Тезисы.

1. Какие компоненты содержатся в составе библиотеки
2. Как скачать и установить
3. Как подготовить модель к загрузке в библиотеку, Model Optimizer, скрипт для конвертации
4. Тензоры как входы и выходы нейронной сети
5. Тензор изображения, 1- и 3- канальные изображения
6. Пример приложения

OpenVINO является библиотекой для работы с нейронными сетями. Она не позволяет обучать сети, только пользоваться уже готовыми моделями. Библиотека заточена на увеличение производительности вычислений с нейронной сетью. В её составе, к примеру, поставляется дистрибутив OpenCV, собранный со специфичными флагами, которые обеспечивают высокую производительность.

Библиотека позволяет запускать нейронную сеть на CPU, GPU, VPU, FPGA и прочих различных устройствах, что важно – для этого применяется универсальный интерфейс, таким образом, можно минимальными изменениями в готовой программе переключить её работу с CPU на GPU (например).

Для работы библиотеки нужна модель в специальном формате, который называется IR. Для приведения модели в этот формат есть специальные скрипты на питоне. Этот скрипт называется Model Optimizer. Он не только конвертирует модель, но также оптимизирует её для достижения большей производительности.

Model Optimizer умеет конвертировать в IR формат из известных форматов нейронных сетей, например Tensorflow, Caffe, Kaldi и прочих.

В нашей ситуации есть модель фреймворка Keras (.h5 + .json). Для конвертации такой модели в IR формат нужно сначала перевести её в формат Tensorflow (.pb), затем применить Model Optimizer.

Модель Tensorflow не совсем легко сконвертировать, у неё есть frozen параметры и unfrozen, прежде, чем сохранять её в .pb формат для дальнейшей конвертации в IR формат необходимо модель заморозить. Есть готовый скрипт, который это делает (это уже наш скрипт, самописный).

Итак, у нас есть модель в IR формате, нужно написать приложение, которое применит эту сеть к изображению.

Код приложения в проекте MSVS.

Замеряем время, на 103 картинки затрачено менее 2 секунд, с учётом загрузки изображений и сохранения на диск. Чисто на распознавание ушло около 700 миллисекунд, чуть более 6 миллисекунд на картинку.

Инструкция по конвертации.

1. В скрипте устанавливаем название файлов сети вашей сеткой: name = 'thin\_lines\_dataset\_alpha1.0\_biline\_weights'
2. Выполняем скрипт Keras to Tensorflow (если в вашей сети используется Relu6 – раскомментируйте соответствующую строку with)
3. Получившийся .pb файл перенесите в папку model\_optimizer
4. В терминале в папке model\_optimizer выполните команду: mo\_tf.py --input\_model thin\_lines\_dataset\_alpha1.0\_biline\_weights.pb --batch 1 , указав правильное название своего файла. Убедитесь в том, что терминал имеет права записи в папку model\_optimizer, возможно его потребуется запустить с правами администратора