OKI AE2100 OpenVINO™ Training

21-Jul. '2021

Yasunori Shimura

intel®

Notices & Disclaimers

- Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex
- Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.
- © Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
- The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that code included in this document is licensed subject to the Zero-Clause BSD open source license (0BSD), https://opensource.org/licenses/0BSD.

Agenda

1. OpenVINO™おさらい

2. AE2100を使ったOpenVINO™開発手順ご紹介

1. OpenVINO™デモプログラムビルドからAE2100での実行までをご紹介
2. インテル提供の学習済みモデルの入手法などもご紹介

3. OpenVINO™プログラムの基本、非同期推論の基本

1. Pythonを使ったOpenVINO™プログラムの基本構造を解説
2. 非同期推論を活用し、パフォーマンスを向上させる方法についても解説

4. OpenVINO™活用Tips

1. OpenVINO™の付属ツール、コンパニオンツール類のご紹介
2. OpenVINO™活用、プログラミング上のポイント、ベンチマークのとり方、知っておいてほしいことなどご紹介

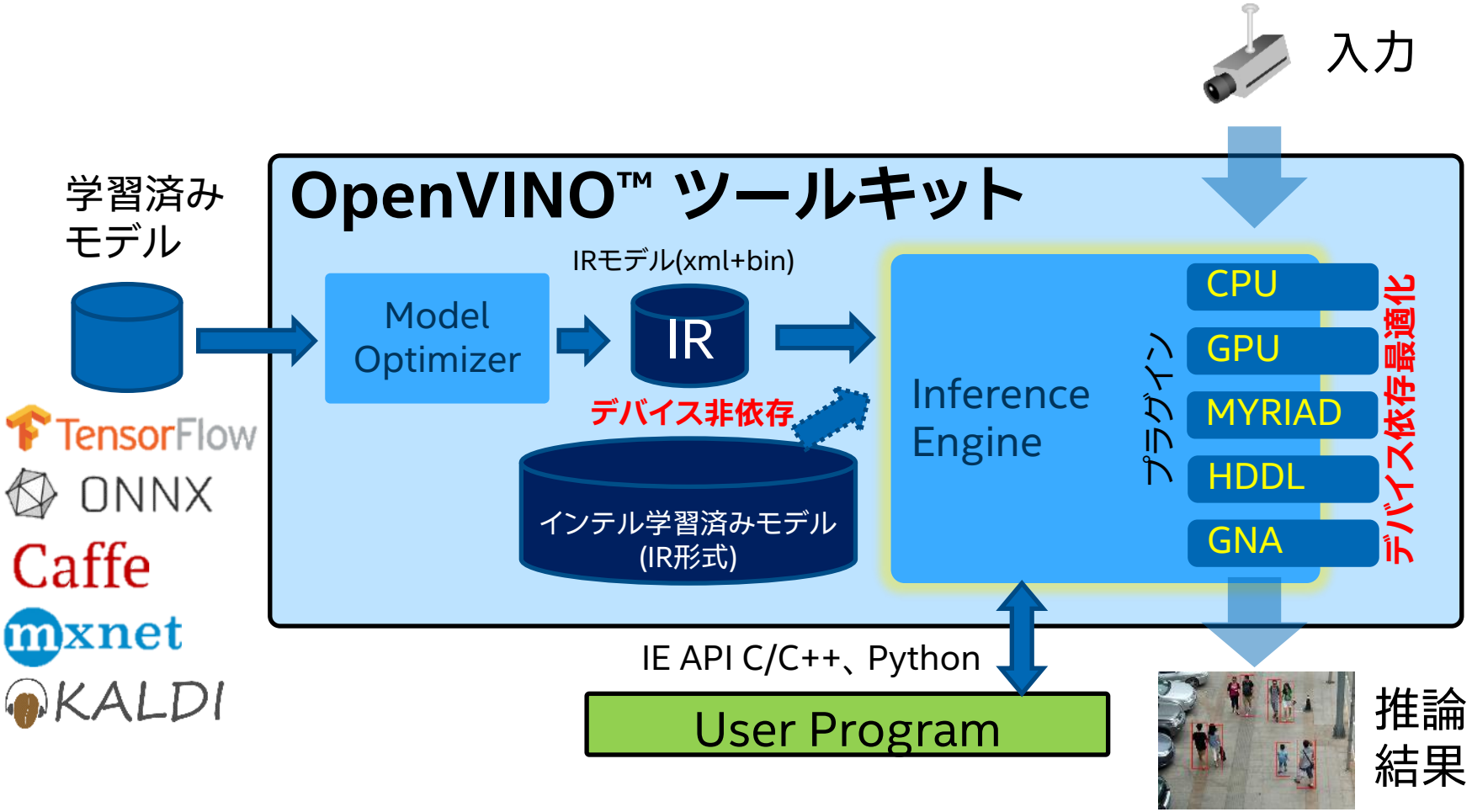
1. OpenVINO™ ツールキット おさらい

OpenVINO™ ツールキットとは?

- OpenCV* + DL推論エンジン + サポートツール
- オープンソース
- CPU/GPUの最新機能、命令にいち早く対応
- CPU、GPU、VPU、GNA をサポート
- 140以上のOpsに対応。カスタムOp作成可能
- 200を超える学習済みモデルを利用可能
- C/C++/Python*でプログラム可能
- INT8化、INT8推論にも対応
- TF、ONNX*、Caffe、MxNet*、Kaldiモデルを変換可能
- GStreamerで利用可能なDLエレメント同梱 (DL streamer)
- 年4回アップデート、LTSあり、無償!

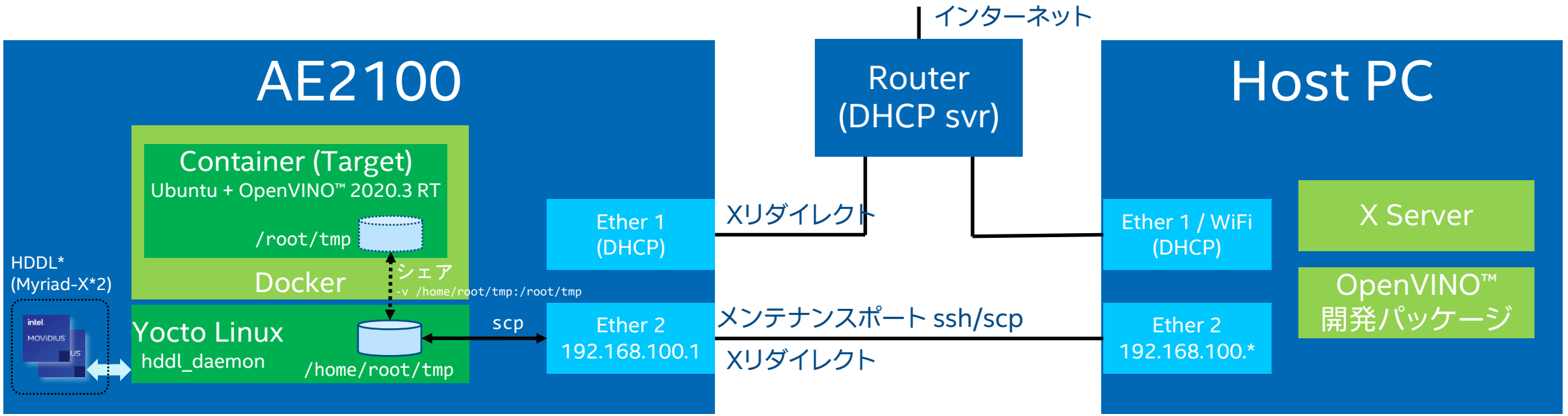
https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html

OpenVINO™ ツールキットの仕組み



2. AE2100を使ったOpenVINO™開発手順ご紹介

今回のトレーニングでの想定構成



- 内蔵GPU、VPU (MYRIAD)がコンテナ内から使うためには、コンテナ起動時に適切なオプションを渡す必要がある
- この環境ではHDDL daemonを通してVPUを使用しているため、Yocto上でHDDL daemonを起動した後にContainerをスタートする必要がある

- AE2100側はAtomプロセッサでAVX命令をサポートしていない。ビルド時はAVXを無効にし、SSEのみ有効になるように注意 (-msse4.2, -DENABLE_AVX2=OFFなど)
- X Serverは通常外部からのコネクションをブロックするようになっている。xhostコマンドなどで受け入れるようにする ('xhost +')

HDDL: High Density Deep Learning accelerator

AE2100上でOpenVINO™を使う場合のポイント

- HDDL daemonの初期設定と起動
  - AE2100 SDK取扱説明書 DL編「1.4 初期設定」
- Dockerコンテナの作成と起動 - 内蔵GPUとHDDL(MYRIAD-X, VPU)を使えるようにオプション指定
  - AE2100 SDK取扱説明書 DL編 「2.4 AE2100へのデプロイ, 3, 4, 5」

```
docker run \
 --device /dev/dri \                    # iGPUに必要
 -it \
 --device=/dev/ion:/dev/ion \           # VPU (HDDL)に必要
 -v /var/tmp:/var/tmp \
 -v /home/root/tmp:/root/tmp \          # ホスト-コンテナ間ファイル共有に便利(optional)
 --name ubuntu-openvino \               # コンテナ名
 -d ubuntu:openvino_2020R3 \
 /bin/bash
```

- 停止したDockerコンテナの再起動と接続

```
docker start ubuntu-openvino
docker attach ubuntu-openvino
```

- OpenVINO™の環境設定 (パス、環境変数など)
  - AE2100 SDK取扱説明書 DL編 「2.4 AE2100へのデプロイ, 10」

```
source /opt/intel/openvino/bin/setupvars.sh
```

ホストPCとAE2100間の接続、データ転送

- 普通にsshでログイン (メンテナンスポート, LAN #2)

