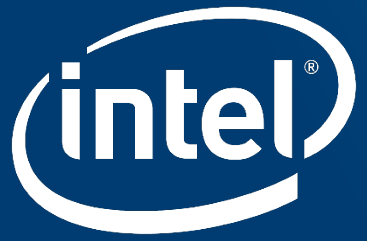


Intel® OpenVINO™ ツールキットによる AIハンズオンセミナー

～ディープラーニングモデルの簡易利用と推論の高速化～

インテル株式会社

APJデータセンター・グループ・セールス

AIテクニカル・ソリューション・スペシャリスト / AI＆アナリティクス・アドボケイト

大内山　浩

@特別ハンズオン for 大阪大学様 (2020/7/21)

本日のスケジュール

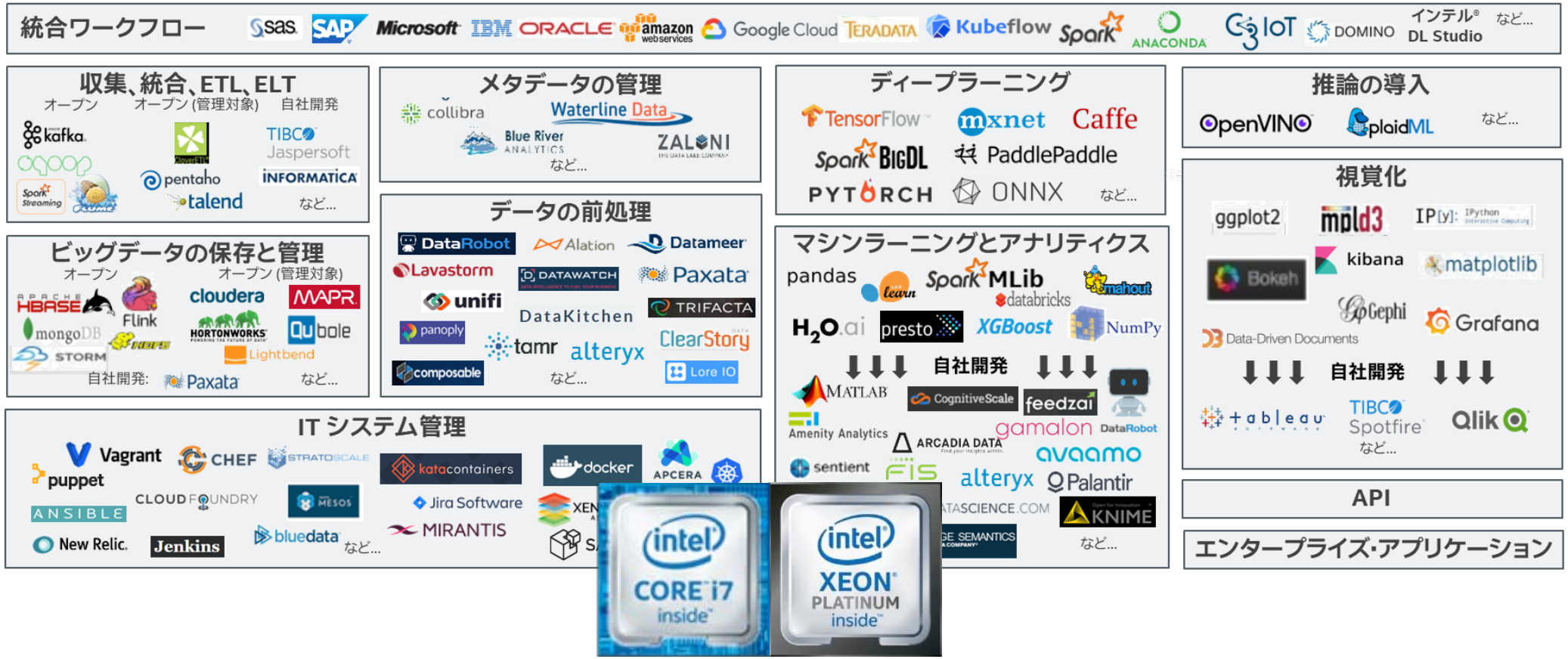
13:00 – 13:05	開会
13:05 – 13:35	IntelのAI製品概要およびOpenVINO ツールキットのご紹介
13:35 – 13:50	ハンズオンの説明と準備
13:50 – 14:50	OpenVINOハンズオンPart1 事前学習済みモデルを用いた推論実行
14:35 – 14:50	休憩
14:50 – 16:20	OpenVINOハンズオンPart2 カスタムモデルを用いた推論実行およびモデルの量子化
16:20 – 16:25	閉会

