**Пояснительная записка к фреймворку OpenVINO**

Intel® Distribution of **OpenVINO ™** toolkit помогает быстро развертывать решения для эмуляции человеческого зрения. Основанный на сверточных сетях (CNN), данный инструмент позволяет запускать алгоритмы компьютерного зрения (CV) на аппаратном обеспечении Intel®, с максимальной производительностью. OpenVINO включает в себя Intel® Deep Learning Deployment Toolkit (Intel® DLDT).

***Плюсы:***

**+** Позволяет запускать натренированные сверточные нейронные сети эффективно прямо на целевом устройстве.

**+** Ускоряет конечную разработку благодаря простоте эксплуатации библиотеки компьютерного зрения и предварительно оптимизированным ядрам.

**+** Поддерживает гетерогенное выполнение на процессорах Intel®, интегрированных графических процессорах Intel®, Intel® Movidius ™ Neural Compute Stick (NCS), Intel® Neural Compute Stick 2 и Intel® Vision Accelerator Design с Intel® Movidius ™ VPU.

**+** Включает оптимизированные вызовы для стандартов компьютерного зрения, включая OpenCV \*, OpenCL ™ и OpenVX.

**+** Продукт сам по себе не тяжелый и имеет практически нулевое количество зависимостей.

**+** Производительность продукта при вычислении сетей на платформах Intel в разы выше по сравнению с популярными фреймворками.

**+** Значительно ниже требования по используемой памяти, что актуально для ряда приложений.

**+** Поддерживает запуск сетей популярных фреймворков на всех платформах Intel и под управлением различных операционных систем, например, полноценный запуск на сетей на платформах Интел под управлением Windows OS. Запуск под управлением Windows – отдельная история. Не все фреймворки поддерживают такой запуск «из коробки», к примеру, запуск caffe не очень прост. TensorFlow поставляет бинарники для Windows, но, если нужно внести модификации и перестроить – может быть проблема. В то же самое время, заметно, что запуск на Windows часто востребован.

**+** Поставляются натренированные оптимизированные модели.

**+** Поставляется сразу несколько натренированных моделей для решения одной задачи (например, детектирования лиц). Некоторые модели могут работать значительно быстрее, за счёт незначительной потери в качестве. В зависимости от требований приложения можно выбрать то или другое.  
Также модели могут отличаться по условиям постановки задачи: к примеру, угол съемки человека может влиять на качество детектирования. OpenVINO предоставляет 2 модели для разных случаев.  
Следует отметить что для моделей существует файл с описанием, там можно посмотреть цифры производительности, аккуратность модели и примеры изображений, которые ожидаются на вход, т.е. описание сценария.

***Минусы:***

**-** Отсутствие тренировки сетей.

**-** Натренированные модели решают узкую задачу (к примеру детектирование пешеходов). Это делается из-за того, что узкая направленность позволяет значительно уменьшить размер модели. В случае публичных моделей делается попытка решить более общую задачу (детектирование нескольких классов объектов) и это требует гораздо более вычислительно сложных моделей с большим количеством параметров. В приведённом выше примере (детектировании пешеходов) OpenVINO модель может решать задачу в 10+ раз быстрее, чем публичная с не худшим качеством.

**-** Интеграция с Web.

**Установка**

Следующие компоненты устанавливаются по умолчанию:

| **Компонент** | **Описание** |
| --- | --- |
| [Model Optimizer](https://software.intel.com/en-us/articles/OpenVINO-ModelOptimizer) | Этот инструмент импортирует, конвертирует и оптимизирует модели, которые были обучены на популярных платформах, в формат, используемый инструментами Intel, особенно Inference Engine.  Примечание. Популярные платформы включают Caffe \*, TensorFlow \*, MXNet \* и ONNX \*. |
| [Inference Engine](https://software.intel.com/en-us/articles/OpenVINO-InferEngine) | Это двигатель, который запускает модель глубокого обучения. Он включает в себя набор библиотек для простой интеграции в ваши приложения. |
| Drivers and runtimes for OpenCL™ version 2.1 | Включает OpenCL на GPU / CPU для Intel® процессоров. |
| Intel® Media SDK | Предоставляет доступ к аппаратно ускоренным видеокодекам и обработке кадров. |
| [OpenCV\*](https://docs.opencv.org/) | Community версия OpenCV \*, скомпилированная для оборудования Intel®. Включает библиотеки PVL для компьютерного зрения. |
| OpenVX\* | Intel реализация OpenVX \* оптимизированная для работы на оборудовании Intel® (CPU, GPU, IPU). |
| Pre-trained models | Набор предварительно обученных моделей Intel для учебных и демонстрационных целей или для разработки ПО для глубокого обучения. |
| Sample Applications | Набор простых консольных приложений, демонстрирующих, как использовать Inference Engine в ваших приложениях. Для получения дополнительной информации о построении и запуске примеров, обратитесь к [Руководство к разработке на Inference Engine](https://software.intel.com/en-us/articles/OpenVINO-InferEngine#Using_Inference_Engine_Samples). |