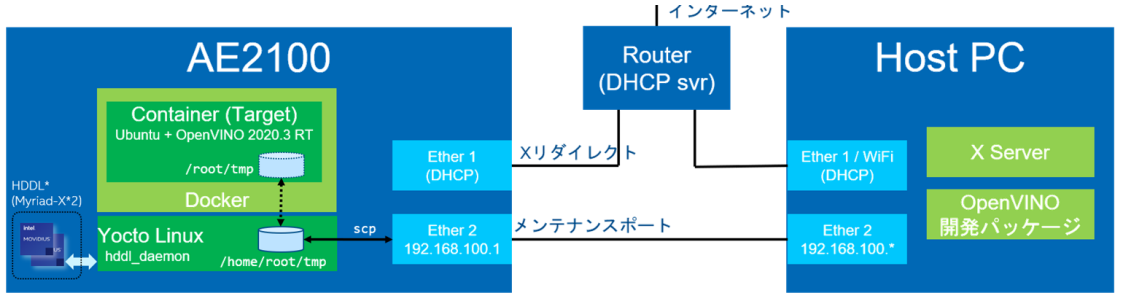
```
# ssh root@192.168.100.1
```

- 普通にscp / rsyncなどで転送

```
# scp aaa.txt root@192.168.100.1:/home/root     # HostPC -> AE2100
# scp root@192.168.100.1:/home/root/bbb.sh .    # AE2100 -> HostPC
```

開発環境のセットアップ

H: HostPC, Y: AE2100 Yocto, U: Ubuntu (AE2100, Ubuntu/OpenVINO container)



- 「AE2100シリーズ SDK取扱説明書 Deep Learning編」に従って環境設定を行う
- **Host PC (開発, Ubuntu)環境**
  - Ubuntu 18.04
  - OpenVINO™ 2020.3 開発パッケージ
- **AE2100 Yocto環境**
  - Target container image (Ubuntu+OpenVINO™)の読み込み。イメージはOkiさんが配布。
  - HDDL daemonのセットアップ (自動起動設定)
- **Ubuntu container (ターゲット)環境**
  - ライブラリのインストール
    - U: apt update
    - U: apt install libgtk-3-0 git

libgtk-3-0はXをリダイレクトしてGUI表示する場合に必要になるはず

AE2100でhuman_pose_estimation_demoを動かしてみよう

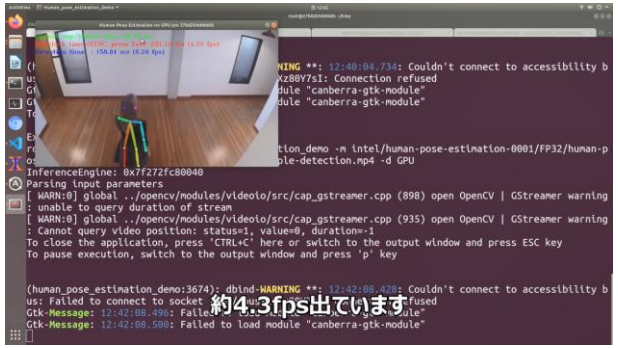
- (H, Y, U) OpenVINO™環境変数設定
  - **かならずsourceまたは'.' (ドット)コマンドで実行。**直接スクリプトを実行すると子シェルを開き、その中で変数設定し、子シェルを破棄して終了するので意味がない
  - H, Y, U: **source** /opt/intel/openvino_2020.3.194/bin/**setupvars.sh**
- (H) X serverのクライアントブロックを解除
  - H: xhost +
- (H) OpenVINO™サンプルプログラムビルドしてAE2100に転送
  - H: cd $INTEL_OPENVINO_DIR/deployment_tools/open_model_zoo/demos
  - deployment_tools/demos/build_demo.sh を修正して**AVX命令を生成しないようにする** (AE2100はAtomでAVX命令をサポートしないため)
  - H: ./build_demos.sh
    H: cd ~/omz_demos_build/intel64/Release
    H: scp human_pose_estimation_demo root@192.168.100.1:/home/root/tmp

```
** open_model_zoo/demos/build_demo.sh

mkdir -p "$build_dir"

(cd "$build_dir" && cmake -DCMAKE_BUILD_TYPE=Release
"${extra_cmake_opts[@]} -DENABLE_AVX2=OFF" "$DEMOS_PATH")
cmake --build "$build_dir" -- "$NUM_THREADS"
```

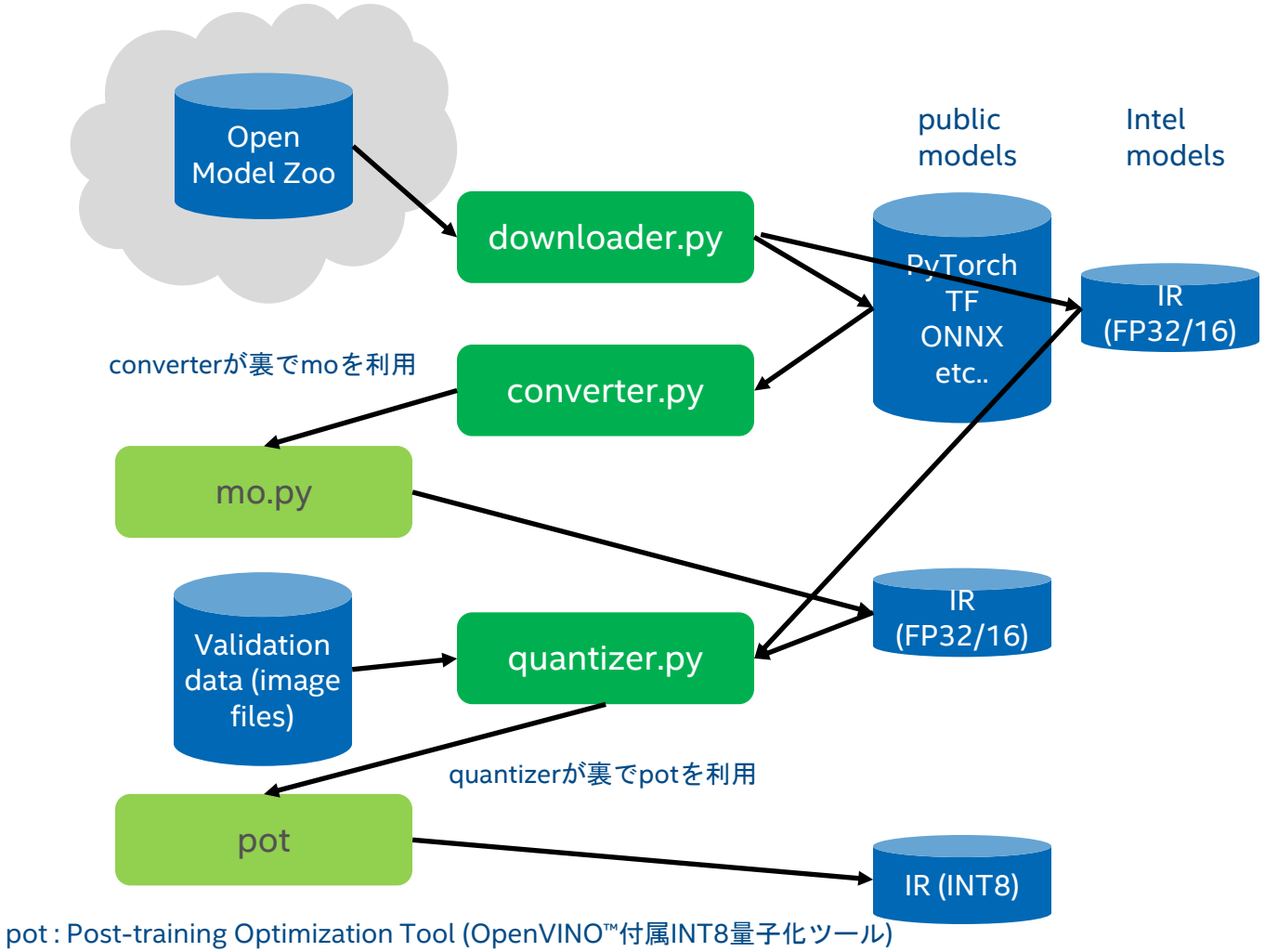
- (H) 必要なDLモデル、入力ムービーをダウンロードしAE2100に転送
  - H: python3 $INTEL_OPENVINO_DIR/deployment_tools/open_model_zoo/downloader/downloader.py --name "human-pose-estimation*"
    H: git clone https://github.com/intel-iot-devkit/sample-videos
    H: scp -r intel root:192.168.100.1:/home/root/tmp
    H: scp -r sample-videos root:192.168.100.1:/home/root/tmp
- (U) デモを実行 (Target)
  - U: cd ~/tmp
    U: ./human_pose_estimation_demo -m intel/human-pose-estimation-0001/FP16/human-pose-estimation-0001.xml -d CPU

human_pose_estimation_demo DEMO

CPU / GPU / HDDL



Open Model Zooモデルの入手



- 下記のようにモデル名指定するだけでOMZモデルをダウンロード、変換することが可能
  python3 downloader.py --name googlenet-v1
  python3 converter.py --name googlenet-v1
- converterを使うためにはModel Optimizerのセットアップがちゃんとできている必要あります
- 通常配布されているTensorFlowはAVX命令を利用するため、Atom環境では動作しない。**これらのツールは開発環境で利用する**
- 入手可能なモデル一覧を表示するには'--print_all'オプション
- OMZモデルはGitHubレポから入手することも可能
  https://github.com/openvinotoolkit/open_model_zoo
- OMZの各モデルのページにはREADME.mdがあり、**モデルの入出力シェイプ、モデルのサイズ、精度などの情報を入手可能**

OMZモデル情報の例

ines (37 sloc) | 2.25 KB Raw

face-detection-0200

Use Case and High-Level Description

Face detector based on MobileNetV2 as a backbone with a multiple SSD head for indoor and outdoor scenes shot by a front-facing cam
During the training of this model, training images were resized to 256x256.

Example

Specification

Metric	Value
AP (WIDER)	86.74%
GFlops	0.786
MParams	1.828
Source framework	PyTorch*

Average Precision (AP) is defined as an area under the precision/recall curve. All numbers were evaluated by taking into account only faces bigger than 64 x 64 pixels.

Inputs

Image, name: `input`, shape: `1, 3, 256, 256` in the format `B, C, H, W`, where:

- `B` - batch size
- `C` - number of channels
- `H` - image height
- `W` - image width

Expected color order: `BGR`.