Intel AI Overview

INTELのAI製品概要、および、
OPENVINO ツールキットのご紹介

AI ”も” 動かすCPU

あらゆるワークロードに対応できる汎用性と柔軟性がCPUの特徴です。



ディープラーニングの開発ステップ

OpenVINO™

モデルの作成 （学習）	モデルの運用 （推論）
実施期間短い（数か月） ハイパフォーマンス	期間長い（数年） コストパフォーマンス
アクセラレータ （GPU、ASIC等）	CPU

ディープラーニング高速化の要 ～AVX-512 & DL Boost～

AVX-512（SIMD）が搭載され、並列演算性能の向上に寄与しております。更に、Deep Learning BoostというAI専用命令により更なるアクセラレーションが期待できます。

Intel® AVX-512

(Intel® Advanced Vector Extensions 512)

\+

Intel® DL Boost

(Intel® Deep Learning Boost)

はいってる

第10世代Ice Lakeから搭載

はいってる

Skylake世代からAVX-512搭載
Cascade Lake世代からDL Boost搭載

インテル® AI ソフトウェア: マシンラーニングとディープラーニング

Developer TOols
App Developers
SW Platform Developer

Topologies & Models
Data Scientist

Frameworks
Data Scientist

Graph
ML Performance Engineer

Kernel
ML Performance Engineer

Machine learning	Deep LEarning	Management Tools
ANALYTICS ZOO	OpenVINO	Containers
▪ Intel Distribution for Python (SKlearn, Pandas)	TensorFlow PyTorch	kubernetes Kubeflow Deep Learning Reference Stack
▪ Intel Data Analytics Acceleration Library (Intel DAAL) ▪ Intel Math Kernel Library (Intel MKL)	▪ Intel Machine Learning Scaling Library (Intel MLSL) ▪ Intel® Deep Neural Network Library (DNNL)	Data Analytics Reference Stack

