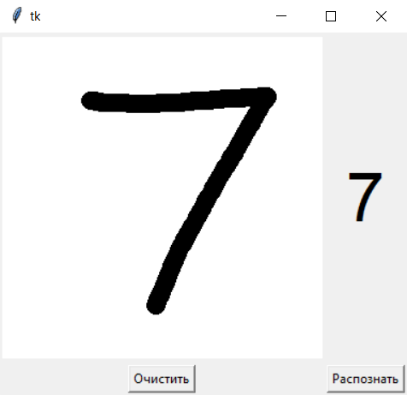
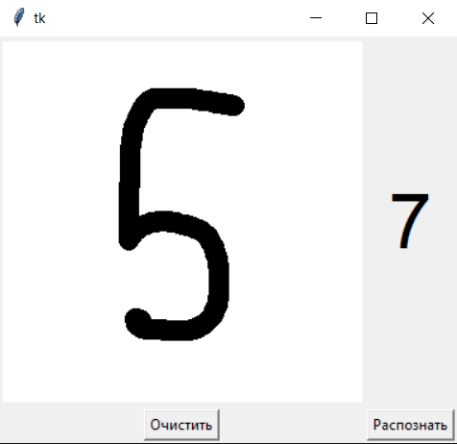
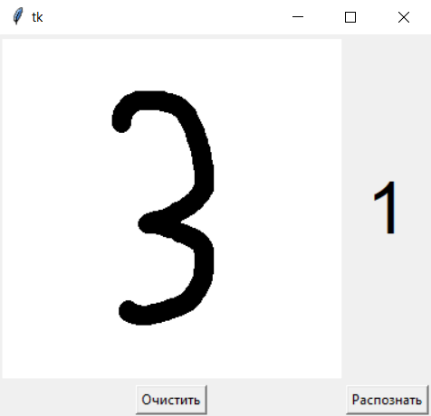
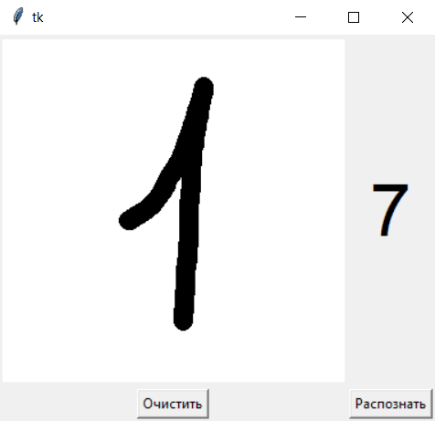


Рисунок 15 – второй блок тестирования первой модели

Третий блок:

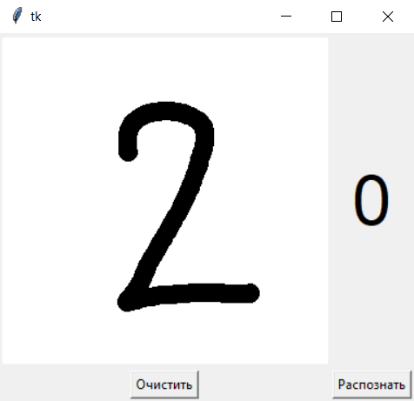
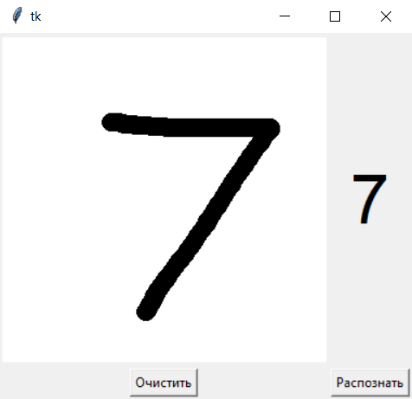
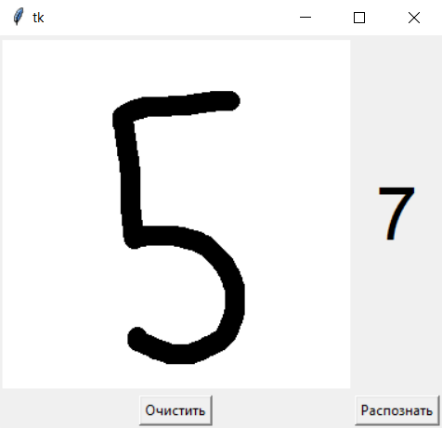
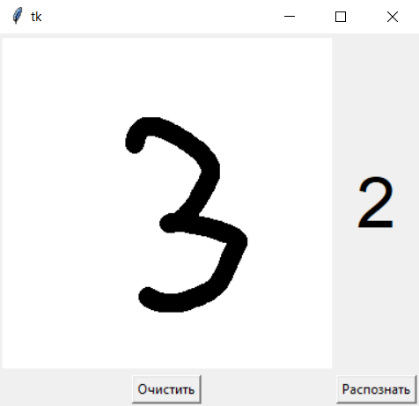
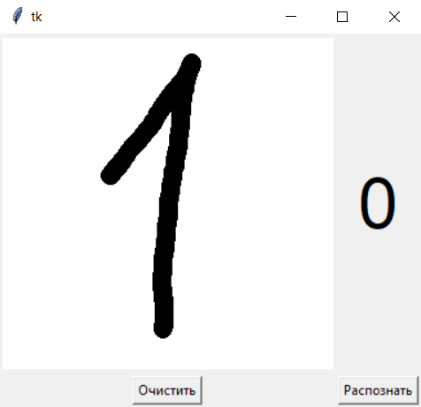


Рисунок 16 – третий блок тестирования первой модели

Четвёртый блок:

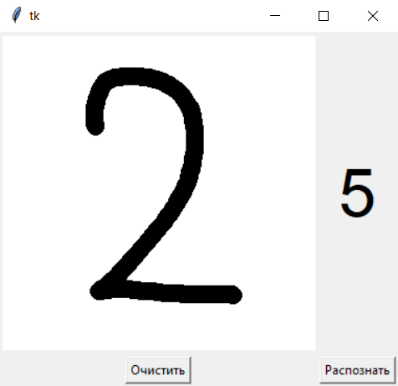
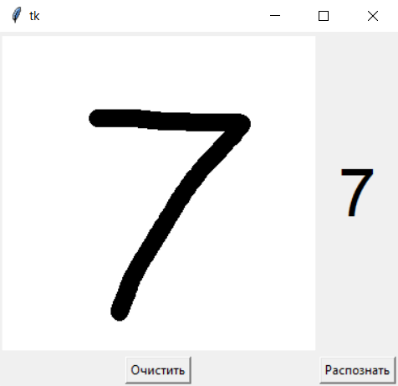
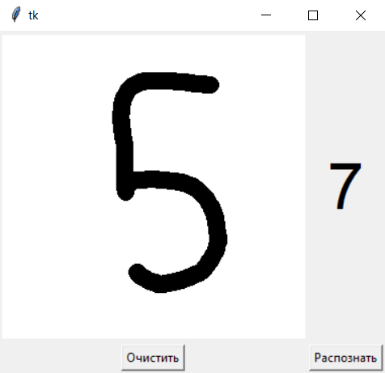
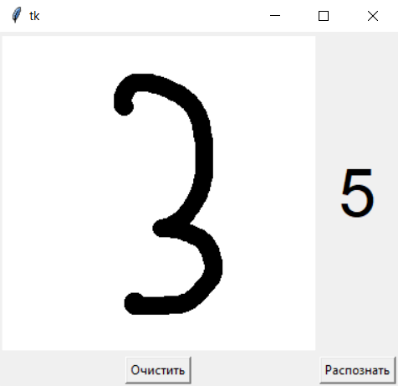
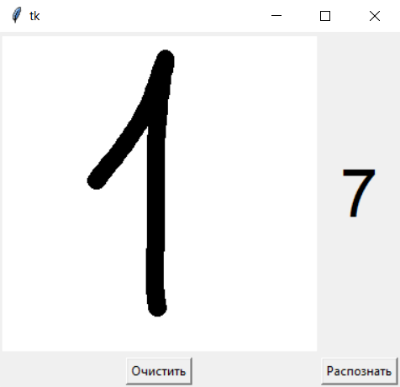


Рисунок 17 – четвертый блок тестирования первой модели

По итогу проверки видим, что “7” нейросеть распознала 3 раза, остальные рукописные цифры успешно считать не удалось.