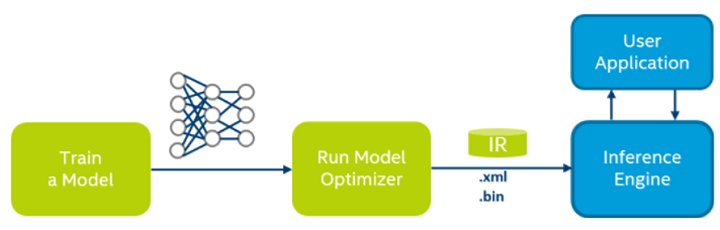
[Установка OpenVINO на Linux.](https://software.intel.com/en-us/articles/OpenVINO-Install-Linux)

[Установка OpenVINO на Windows.](https://software.intel.com/en-us/articles/OpenVINO-Install-Windows)

**Inference Engine**

**Рабочий процесс развертывания**

Процесс развертывания Inference Engine предполагает, что вы использовали Model Optimizer, чтобы преобразовать вашу обученную модель в промежуточное представление(IR - Intermediate Representation). Схема ниже иллюстрирует типичный рабочий процесс для развертывания натренированной модели глубокого обучения.



Краткое изложение шагов по оптимизации и развертыванию обученной модели:

* Настройте Model Optimizer для вашей платформы.
* Преобразуйте обученную модель для получения оптимизированного промежуточного представления (IR) модели на основе топологии обученной сети, значений весов и смещений.
* Протестируйте модель в формате промежуточного представления используя Inference Engine в целевой среде с помощью приложения проверки или приложений-примеров.
* Интегрируйте Inference Engine в свое приложение, чтобы развернуть модель в целевой среде.

**Вводная в Inference Engine**

Inference Engine – это C++ библиотека с набором C++ классов чтобы запусить сети на входных данных (изображениях) и получить результат. C++ библиотека предоставляет некий API чтобы считывать промежуточное представление (IR), устанавливать входящие и выходящие форматы и запускать модели на устройствах.

Inference Engine использует архитектуру плагинов. Плагин Inference Engine - это программный компонент, который содержит полную реализацию для инференса на определенном аппаратном устройстве Intel®: например, CPU, GPU, VPU, FPGA. Каждый плагин реализует унифицированный API и предоставляет дополнительные API-интерфейсы, специализированные к конкретному аппаратному устройству.

**Общий рабочий процесс c Inference Engine**

1. **Прочитайте IR** - Используя класс InferenceEngine::CNNNetReader, прочтите файл IR в класс CNNNetwork. Этот класс представляет сеть в памяти хоста.
2. **Подготовьте формат входов и выходов**. После загрузки сети укажите точность и схему входа и выхода сети. Для этих спецификаций используйте CNNNetwork::getInputInfo() и CNNNetwork::getOutputInfo().
3. **Выберите плагин** - выберите плагин для загрузки вашей сети. Создайте плагин с помощью вспомогательного класса загрузки InferenceEngine::PluginDispatcher. Передайте по-девайсные конфигурации загрузки, специфичные для данного устройства, и зарегистрируйте расширения для данного устройства.
4. **Постройте и загрузите** - используйте класс-оболочку над интерфейсом плагина InferenceEngine::InferencePlugin для вызова LoadNetwork(), чтобы скомпилировать и загрузить сеть на устройстве. Передайте по-целевую конфигурацию загрузки для данной операции построения и загрузки.
5. **Задать входные данные** – После загрузки сети у вас появляется объект класса ExecutableNetwork. Используйте этот объект для создания InferRequest, в котором вы указываете входные буферы для в и вывода. Укажите выделенную устройству память и скопируйте ее непосредственно в память устройства или попросите устройство использовать память приложения для сохранения копии.
6. **Выполнить** - Теперь, когда память ввода и вывода определена, выберите режим выполнения:

* Синхронно - метод Infer(). Блокирует до завершения инференса.
* Асинхронно - метод StartAsync(). Cтатус проверяется с помощью метода wait() (0 timeout), ждите или укажите обратный вызов для завершения.

1. **Забрать результат** - после завершения инференса прочитайте выходную память или память, предоставленную ранее. Сделайте это с помощью InferRequest GetBlob API.

[Интеграция IE API в приложение наглядно.](https://software.intel.com/en-us/articles/OpenVINO-InferEngine#integrate-with-api)