Architect & DevOps

CPU

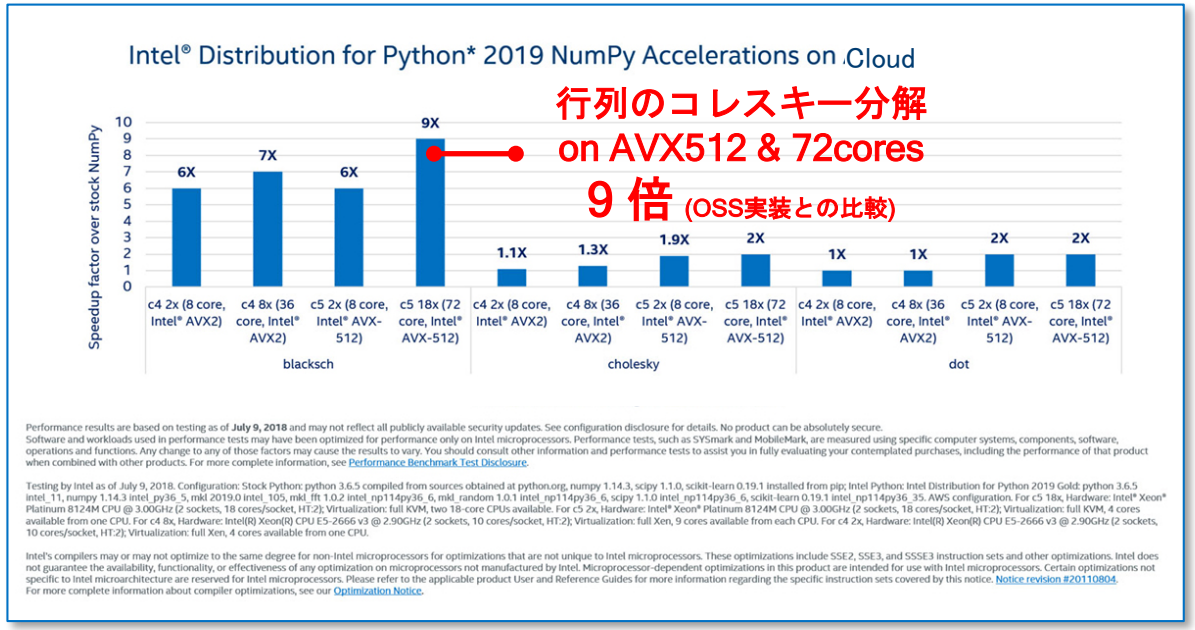
CPU ▪□ GPU ▪□ FPGA ▪□ 専用

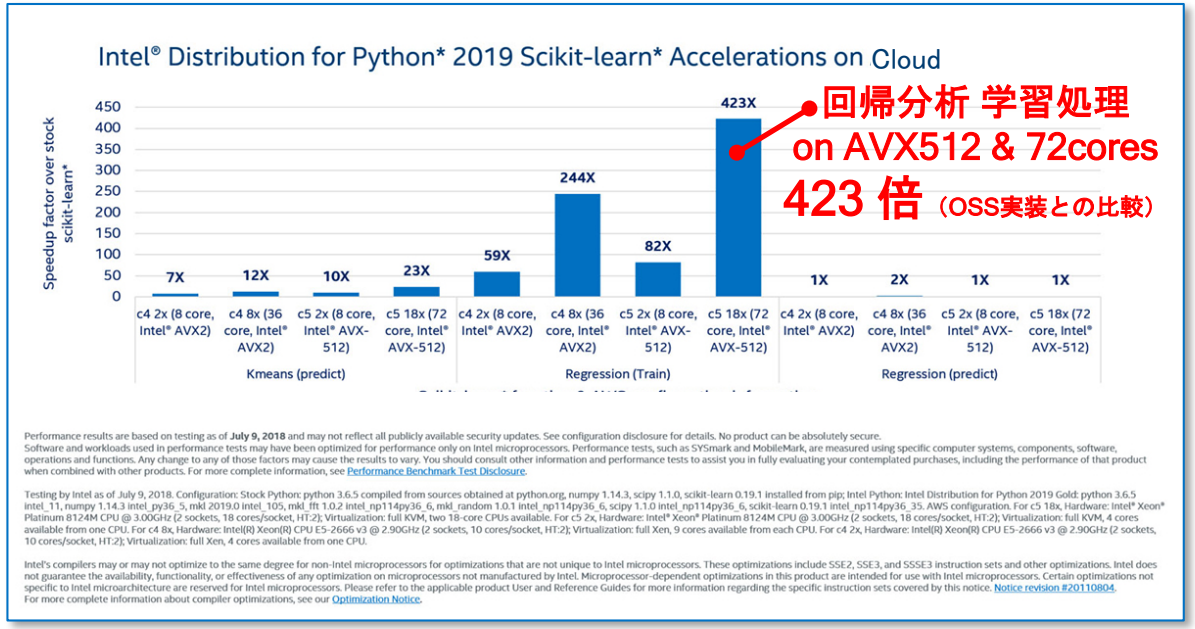
Red font products are the most broadly applicable SW products for AI users

Intel® Distribution for Python*

インテルが実装、かつ、最適化した Python、および、周辺ライブラリ

- Numpy
- Pandas
- Scipy
- Scikit-learn
- XGBoost
- TensorFlow
- etc..





https://software.intel.com/en-us/distribution-for-python/benchmarks

Intel® Distribution for Python* – 導入方法一覧

Build from Source

https://software.intel.com/en-us/distribution-for-python/choose-download/linux

Anaconda

https://software.intel.com/en-us/articles/using-intel-distribution-for-python-with-anaconda

Pip

https://software.intel.com/en-us/articles/installing-the-intel-distribution-for-python-and-intel-performance-libraries-with-pip-and

Docker Image

https://software.intel.com/en-us/articles/docker-images-for-intel-python

Linux Repositories

YUM

https://software.intel.com/en-us/articles/installing-intel-free-libs-and-python-yum-repo

APT

https://software.intel.com/en-us/articles/installing-intel-free-libs-and-python-apt-repo