Вторая модель

Показатели: Epochs = 2500, validation\_split = 0,2. Результаты проверки приведены ниже (рис. 18, 19, 20, 21).

Первый блок:

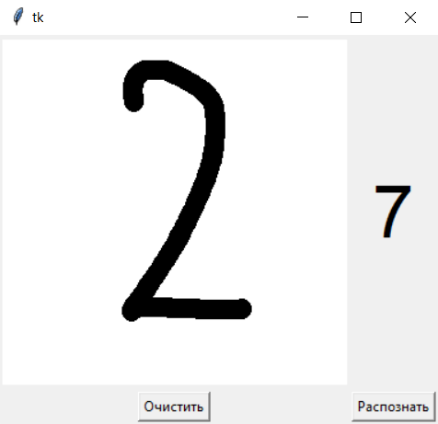
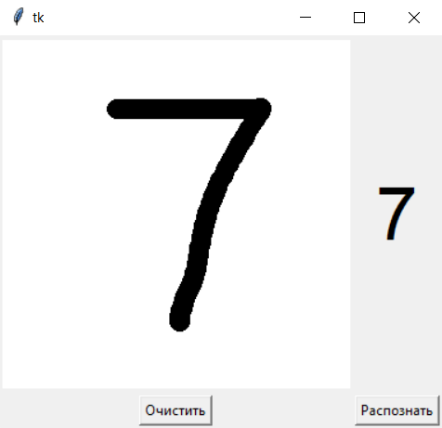
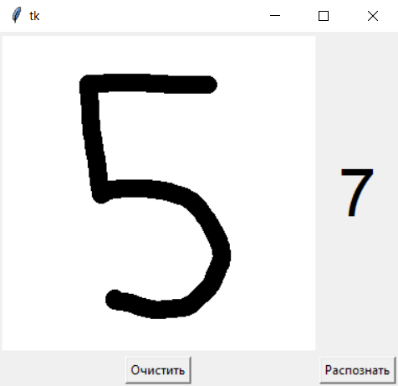
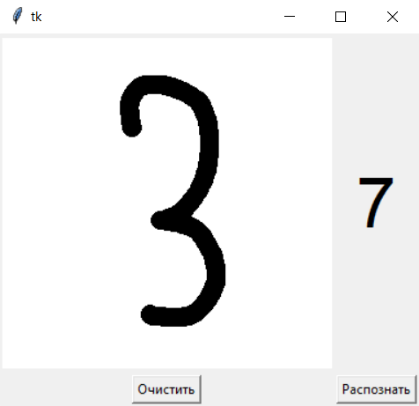
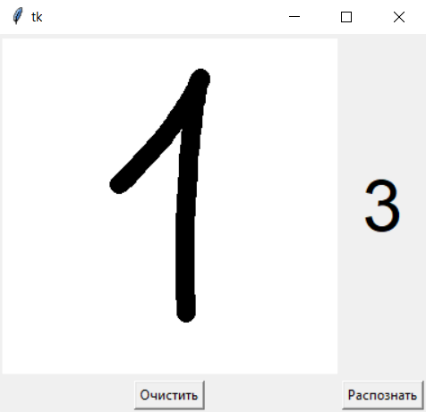


Рисунок 18 – первый блок тестирования второй модели

Второй блок:

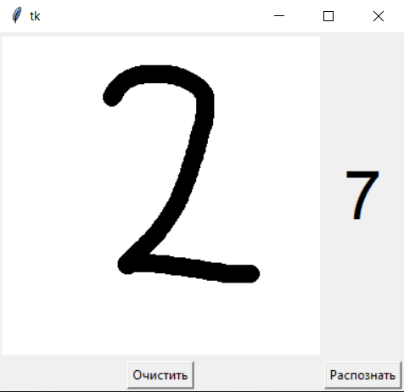
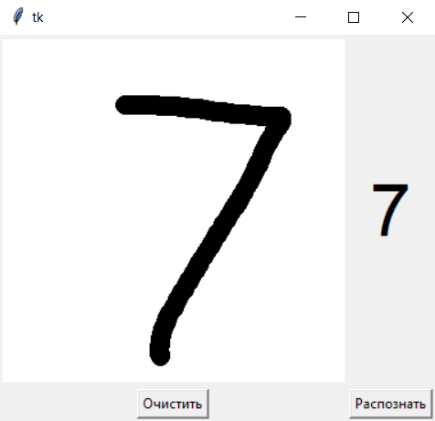
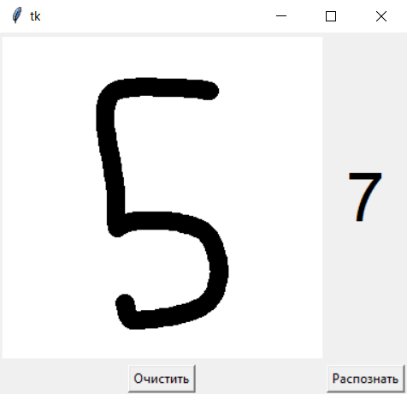
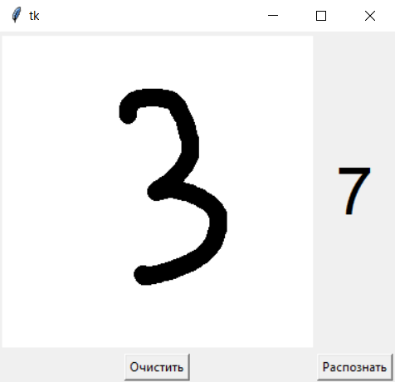
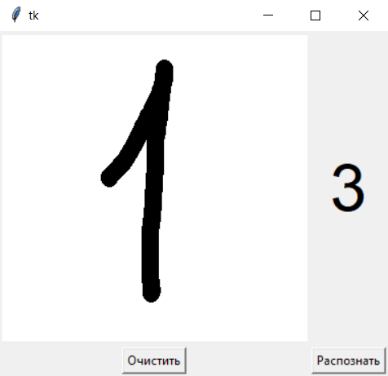


Рисунок 19 – второй блок тестирования второй модели

Третий блок:

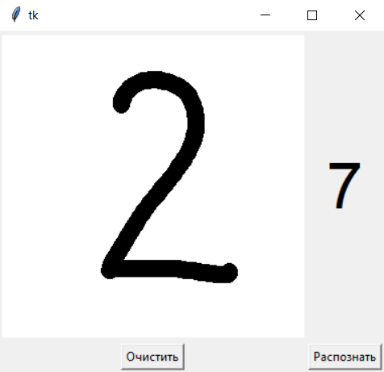
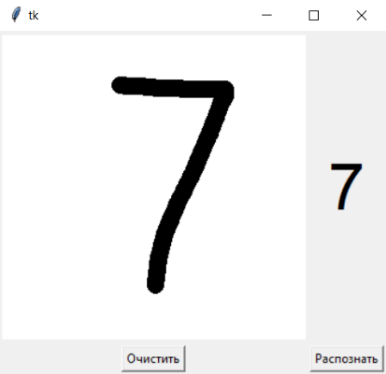
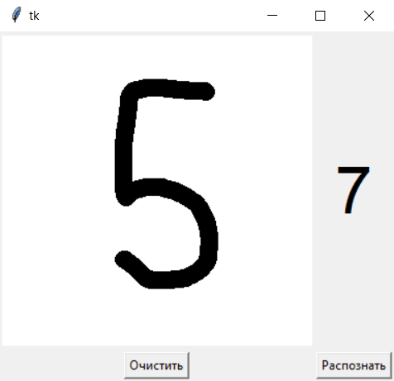
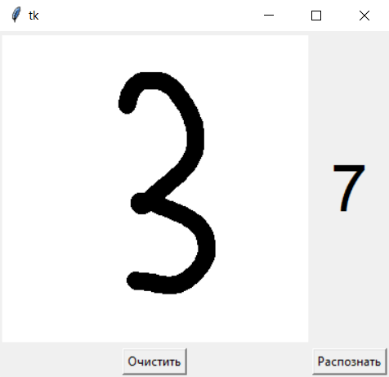
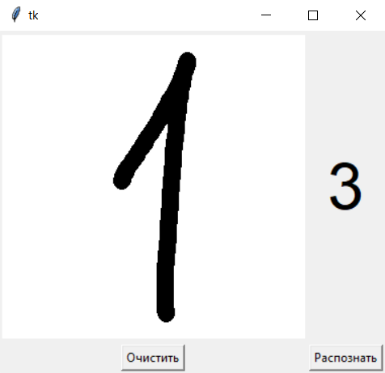