Outputs

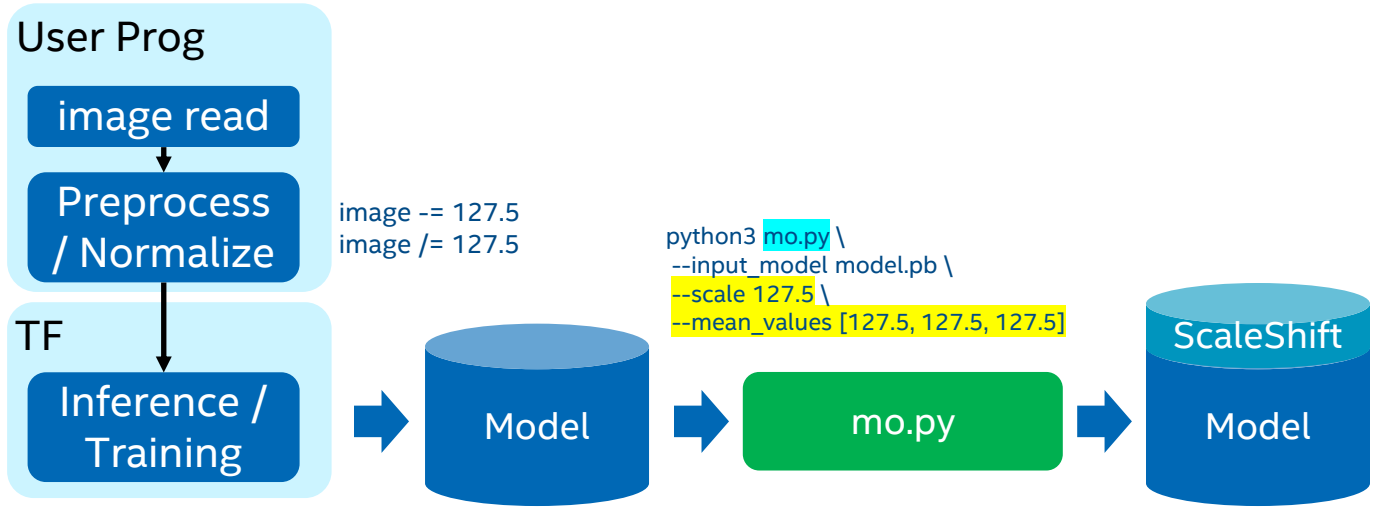
The net outputs blob with shape: `1, 1, 200, 7` in the format `1, 1, N, 7`, where `N` is the number of detected bounding boxes. Each detection has the format [`image_id`, `label`, `conf`, `x_min`, `y_min`, `x_max`, `y_max`], where:

- `image_id` - ID of the image in the batch
- `label` - predicted class ID (0 - face)
- `conf` - confidence for the predicted class
- (`x_min`, `y_min`) - coordinates of the top left bounding box corner
- (`x_max`, `y_max`) - coordinates of the bottom right bounding box corner

Training Pipeline

Model Optimizer (MO)

- MOはTensorflow, Caffe, Mxnet, ONNX, Kaldiの学習済みモデルをOpenVINO™のIR (Intermediate Representation)に変換するツール
- PyTorchなどは一度ONNXに変換してからIRに変換 (Model Downloaderのディレクトリにpytorch_to_onnx.pyというツールがあります)
- MOでは入力データ正規化のパラメータ指定して変換をするとIRモデルの先頭に正規化レイヤ (ScaleShift)を挿入します。この場合、ユーザープログラムでも正規化を行うと、処理が2重になり正しい結果が得られないので注意してください。



- MOの詳細な使い方などはデベロッパーガイドを参照ください。フレームワークごとの説明やFAQも参考になります
https://docs.openvinotoolkit.org/latest/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html

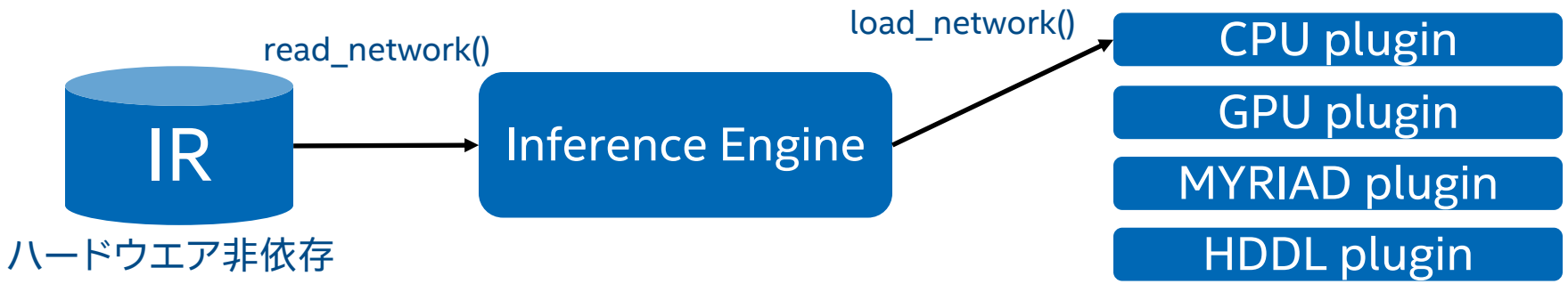
MOでうまく変換できない場合。。

- FAQを調べてみてください
  - https://docs.openvinotoolkit.org/latest/openvino_docs_MO_DG_prepare_model_Model_Optimizer_FAQ.html
- TFのモデルで入力シェイプが確定していない(シェイプに-1を含んでいる場合 ([-1, 224,224,3]など))
  - 入力シェイプを指定すればOKです。MOの--input_shapeオプションや--batchオプションで数値を指定してください
- モデルがモデル以外の処理や推論に不要な処理を含んでいる
  - JPEGデコードレイヤーがモデルの先頭にある場合、モデルの最後に精度チェックやテスト用のロジックが残っている場合など。。
  - このような場合は、MOの--input, --outputオプションでモデルの変換範囲を指定することで解決できる場合があります(例えば、JPEGレイヤの次のレイヤを--inputで指定することでJPEGレイヤを変換対象外とする)
- モデルがカスタムレイヤー/Opを含んでいる
  - OpenVINO™もカスタムレイヤーを追加する事が可能です
  - CPU, GPU, VPU (Myriad-X)向けのカスタムカーネルを開発可能
  - 詳細は下記webページを参照ください
    https://docs.openvinotoolkit.org/latest/openvino_docs_HOWTO_Custom_Layers_Guide.html

3. OpenVINO™でのプログラムの基本、非同期推論の基本

OpenVINO™でのプログラムの骨組みを理解する (Python)

```python
import numpy as np
from openvino.inference_engine import IECore

ie = IECore()                                                    # Instantiate IE core object
model='./public/googlenet-v1/FP16/googlenet-v1'
net = ie.read_network(model+'.xml', model+'.bin')                # Read model file
exenet = ie.load_network(network=net, device_name='CPU') # Load the model to device plugin
dummy_in = np.zeros((1,3,224,224), dtype=np.uint8)               # Prepare dummy input data
res = exenet.infer({'data':dummy_in})                            # Infer ('data' is input blob name)
print(res['prob'].shape)                                         # Inference result is in 'res['prob']'
```



イメージ分類プログラムの例 (CNN, Image Classification)

```
import cv2
import numpy as np
from openvino.inference_engine import IENetwork, IECore

ie = IECore()
model = './public/googlenet-v1/FP16/googlenet-v1'
net = ie.read_network(model+'.xml', model+'.bin')              # ファイルからモデルを読み込み
inputBlobName  = next(iter(net.inputs))                        # Input blobの名前を取得
outputBlobName = next(iter(net.outputs))                       # Output blobの名前を取得
batch,channel,height,width = net.inputs[inputBlobName].shape  # Input blobのshapeを取得
exec_net = ie.load_network(network=net, device_name='CPU')     # モデルをpluginに読み込み

img = cv2.imread('./car.png')                                  # ファイルから入力画像を読み込み
cv2.imshow('input', img)                                       # 読み込んだ画像を表示
img = cv2.resize(img, (width,height))                          # 画像を入力shapeに合わせてリサイズ
#img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)                    # 色チャネル入れ替え(BGR->RGB)
img = img.transpose((2, 0, 1))                                 # ランクの入れ替え HWC -> CHW
img = img.reshape((1, channel, height, width))                 # バッチランクを追加 CHW -> NCHW

res = exec_net.infer(inputs={inputBlobName: img})              # 推論実行

res = res[outputBlobName][0]                                   # 推論結果取り出し
idx = np.argsort(res)[::-1]
for i in range(5):
    print(idx[i]+1, res[idx[i]])
cv2.waitKey(3*1000)
```

初期化
だいたい同じ

前処理
モデルによって変わるが画像推論モデルならだいたい似たようなもの

後処理
モデルによって変わる

C++プログラムの構造とビルド方法について

- 開発PC上に下記3ファイルを作成し、build.shを実行
- ./build/simple_cnnが生成されるので、AE2100 Ubuntu/OpenVINO™環境に転送して実行
  - scp simple_cnn root@192.168.100.1:/home/root/tmp

