**Документация по проекту DigitRecognizer *by Zaitseva&Kapranov***

Проект можно условно поделить на две части: приложение, написанное на PyQT5 и реализация распознавания рукописных цифр с помощью машинного зрения и tensorflow.

Для корректной работы программы необходимо иметь установленные библиотеки *tensorflow* **2.0.0 (!),** *cv2, h5py*и PyQT5.

Ход действий:

1. Скачать из репозитория [ksyu-zait](https://github.com/ksyu-zait)/[project\_2020](https://github.com/ksyu-zait/project_2020) файлы *mnist\_trainedCNN.py, test.py, window.py, web.jpg.*
2. Запустить файл *window.py.*
3. Нарисовать в белом окошке цифру с помощью мышки или тачпада.
4. Нажать кнопку *“Обработать изображение”*.
5. При выведении ошибки *“Нейросеть не обучена!”* закрыть диалоговое окно с ошибкой и нажать кнопку *“Обучить сеть”.* После получения диалогового окна *“Сеть обучена!”* закрыть его и снова нажать кнопку *“Обработать изображение”*. Обучение сети может занимать до десяти минут.

**Используемые библиотеки и функции:**

Для визуализации приложения использовалась библиотека PyQT5. Основное окно описано классом MainWind, унаследованным от класса QMainWindow. На нем установлена сетка для виджетов 7х2, первый столбец которой занимают строки (QLabel) и кнопки “Очистить”, “Обработать изображение” и “Обучить сеть” (QPushButton). Две последние связаны с методами класса MainWind recognition() и learn(). Метод recognition() сохраняет в папку с программой изображение размером 28х28 пикселей с названием “image.jpg”, далее с помощью конструкции “try… except…” запускаем распознавание цифры с картинки. При возникновении ошибки, связанной с необученной сетью, программа вызывает QMessageBox с ошибкой. Метод learn() запускает функцию learner(), обучающую сеть, а после окончания обучения выводит QMessageBox с подтверждением успешного обучения. Таким образом при нажатии “Обработать изображение” наша программа определяет цифру и вероятность успеха и выводит эти значения на основное окно. Второй столбец занимает виджет drawer, описанный одноименным классом, унаследованным от QWidget. Размер полотна 250х250 пикселей, цвет фона - белый. Также с помощью набора методов paintEvent(), mousePressEvent() и mouseMoveEvent() мы реализуем рисование на этом виджете кривой, которая появляется при нажатии ЛКМ. Ширина кисти для успешной работы сети была выбрана 20 пикселей. Полученное изображение хранится в виде объекта класса QImage. Также помимо основной функции программы, в окне есть меню, позволяющее закрывать предложение.

В части проекта, посвященной машинному зрению, используется CNN (сверточная нейронная сеть), реализованная библиотекой tensorflow. Для обучения используется датасет MNIST из 70000 рукописных цифр (белые на черном фоне). Для возможности обработки нейросетью производится нормировка, чтобы получить значения для каждого пикселя от 0 до 1. Нейросеть реализована с помощью архитектуры *keras* и состоит из двух сверточных слоев, двух слоев подвыборки (*pooling*) и двух разных функций активации - *ReLU* для отсечения отрицательных значений и уменьшения времени работы программы и *Softmax* для получения вектора вероятности каждого из десяти классов (цифры от 0 до 9). В качестве лосс-функции используется категорическая кросс-энтропия, типичная для определения класса, для определения оптимальных параметров сети используется градиентный спуск. Обучение происходит около 5 минут при одном цикле работы, примерная точность на тестовых данных MNIST около 98%.

Изначально предполагается, что точность распознавания наших данных будет ниже, чем для MNIST, так как рукописное написание цифр, на котором основан датасет обучения, отличается от рисования мышкой. Первичная обработка изображения происходит для приведения его к виду MNIST, т.е. форматирование в тензор 28х28, инверсия цвета для получения белых цифр на черном фоне и нормировка. После этого изображение обрабатывается нейросетью, на выходе мы получаем наиболее вероятный класс и вероятность успеха.