Рисунок 20 – третий блок тестирования второй модели

Четвёртый блок:

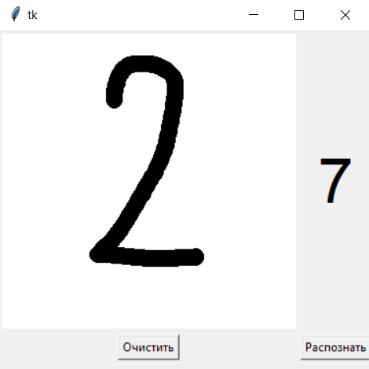
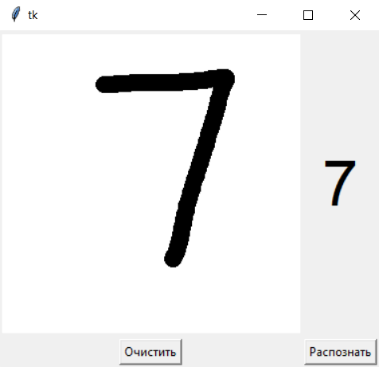
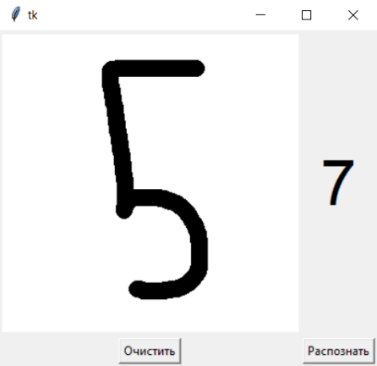
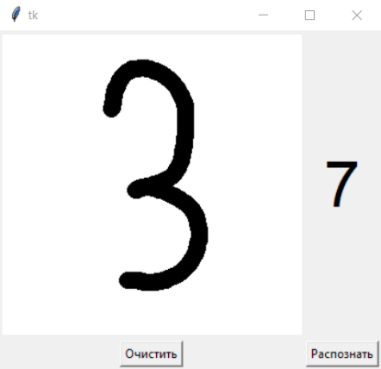
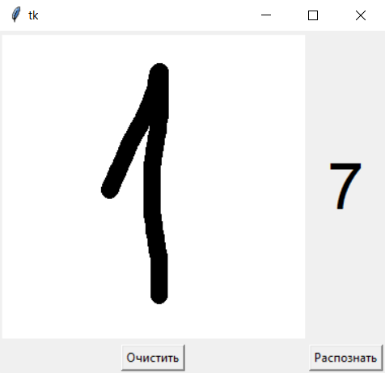


Рисунок 21 – четвёртый блок тестирования второй модели

В сравнении с первой моделью вторая показала схожие результаты, большинство цифр считать не удалось, но “7” была распознана 4 раза, что на 1 больше, чем при тестировании первой модели.

Третья модель

Показатели: Epochs = 4900, validation\_split = 0.3. Результаты проверки приведены ниже (рис. 22, 23, 24, 25).