ディープラーニング・フレームワーク（インテルによる最適化）

拡張性
- 負荷分散の効率化
- 同期イベント、多対多通信の低減

すべてのコアの利用
- OpenMP、MPI
- 同期イベント、逐次処理コードの低減
- 負荷分散の効率化

ベクトル化 / SIMD
- SIMD レーンごとのユニットストライド式アクセス
- 高効率ベクトル制御
- データ・アライメント

メモリー / キャッシュの効率的な使用
- ブロッキング
- データの再利用
- プリフェッチ
- メモリー割り当て

= TensorFlow, mxnet, Caffe2, PyTorch, BigDL + Spark

インストール・ガイド
http://ai.intel.com/framework-optimizations/ （英語）を参照

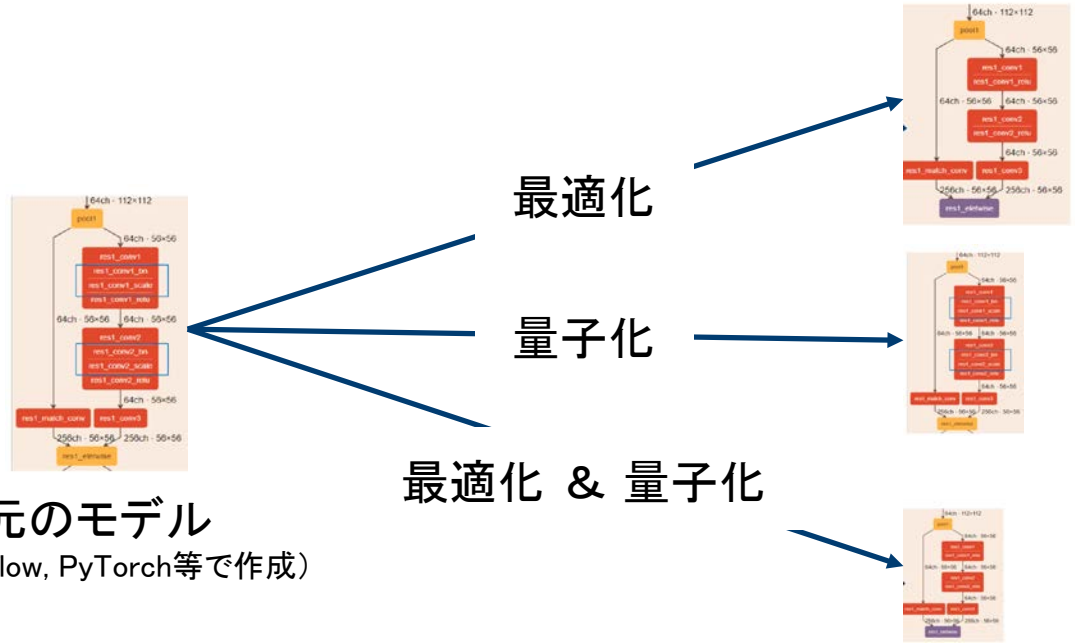
その他のフレームワーク最適化が進行中（PaddlePaddle*、CNTK* など）

関連項目: Python* 用マシン・ラーニング・ライブラリー（Scikit-learn、Pandas、NumPy）、R（Cart、randomForest、e1071）、分散型（MlLib on Spark*、Mahout*）
* 現在は限定的に提供。
最適化に関する注意事項