CMakeLists.txt

```
# Sample CMakeLists.txt file for an OpenVINO Inference Engine project
cmake_minimum_required (VERSION 2.8.1)

set(TARGET_NAME simple_cnn)     # name of executable file
set(CMAKE_BUILD_TYPE "Release")

set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -msse4.2 -fPIE")

find_package(InferenceEngine 1.1 REQUIRED)
find_package(OpenCV REQUIRED)
add_definitions(-DUSE_OPENCV)

include_directories( ${InferenceEngine_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS} )
link_directories( )

add_executable( ${TARGET_NAME} main.cpp )    # list of source file(s)
set_target_properties(${TARGET_NAME} PROPERTIES "CMAKE_CXX_FLAGS" "${CMAKE_CXX_FLAGS}")
target_link_libraries(${TARGET_NAME} ${InferenceEngine_LIBRARIES} ${OpenCV_LIBS} )
```

build.sh

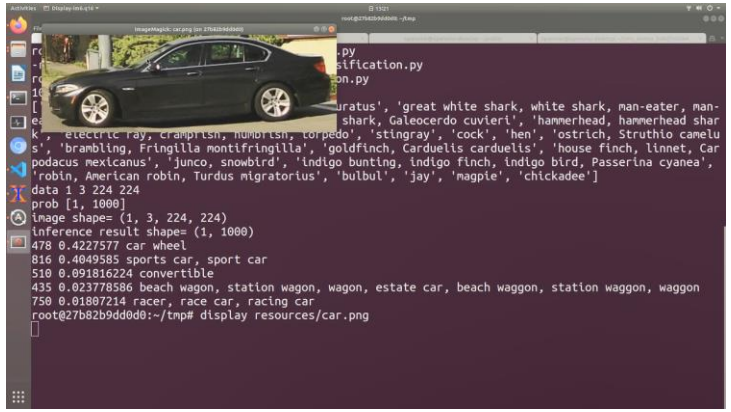
```
mkdir -p build && pushd ./build
cmake -DCMAKE_BUILD_TYPE=Release ..
make
popd
```

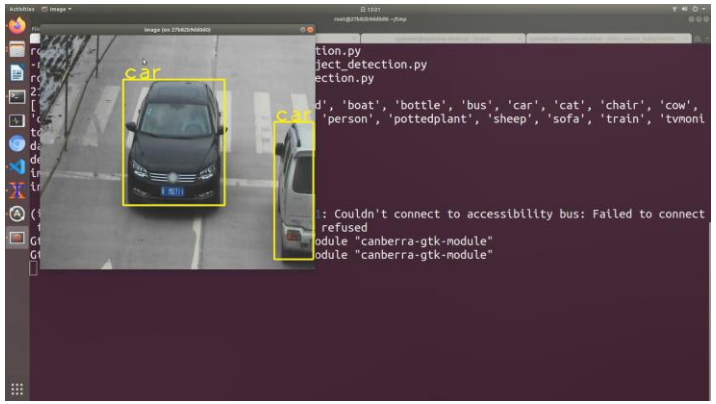
main.cpp

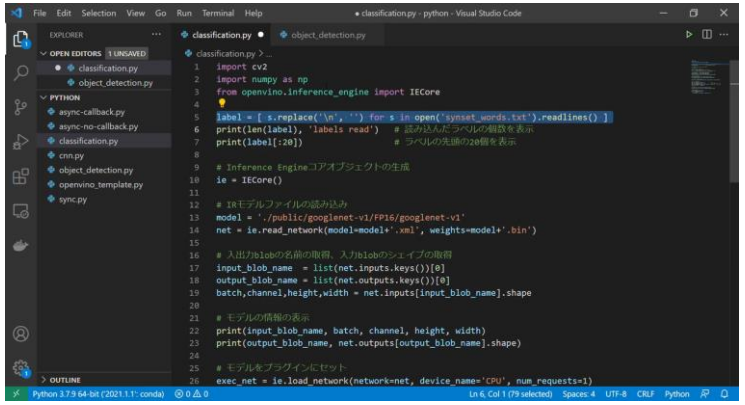
```
#include <vector>

#include <opencv2/opencv.hpp>
#include <inference_engine.hpp>

namespace ie = InferenceEngine;

int main(int argc, char *argv[]) {
    ie::Core ie;

    ie::CNNNetwork network;
    network = ie.ReadNetwork("./public/googlenet-v1/FP16/googlenet-v1.xml",
                             "./public/googlenet-v1/FP16/googlenet-v1.bin");
    std::shared_ptr<ie::InputInfo> input_info = network.getInputsInfo().begin()->second;
    std::string input_name = network.getInputsInfo().begin()->first;
    input_info->getPreProcess().setResizeAlgorithm(ie::RESIZE_BILINEAR);
    input_info->setLayout(ie::Layout::NHWC);
    input_info->setPrecision(ie::Precision::U8);

    ie::DataPtr output_info = network.getOutputsInfo().begin()->second;
    std::string output_name = network.getOutputsInfo().begin()->first;
    output_info->setPrecision(ie::Precision::FP32);

    ie::ExecutableNetwork executable_network = ie.LoadNetwork(network, "CPU");
    ie::InferRequest infer_request = executable_network.CreateInferRequest();

    cv::Mat image = cv::imread("./car.png");

    ie::TensorDesc tDesc(ie::Precision::U8,
            {1, 3, static_cast<long unsigned int>(image.rows),
                   static_cast<long unsigned int>(image.cols)}, ie::Layout::NHWC);
    infer_request.SetBlob(input_name, ie::make_shared_blob<uint8_t>(tDesc, image.data));

    infer_request.Infer();

    float* output = infer_request.GetBlob(output_name)->buffer();
    std::cout << "\nresults\n-----------------" << std::endl;
    std::vector<int> idx;
    for(int i=0; i<1000; i++) idx.push_back(i);
    std::sort(idx.begin(), idx.end(),
            [output](const int& left, const int& right) { return output[left]>output[right]; } );
    for (size_t id = 0; id < 5; ++id) {
        std::cout << id <<  " : " << idx[id] << " : " << output[idx[id]]*100 << "% " << std::endl;
    }
}
```

DEMO & CODE REVIEW

Classification.py, object_detection.py







OpenVINO™の高速化手法

OpenVINO™の高速化ストラテジー

ユーザープログラム自身の最適化	•Intel® vTune™ プロファイラー, Intel® アドバイザーなどのツール活用してプログラム中のホットスポットを重点的に高速化 •並列度向上もパフォーマンス改善には極めて重要
モデル軽量化	•モデル自身のアーキテクチャを最適化する (human-pose-estimationなど) •Open Model Zooの軽量モデルを検討する •SOTAモデルにこだわらず、目的精度を達成する軽量モデルの採用を検討する
モデル軽量化 (FP16化)	•Model Optimizerでモデル変換時に"--data_type FP16"オプションを指定してIRモデルを生成
モデル軽量化 (INT8化)	•OpenVINO™に付属しているPost-training Optimization Tool (POT)を利用してFP32/FP16のIRモデルをさらに量子化する •POTの変換設定ファイル(YAML)、バリデーションデータ(入力画像など)の準備が必要
量子化を考慮したトレーニング	•Quantize Aware Training (QAT)ツールであるNNCF (Neural network Compression Framework)を使用してモデルの学習を行う。学習時から重みデータの最適化、スパース化が可能になる。 •https://github.com/openvinotoolkit/nncf
非同期推論	•非同期推論APIを使用する(infer() -> async_start())ことで推論デバイスを飽和させ最大のパフォーマンスを引き出す
高速CPU	•CPUは推論処理だけではなく、プリ・ポストプロセスや、その他一般ワークロードまで高速化することが可能 •内蔵GPUも高速化することが多いので、トータルSoC処理能力が大幅に上がる •最新CPUではCPU, GPUともにINT8演算を高速化する新命令により性能が大幅に向上している
OpenVINO™ Model Server	•推論処理を推論サーバーに任せることでエッジデバイスだけでは達成できない高性能を実現 •推論サーバーに投資を集中することで効率良いシステムを構築

ソフトウエア、モデルの見直し、最適化でパフォーマンスが何倍にもなることは珍しくありません。これは数倍高速なCPUをタダで手に入れるのと(ほぼ)同義です

benchmark_appで同期推論と非同期推論のパフォーマンス差を確認する

- benchmark_appの準備
  - python3 -m pip install -r $INTEL_OPENVINO_DIR/deployment_tools/tools/benchmark_tool/requirements.txt
- sync(同期)とasync(非同期)推論モードでbenchmark_appを実行してみる
  - python3 $INTEL_OPENVINO_DIR/deployment_tools/tools/benchmark_tool/benchmark_app.py \
    -m public/googlenet-v1/FP16/googlenet-v1.xml \
    -niter 100 -api sync -d HDDL
    - => 40.9 fps
  - python3 $INTEL_OPENVINO_DIR/deployment_tools/tools/benchmark_tool/benchmark_app.py \
    -m public/googlenet-v1/FP16/googlenet-v1.xml \
    -niter 100 -api async -d HDDL
    - => 184.98 fps (x4.5)

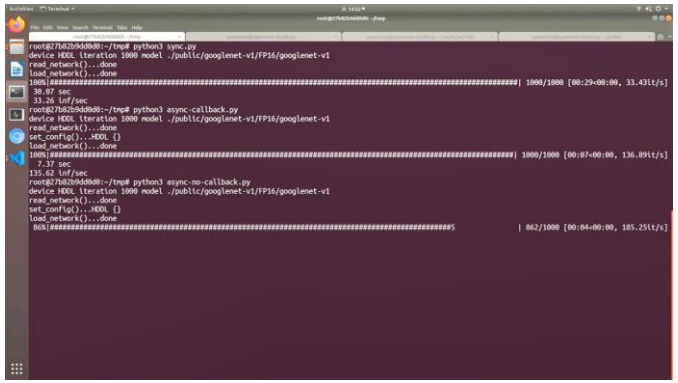
- benchmark_appは汎用ベンチマークツール。どんなモデルでもベンチマーク可能。様々なパラメータを変化させてパフォーマンスの差を確認できる。インテル推奨のパフォーマンス確認ツール
- デモプログラムはパフォーマンスチューニングされているわけではありません。デモやサンプルプログラムでのパフォーマンス計測は推奨しません。
- C++版のbenchmark_appとPython版のbenchmark_app.pyがありますが、パフォーマンス計測上の差はありません。

自作Pythonプログラムでのパフォーマンス差 (同期/非同期)