Первый блок:

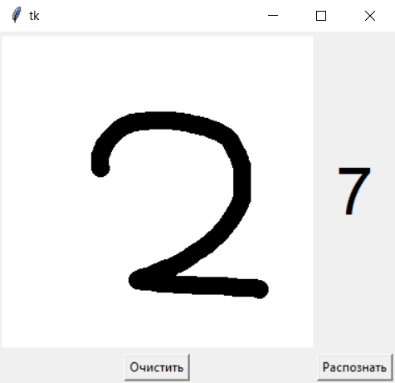
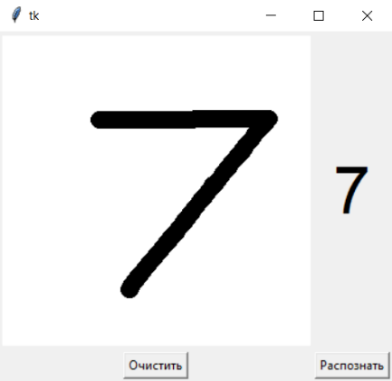
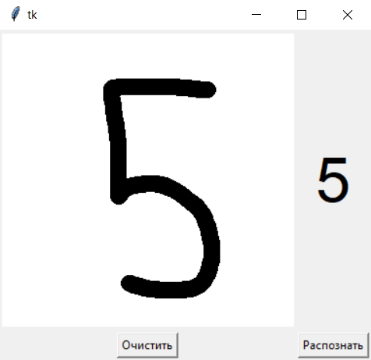
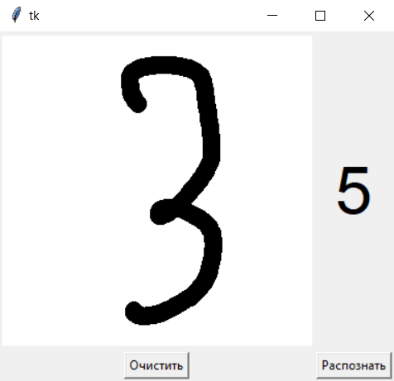
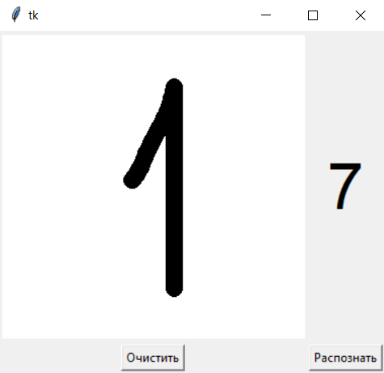


Рисунок 22 – первый блок тестирования третьей модели

Второй блок:

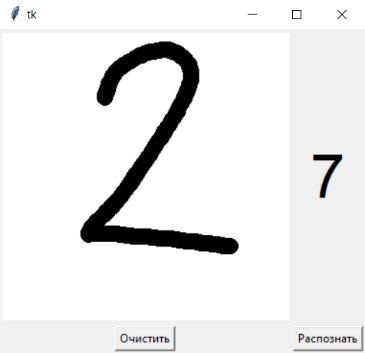
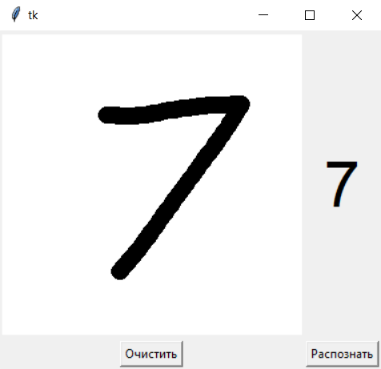
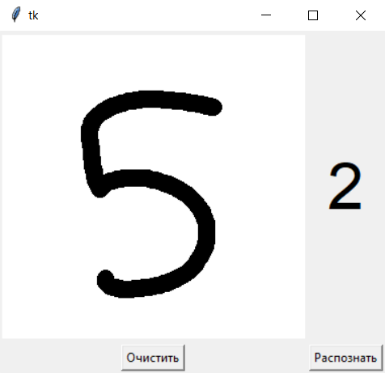
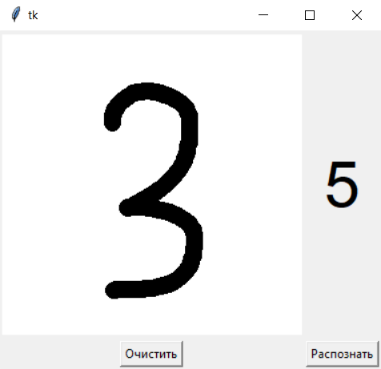
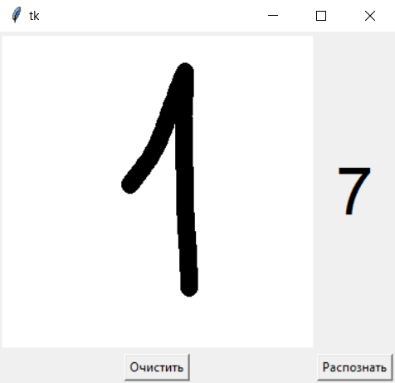


