Результаты выполнения тестового задания

**1. Поиск рамок чёрного цвета разной ширины при помощи ИИ**

*Что использовано.* Для выполнения этого задания использованы такие библиотеки Python как OpenCV, SciPy, NumPy и некоторые другие стандартные библиотеки. Примерно за 1 час реализована наиболее простая версия программы, которая определяет на RGB-изображении черные области размера 3\*26 (размер массива – 3\*26\*3 пикселя).

Принцип работы программы – выполнение свертки исходного изображения с окном размера 3\*26\*3, заполненного единицами. По сути, результатом свертки является сумма всех значений пикселей окна. Условие отнесения фрагмента изображения к черному цвету самое простое – сумма значений пикселей должна быть равна 0. Подходящие под данное условие области изображения закрашиваются зеленым цветом и сохраняются в специальную папку.

Программа протестирована на изображениях из папки «Crops/60img/G32V3H0» (60 изображений). Обе папки (исходные и обработанные изображения), а также код программы можно найти в репозитории.

*Перспективные инструменты.* В случае востребованности данной программы возможен такой вариант улучшения программы как экспериментальное определение оптимального порогового значения суммы пикселей по окну. Для этого можно применять размеченные изображения и специальные метрики машинного обучения.

Кроме того, необходимо будет провести оптимизацию кода программы для увеличения ее производительности (например, применять np.sum вместо свертки, если веса окна одинаковые, или переписать код с использованием многопоточности).

**2. Поиск соответствия 10-ти классам (mnist)**

Одна из типовых задач классификации – определить, какая из рукописных цифр отображена на черно-белом изображении 28\*28 пикселей.

*Что использовано.* Ранее разрабатывался простейший фреймворк для работы с нейронными сетями на языке MATLAB. А именно – с MLP и сверточными нейронными сетями (CNN, СНС). Собственно, модели СНС и применялись мной ранее для классификации цифр из набора MNIST.

Архитектура сети самая простая: по одной карте признаков в каждом слое свертки. Даже на таких простых архитектуре СНС и алгоритмах обучения сети, инициализации весовых коэффициентов наибольшее значение F-меры достигло 70%. Более подробно результаты изложены в статье собственного авторства (pdf-файл в папке 2 задания).

*Перспективные инструменты.* Работа выполнялась в конце 2018 года – когда мои навыки программирования на Python были еще низкими, а MATLAB уже тогда знал очень хорошо. Тогда еще не было соответствующего Toolbox’а в MATLAB (он вышел уже спустя пару месяцев под названием Deep Learning Toolbox, с оптимизированными и усовершенствованными алгоритмами).

И даже после этого, когда уже хорошо освоил Python и ML-библиотеки Keras, PyTorch, Scikit-Learn, результаты обучения даже базовых CNN-моделей на том же самом датасете MNIST уже превышали 99% (точность). А следовательно, если есть готовые фреймворки, то зачем оптимизировать свой? Тем более, что работа была скорее теоретическая и была направлена на более глубокое погружение в архитектуры сверточных сетей.

Однако если бы встал вопрос об улучшении фреймворка, то стоило бы улучшить, в первую очередь, алгоритм обучения СНС (добавить поддержку нескольких карт признаков в слое свертки, переменную скорость обучения, усовершенствованные методы генерации весовых коэффициентов), а также добавить такие features как пакетная нормализация, dropout, предобработка входа и т.д.