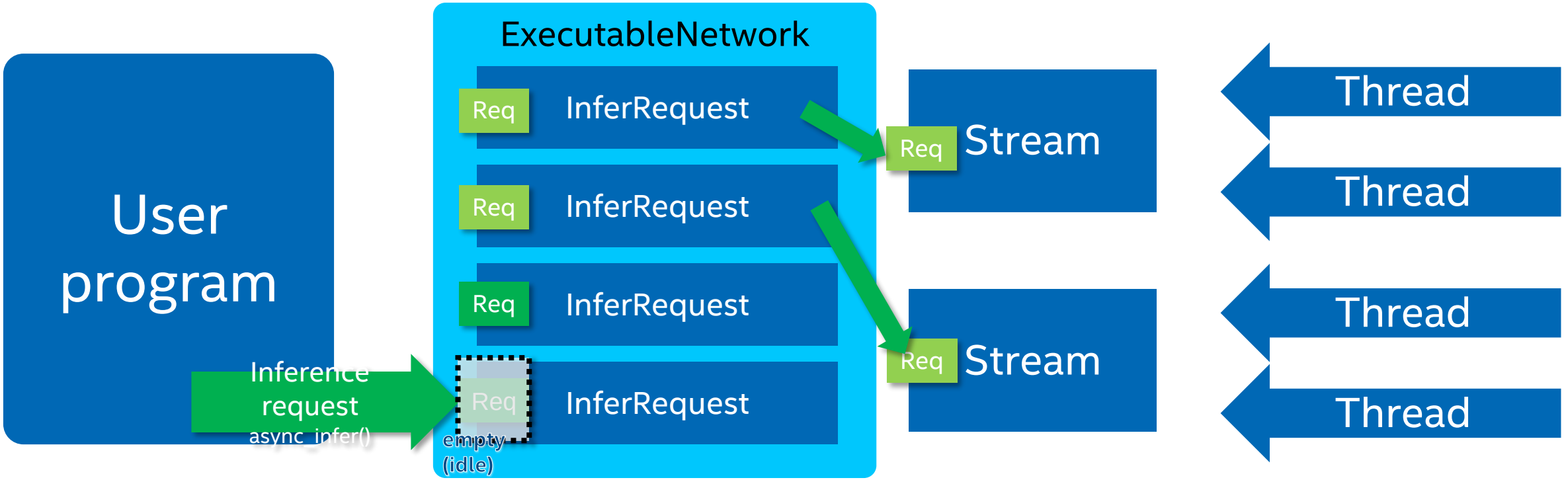
- AE2100, HDDL (Myriad-X x2)を使用した場合のパフォーマンスを計測
- カッコ内はpre/post process相当のダミーwaitの時間
  - sync.py (1ms) 33.66 fps
  - sync.py (3ms) 29.44 fps (-12%)
  - async-callback.py (1ms) 136.49 fps
  - async-callback.py (3ms) 127.67 fps (-6%)
  - async-no-callback.py(1ms) 177.77 fps
  - async-no-callback.py(3ms) 177.42 fps (-0.2%)
- 非同期推論でPythonでもパフォーマンスが伸びる
- callback関数を使わない方式のほうがlockなどの同期処理が不要でpost process分の時間を隠しやすいが、プログラムは複雑になる傾向がある

同期、非同期推論パフォーマンスデモ

sync.py, async-callback.py, async-no-callback.py



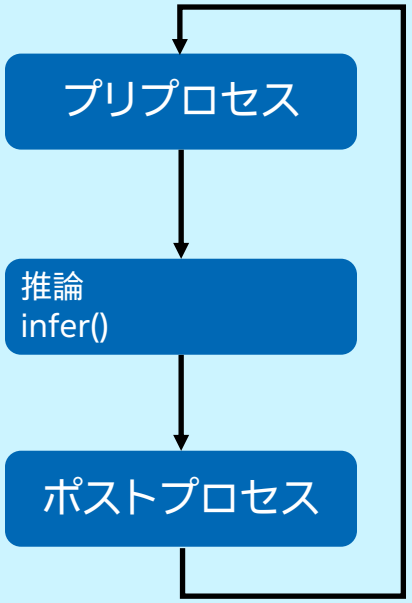
Streams? Threads? InferRequests?



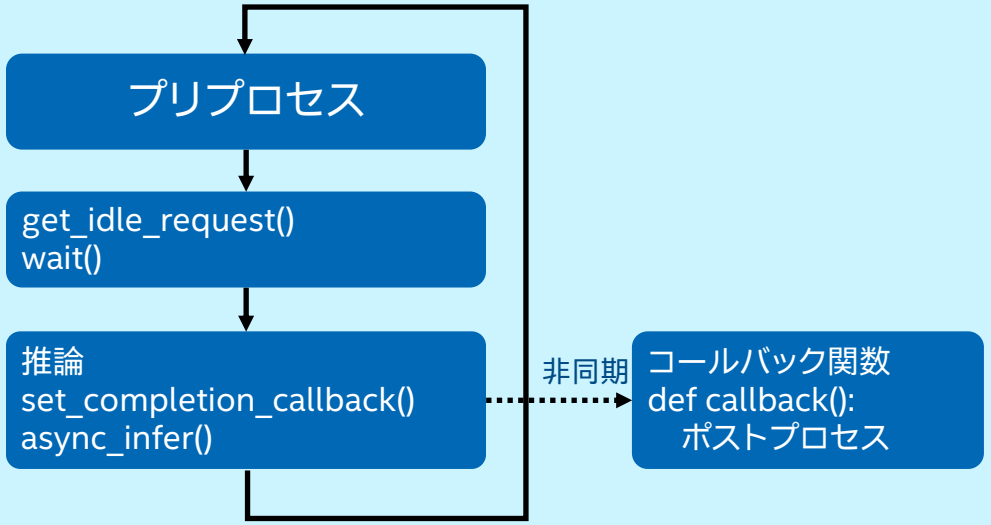
OpenVINO™ benchmark_app params	-nireq	-nstreams	-nthreads
OpenVINO™ API	ie.load_network(network, device_name, num_request=n)	ie.set_config(cfg, dev)	ie.set_config(cfg, dev)

同期推論、非同期推論プログラムの構造と特徴

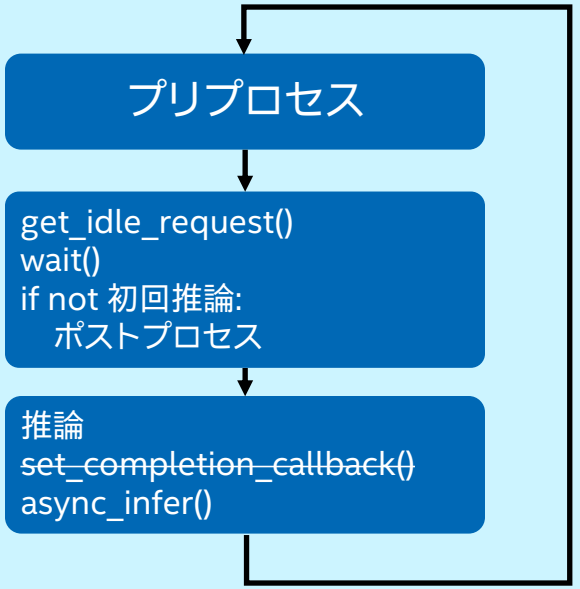
sync.py



async-callback.py



async-no-callback.py



同期推論

- プログラムが簡単
- 動作確認などには最適

非同期推論

- コールバックが完全に非同期で呼び出されるので、メインスレッドと共有するオブジェクト(executable_networkやinfer_requestなど)を操作する場合に同期処理(lock, mutexなど) が必要
- 同期処理さえなんとかすれば比較的書きやすく、高速

非同期推論 (コールバックなし)

- メインスレッドしか使わないので同期処理は不要
- メインスレッド内でのステート処理、分岐が複雑になりがち
- 自由度が低く難易度は高めだが、性能は出しやすい(同期処理がないため)

4. OpenVINO™の活用Tips

OpenVINO™の 付属ツール、外部add-onツールのご紹介

OpenVINO™ ツールキット 付属ツール

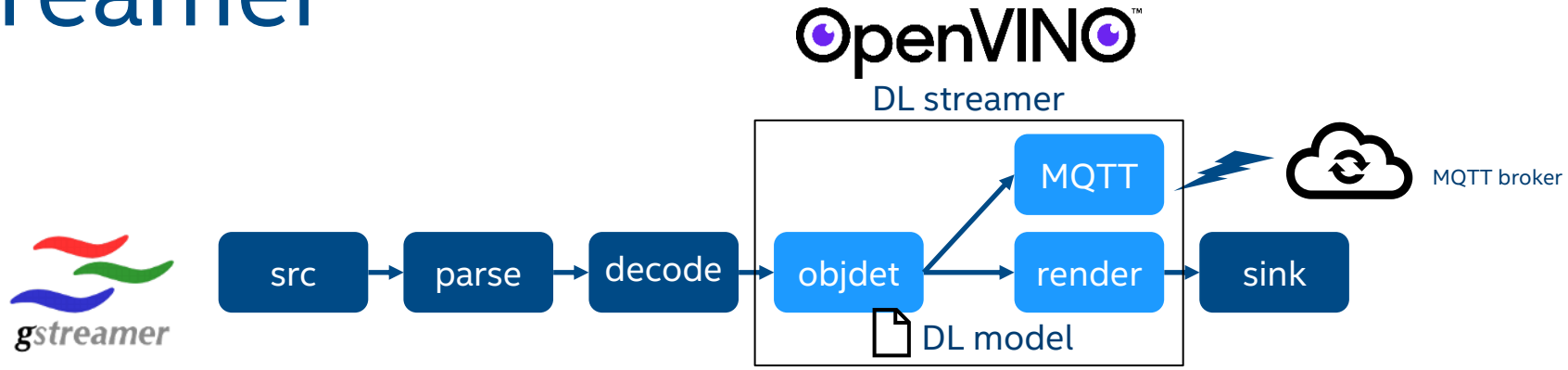
- Model Optimizer - モデル変換
- Model Downloader
  - Downloader - ダウンロード
  - Converter - IR変換
  - Quantizer - INT8化
  - Pytorch to ONNX converter
  - Caffe2 to ONNX converter

- Post-training Optimization Tool (POT) IRモデルのINT8化
- Accuracy Checker モデル精度チェック
- Cross check tool モデル間の精度チェック
- Deployment Manager RTEパッケージ作成
- Benchmark_app 汎用モデル・ベンチマーク
- VPU OpenCL compiler VPUカスタムレイヤー用
- DL Streamer gstreamer用DLエレメント
- OpenCV* 画像処理API
- DL workbench Web GUIフロントエンド