Рисунок 23 – второй блок тестирования третьей модели

Третий блок:

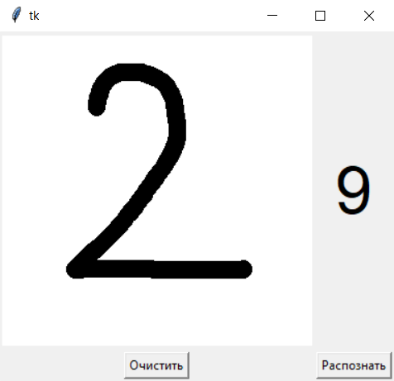
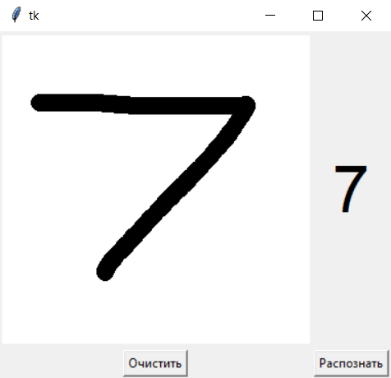
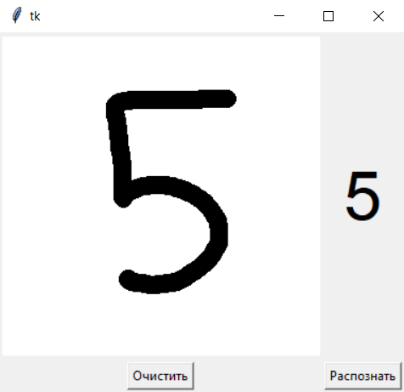
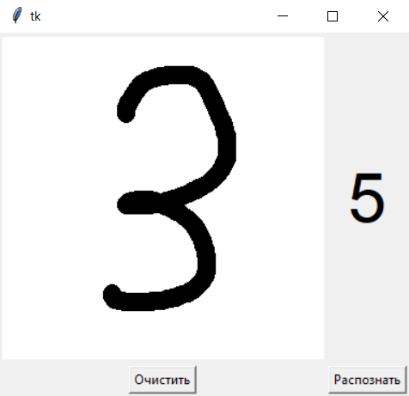
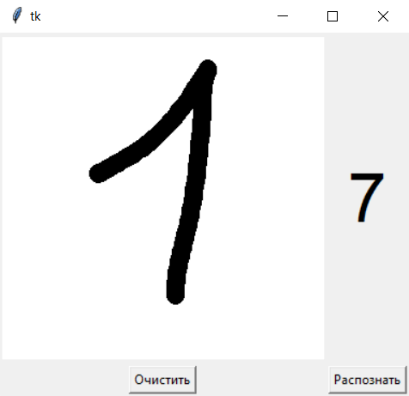


Рисунок 24 – третий блок тестирования третьей модели

Четвёртый блок:

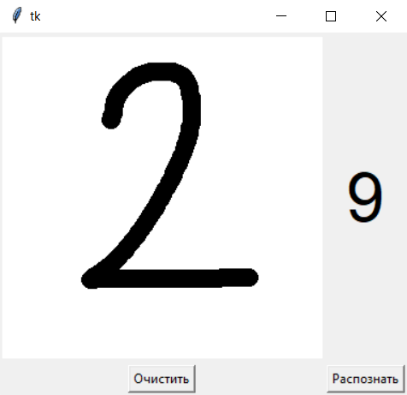
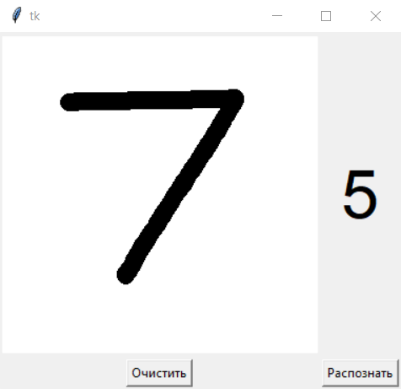
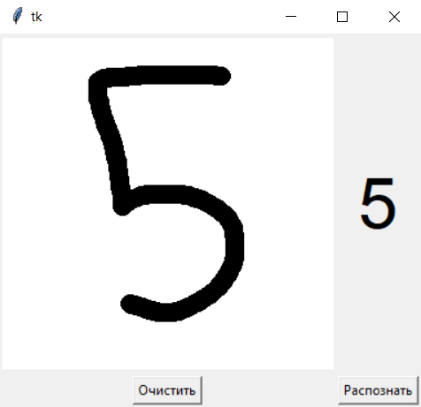
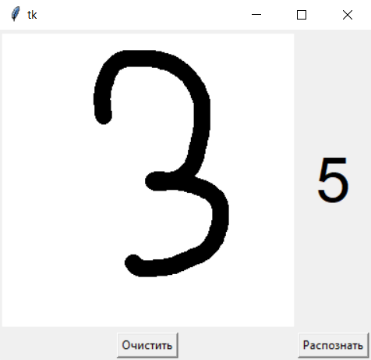
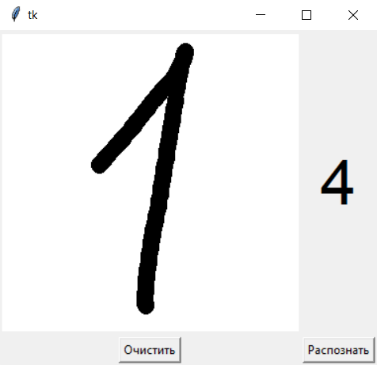


Рисунок 25 – четвёртый блок тестирования третьей модели

Третья модель показала лучшие результаты. “7” была распознана 4 раза, как и во второй модели, а “5” – 3 раза.

Для удобства сравнения и большей наглядности полученных при тестировании каждой из трёх моделей обучения данных составим график.

График 1 – Результат распознавания цифр тремя моделями

Стоит отметить, что представленные выше данные показывают результат на конкретном исполняющем компьютере, однако производительность нейронной сети зависит от ресурсов компьютера, поэтому при запуске на более мощном устройстве результаты будут более точными

**Заключение**

В ходе лабораторной работы была разобрана нейросеть, её принципы работы, применены навыки машинного обучения и последующего тестирования. Проанализировав работу нейронной сети, можно сделать вывод, что наилучшие результаты показывает нейросеть с показателями 4900 эпох и валидацией, равной 0,3, она выдаёт наибольшее количество распознаваний рукописных цифр. Худший результат видим у нейросети с 1500 эпохами и валидацией 0,1.

Продуманы рекомендации для улучшения работы нейросети: по наблюдениям большее количество эпох гарантирует более точные показатели, поэтому одно из решений – увеличить число проходимых эпох. Аналогично со значением валидации: чем больше, тем качественнее обучается сеть. Более того, можно увеличить базу данных угадываемых цифр, чтобы нейросеть распознавала различные рисунки, учитывая помехи.