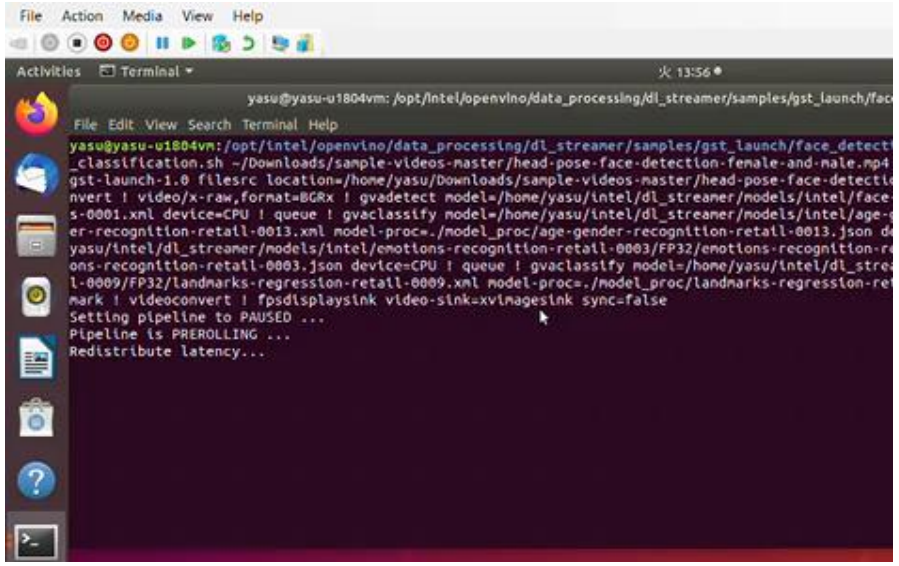
OpenVINO™ ツールキット 外部add-onツール

- NNCF - 学習時モデル最適化 (PyTorch)
- CVAT - 分散アノテーション・ツール
- Model Server - 推論サービスサーバー (gRPC、REST)
- Datumaro - データセット操作、変換
- Training Extension - インテル®OMZモデル再学習、転移学習用スクリプト、データ

DL Streamer

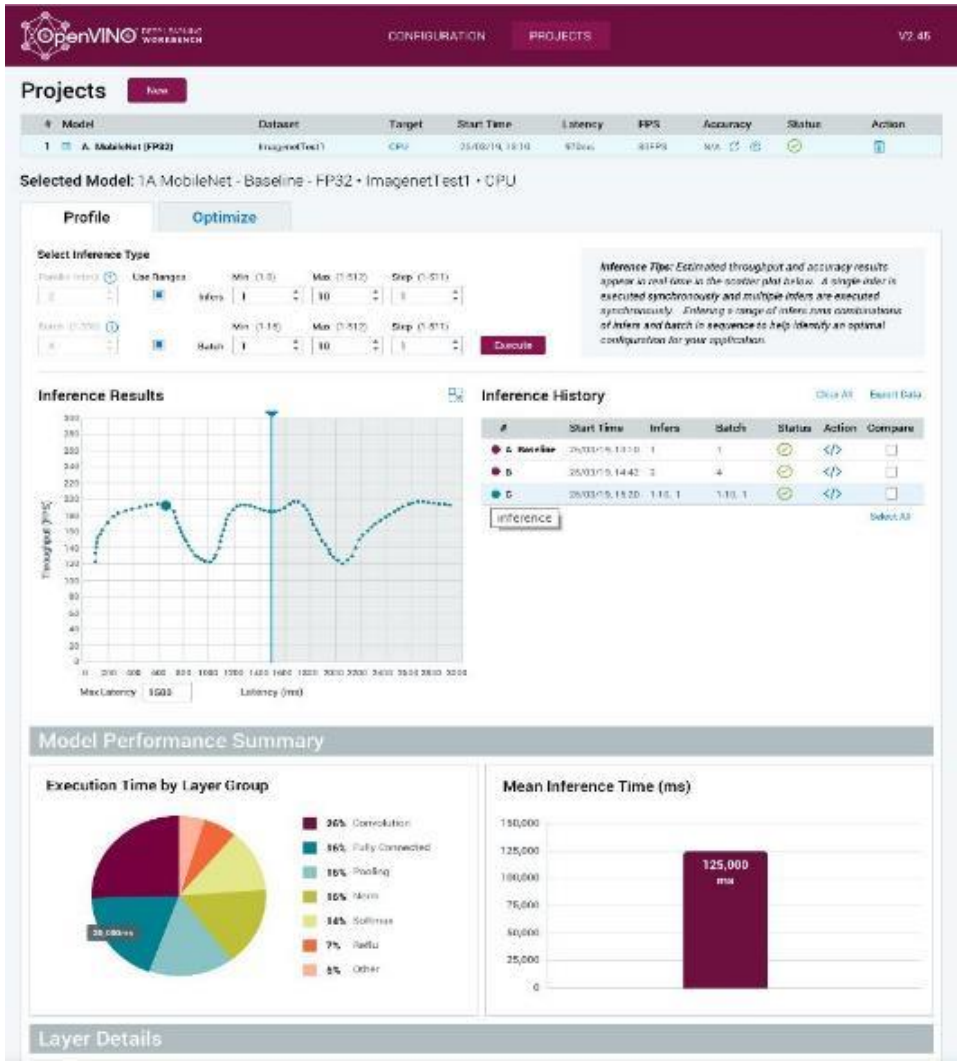


- gstreamer*メディアパイプライン中でDL推論、結果の描画などが可能
- クラス分類、物体検出、トラッキング、一般DL推論、結果のレンダリングなどなど
- 任意のPython*コードを実行するエレメントもあります



https://docs.openvinotoolkit.org/latest/gst_samples_README.html

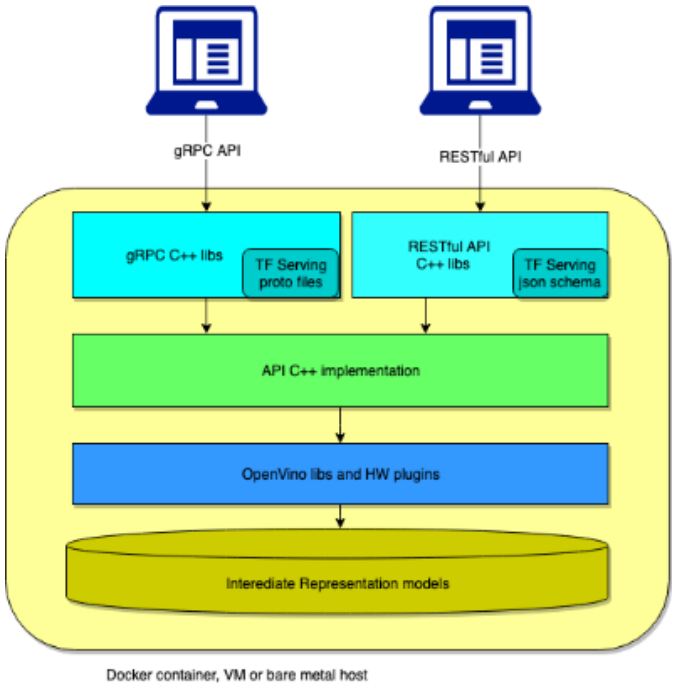
DL Workbench



OpenVINO™ ツールキット・コマンドライン・ツールのウェブ UI フロントエンド

- モデル・ダウンロード、IR変換
- パフォーマンス・データの可視化
- レイヤー情報の確認
- 自動ベンチマーク (batch、stream)
- INT8、Winograd最適化
- 精度テスト
- リモート・ベンチマーク

OpenVINO™ ツールキット・モデル・サーバー(OVMS)



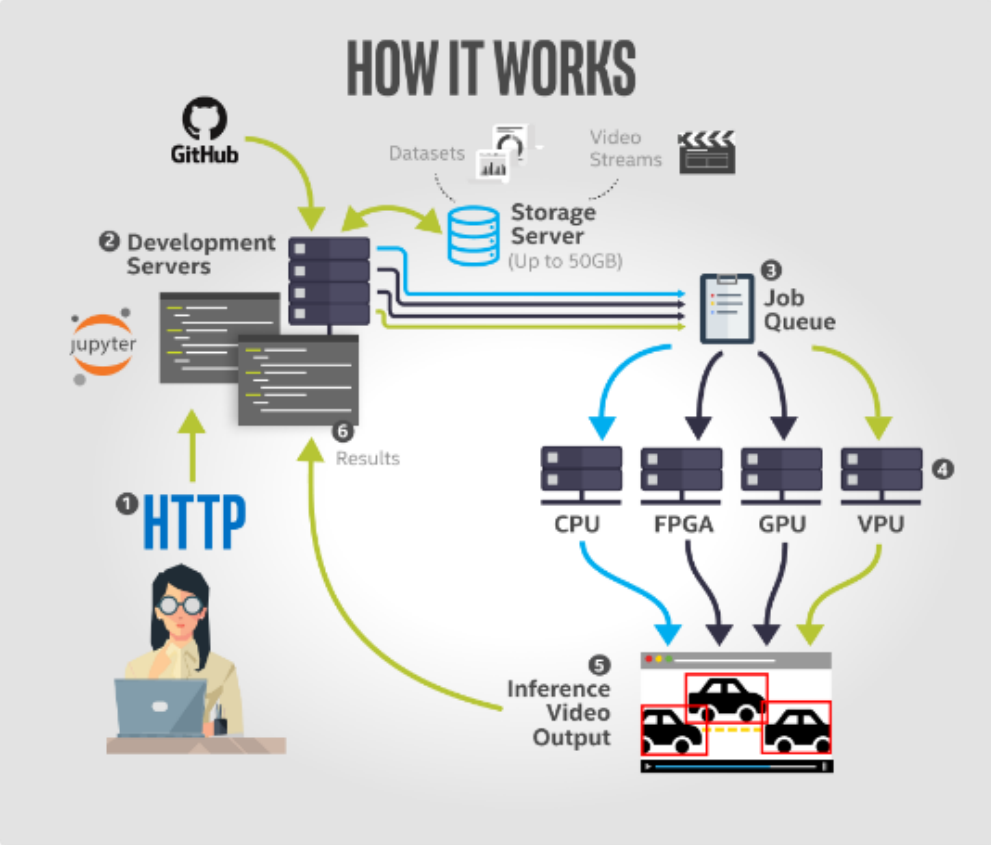
- **DL推論サービスを提供するコンテナーベースの マイクロサービス** (ベアメタル可)
- 推論エンジンはOpenVINO™ ツールキット。アクセラレーターもサポート
- 1つのサーバーで複数のクライアント・デバイスからの推論要求を処理
- 非力なIoTデバイスの推論エンジンとして活用可能
- gRPCかREST APIで推論リクエストを送出
- オープンソース
- Kubernetes*でのオート・スケーリングにも対応
- セキュリティーadd-on: モデル暗号化、ライセンス管理

その他情報

CPU / 内蔵GPU / VPU の特性

デバイス	コスト	対応モデル	特徴	説明
CPU	CPUそのもの	• FP32 / FP16 / INT8 • INT8あるいはFP32推奨 • FP16モデルは内部でFP32に変換して実行	• 汎用	• DL boostサポートCPUならINT8推論が高速(~x4 vs FP32) • FP16のモデルを与えても内部処理はFP32。速くはならない。 • サポートレイヤーが一番多い • AtomからXeonまで広いパフォーマンスレンジ • CPUの広大なメモリ空間が使える。大きめのモデルや入力が大きなモデルの実行に向く
GPU	CPU内蔵	• FP32 / FP16 / INT8 • INT8あるいはFP16推奨	• 無償のオフロードエンジン	• モデルロード時のOpenCLコンパイルに時間がかかる (2021.4で改善, CACHE_DIR) • FP16でFP32の倍近いパフォーマンスが出る時がある • サポートレイヤー比較的多い • DL boostサポートGPUならINT8推論が高速(~x4 vs FP32) • CPUとメモリ空間を共有。メモリコピーオーバーヘッドなし
VPU	HDDL, Compute Stickなど別売り	• FP16	• 外付けオフロードデバイス • 小さいモデル向き	• 外部アクセラレータなのでデータ転送オーバーヘッドはある • VPU内部メモリは小さいので比較的小型のモデルを実行するのに向く (googlenet-v1程度) • モデルが大きくなると極端に性能が落ちるケースが有る • ハードウエアDLアクセラレータx2, DSP x1を搭載しているため、同時推論数を上げて飽和させると最大のパフォーマンスを発揮

インテル® DevCloud for the Edge



- インテルが提供するクラウドベースのサービス
- 簡単な手続きで、世界中からリモートアクセス
- 各種CPU、内蔵GPU、VPU、FPGAがクラウド上に用意されている
- OpenVINO™ ツールキットのセットアップ済み
- ハードウェア選定のためのベンチマーキングが可能

https://devcloud.intel.com/edge/

パフォーマンスチューニング

- パフォーマンス最適化ガイド、最適化ツール
  - https://docs.openvinotoolkit.org/latest/tuning_for_performance.html
- ベンチマーク結果
  - https://docs.openvinotoolkit.org/latest/openvino_docs_performance_benchmarks_openvino.html

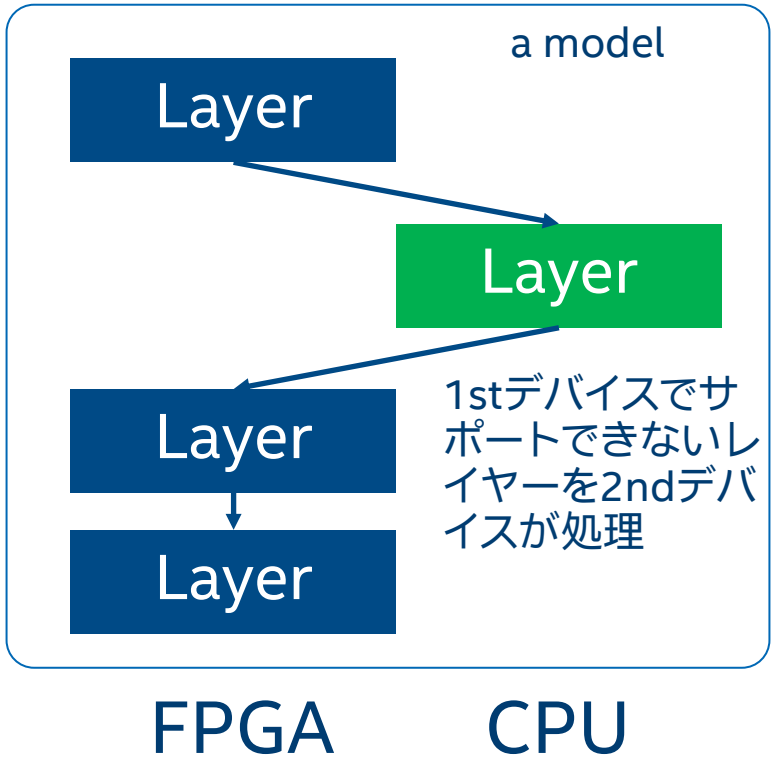
HDDL スケジューラー

- HDDL daemonは複数のVPUを管理するためのスケジューラーを持っています
  - VPUへのタスクの割り振り方、タスク/デバイス固定方法などを提供
  - うまく割り振ることでパフォーマンス向上が期待できる
- Squeeze / Tag / Stream / SGAD / Bypass スケジューラー
  - Squeeze - 推論要求を順次空いているデバイスに割り振っていく (デフォルトスケジューラー)
  - Tag - タグごとに何個VPUを割り振るか指定。モデルロード時にタグを指定。
  - Stream - RNN/LSTMなどのcontext awareモデル実行時に指定。次の推論要求も同じVPUに割り振られる
  - SGAD - 全モデルを全デバイスに先にロードしてしまう。モデル切り替えオーバーヘッドがない
  - Bypass - HDDLのスケジューラを介さないモード。ユーザープログラムで使用するVPUデバイスを指定して使用
- スケジューラーの設定はhddl_service.configで、ユーザープログラムではload_network()時のconfig指定で行う
- https://docs.openvinotoolkit.org/latest/openvino_docs_IE_DG_supported_plugins_HDDL.html

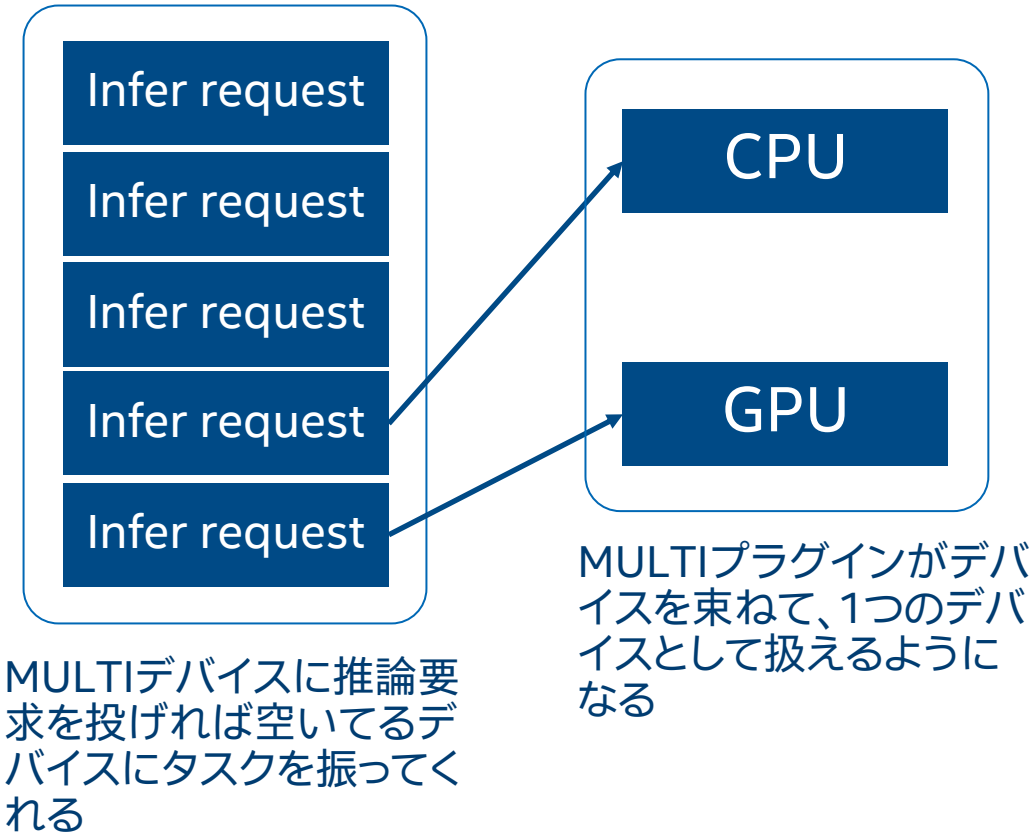
SGAD - Single Graph All Device

Special Plugins - HETERO and MULTI plugins

HETERO:FPGA,CPU



MULTI:GPU,CPU



サンプルプログラム

- OpenVINO™には多くののサンプル、デモプログラムが添付されています
  - $INTEL_OPENVINO_DIR/ deployment_tools/inference_engine/samples
    - 比較的シンプルなサンプルプログラムが置かれています。コーディング時やAPI使用方法の参考になります
  - $INTEL_OPENVINO_DIR/ deployment_tools/open_model_zoo/demos
    - 比較的複雑で高度なデモプログラムが置かれています。見た目に楽しい物が多いです。
  - $INTEL_OPENVINO_DIR/ deployment_tools/demo
    - いくつかのデモプログラムを簡単にビルド、実行できるスクリプトファイルが置かれています
  - $INTEL_OPENVINO_DIR/data_processing
    - 非画像系のデモなどが置かれています

- Pythonベースのデモのいくつかは C++で書かれたモジュールが必要なものがあります (human_pose_estimation_3d_demoなど)
  - ./build_demos.sh -DENABLE_PYTHON=YESとして、デモビルド時にPythonモジュールもビルドするように指定する必要があります
  - 出来上がったPythonモジュール(*.pyd, *.so)はPythonデモプログラムフォルダにコピーして使用します

モデル変換情報

- $INTEL_OPENVINO_DIR/deployment_tools/open_model_zoo/models/[intel|public]というディレクトリがあり、その中にモデルごとのディレクトリがあります
- モデルディレクトリの中にはいくつかのYAMLファイルが有り、model downloader, model converter, model quantizerなどが利用しています
- これらのYAMLファイルの中を見るとそのモデルの変換に必要な情報を簡単に得ることが可能です
- のモデルの変換で困った場合に、似たようなモデルの定義ファイルを参照することでヒントを得られることがあります
- モデルを取得しているURLなどもわかるので、何らかの原因でdownloaderがダウンロードに失敗する場合なども原因究明の役に立つことがあります
- **model converter実行時に表示されるModel Optimizerコマンドラインも参考になります**
  - ただし、省略できるオプションも結構表示されて少しややこしかったりします。下記の例ではピンク色のオプションはなくても変換できます

```
description: >-
  The "googlenet-v3" model is the first of the Inception family of models designed
  to perform image classification. For details about this family of models, check
  out the paper <https://arxiv.org/abs/1602.07261>.
task_type: classification
files:
  - name: googlenet-v3.tar.gz
    size: 88668554
    sha256: 7045b72a954af4dce36346f478610acdccbf149168fa25c78e54e32f0c723d6d
    source: https://storage.googleapis.com/download.tensorflow.org/models/inception_v3_2016_08_28_frozen.pb.tar.gz
postprocessing:
  - $type: unpack_archive
    format: gztar
    file: googlenet-v3.tar.gz
model_optimizer_args:
  - --reverse_input_channels
  - --input_shape=[1,299,299,3]
  - --input=input
  - --mean_values=input[127.5,127.5,127.5]
  - --scale_values=input[127.5]
  - --output=InceptionV3/Predictions/Softmax
  - --input_model=$dl_dir/inception_v3_2016_08_28_frozen.pb
framework: tf
quantizable: yes
license: https://raw.githubusercontent.com/tensorflow/models/master/LICENSE
```

モデル変換情報

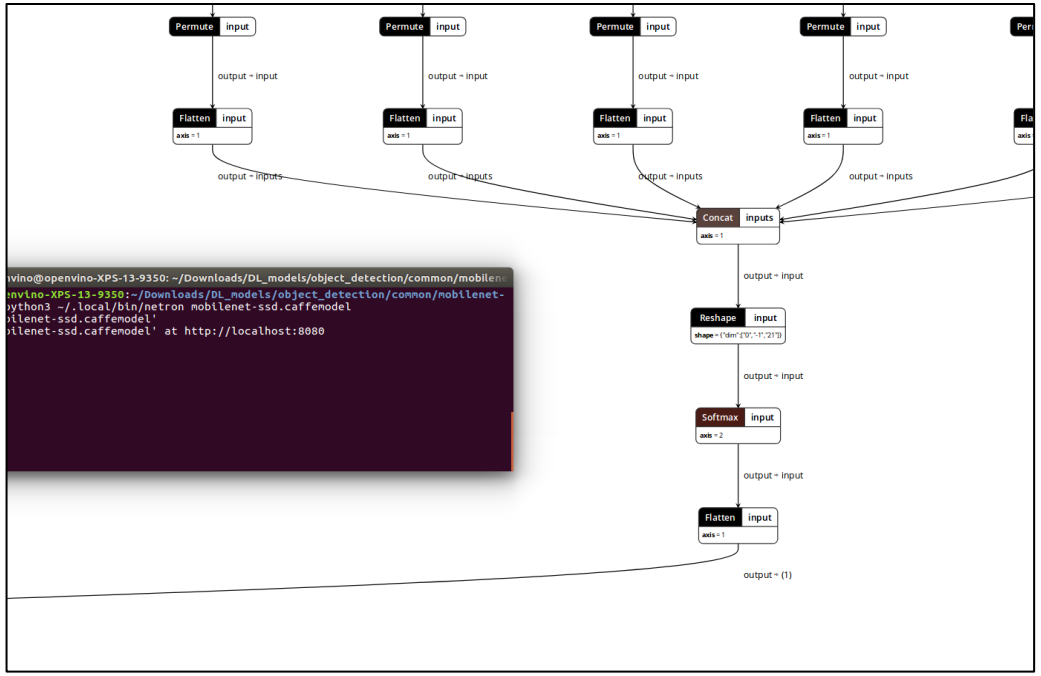
model.yamlファイル例 (一部)

```
>python converter.py --name googlenet-v1-tf
   :
   :
========== Converting googlenet-v1-tf to IR (FP32)
Conversion command:
C:\Users\yshimur1\AppData\Local\Programs\Python\Python36\python.exe -
- "C:\Program Files
(x86)\IntelSWTools\openvino_2021\deployment_tools\model_optimizer\mo.
py" --framework=tf --data_type=FP32 --
output_dir=C:\Users\yshimur1\Documents\Intel\OpenVINO\work\public\goo
glenet-v1-tf\FP32 --model_name=googlenet-v1-tf --
input_shape=[1,224,224,3] --input=input --
mean_values=input[127.5,127.5,127.5] --scale_values=input[127.5] --
output=InceptionV1/Logits/Predictions/Softmax --
input_model=C:\Users\yshimur1\Documents\Intel\OpenVINO\work\public\go
oglenet-v1-tf/inception_v1.frozen.pb --reverse_input_channels
```

model converter実行時のMOコマンドライン例

Model Visualization - Netron

- You may want to see how the DL model is designed when you got some error while converting a DL model with Model Optimizer. You can use DL model visualization tools to do it.
- pip3 install netron
  netron <model-name> (python3 ~/.local/bin/netron <model-name>)
- Open a web browser and go to localhost:8080
- **Supported models:**
  ONNX, Keras, CoreML, TensorFlow Lite,
  Caffe, Caffe2, MXNet, Tensorflow, IR
- Many other similar tools are available on the web



Network Configuration Parameters

You can enable, disable or modify the plugin features using ie.setConfig() API
Please refer to the OpenVINO™ offline document for full list of configuration parameters

https://docs.openvinotoolkit.org/latest/openvino_docs_IE_DG_supported_plugins_GPU.html

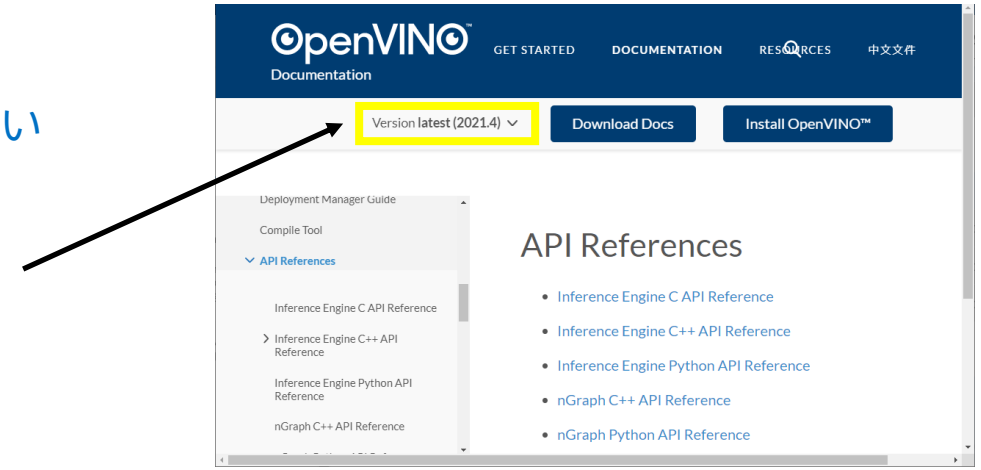
Examples:

Key	Value	Description
KEY_CPU_THREADS_NUM	Number	Number of threads CPU plugin should use. 0 for all cores.
KEY_CPU_BIND_THREAD	YES/NO	Binds inference worker threads to CPU cores
KEY_PERF_COUNT	YES/NO	Enable /Disable performance counter feature
KEY_CLDNN_PLUGIN_PRIORITY	0-3	OpenCL queue priority

e.g. ie.set_config({'CLDNN_PLUGIN_THROTTLE', '1' }, 'GPU') # Python
ie.set_config({'CPU_THREAD_NUM': '3'}, 'CPU') # Python

OpenVINO™のAPI仕様の変更について

- OpenVINO™が進化するに連れ、軽微なAPI仕様の変更が入っていたりします
- 以下のものが全てではありませんが、よく使う辺りの変更点です
- APIの詳細につきましては公式webサイトのドキュメントを御覧ください
  - https://docs.openvinotoolkit.org/latest/api_references.html
  - また、webページでは過去のドキュメント (過去のAPI仕様)も参照することが可能です。




```
# 入出力blobの名前の取得、入力blobのシェイプの取得
input_blob_name  = list(net.inputs.keys())[0]
output_blob_name = list(net.outputs.keys())[0]
input_shape  = net.inputs [input_blob_name ].shape
output_shape = net.outputs[output_blob_name].shape
```

```
# 入出力blobの名前の取得、入力blobのシェイプの取得
input_blob_name  = list(net.input_info.keys())[0]
output_blob_name = list(net.outputs.keys())[0]
inputShape  = net.input_info[input_blob_name ].tensor_desc.dims
outputShape = net.outputs   [output_blob_name].shape
```

```
def callback(status, pydata):
    global completion_count, inuse, exenet, outputBlobName,time
    request_id = pydata
    res = exenet.requests[request_id].outputs[outputBlobName]
```

```
def callback(status, pydata):
    global completion_count, inuse, exenet, outputBlobName,time
    request_id = pydata
    res = exenet.requests[request_id].output_blobs[outputBlobName]
```

本日のセミナーのコンテンツ

- GitHubレポジトリにアップロードしてあります
  - git clone https://github.com/yas-sim/ae2100-openvino-samples


intel