更なる推論高速化のためにモデルの最適化と量子化

最適化：不要なOpsの除去、複数のOpsの統合などによりモデルをスマート化

量子化*：モデル内部の数値表現をFP32→FP16/INT8等に変換することでスリム化



by OpenVINO™

by TensorFlow PyTorch

※各フレームワークごとに
量子化ツールを用意

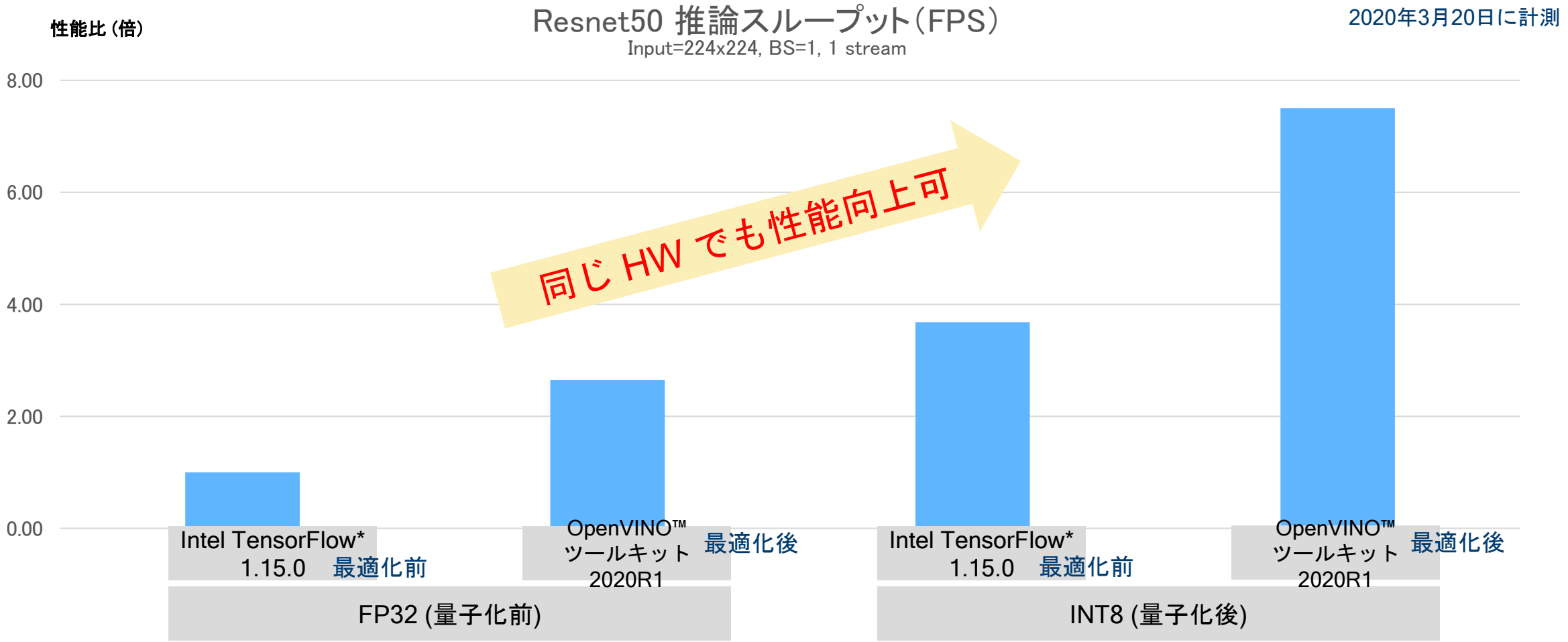
by OpenVINO™

- 2020年5月現在、Intel Deep Learning Boost（VNNI）が搭載された 第 2 世代インテル® Xeon® スケーラブル・プロセッサー、第10世代 インテル® Core™ プロセッサー(Ice Lakeのみ) にてより効力を発揮する。

ディープラーニング推論処理ベンチマーク

参考値

インテル® Xeon® Gold 6254 プロセッサー @ 2.10GHz (18 cores × 1 sockets)



注）インテル社員による性能確認のための個人的なベンチマーク結果であり、インテルの公式結果ではありません。

OpenVINO™ ツールキット

https://software.intel.com/en-us/openvino-toolkit

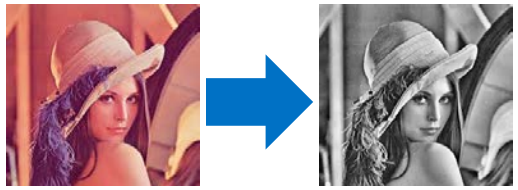
画像処理とディープラーニング推論のためのライブラリスイートです。3つの特徴をぜひご理解ください。

OpenVINO™

コンピュータビジョンアプリ向け
ソフトウェア・ライブラリ・スイート
（Python、C++対応）

Ubuntu,
CentOS,
Yocto,
Win10
MacOS

画像処理



ディープ
ラーニング推論

【特徴１】AIパーツ

【特徴2】モデルコンパイラ

【特徴3】
ヘテロジニアス・オーケストレータ

特徴1．AIパーツとしてのOpenVINO

学習済みモデルとサンプルを多数提供しております。アプリケーション導入の迅速化を促進します。

OpenVINO™ ツールキットのインテル® ディストリビューション提供の事前学習済みモデル

- 年齢と性別
- 顔検出 – 標準および拡張
- 頭の位置
- 人物検出 – 眼高 / 高角検出
- 人、車、自転車の検出
- ナンバープレート検出: 小型および前面
- 車両メタデータ
- 人体姿勢推定
- 行動認識 – エンコーダーとデコーダー
- テキストの検出と認識
- 車両検出
- 小売環境
- 歩行者検出
- 歩行者と車両の検出
- 横断者の属性認識
- 感情認識
- 特定の人物をさまざまな動画で識別 – 標準および拡張
- 顔のランドマーク検出
- 視線推定
- 路側物の識別
- 高度な路側識別
- 人の検出と行動認識
- 人の再識別 – 極小 / 超高速
- 顔の再識別
- ランドマーク回帰
- スマート・クラスルームのユースケース
- 単一画像超解像 (3 モデル)
- インスタンス・セグメンテーション
- など…

バイナリーモデル

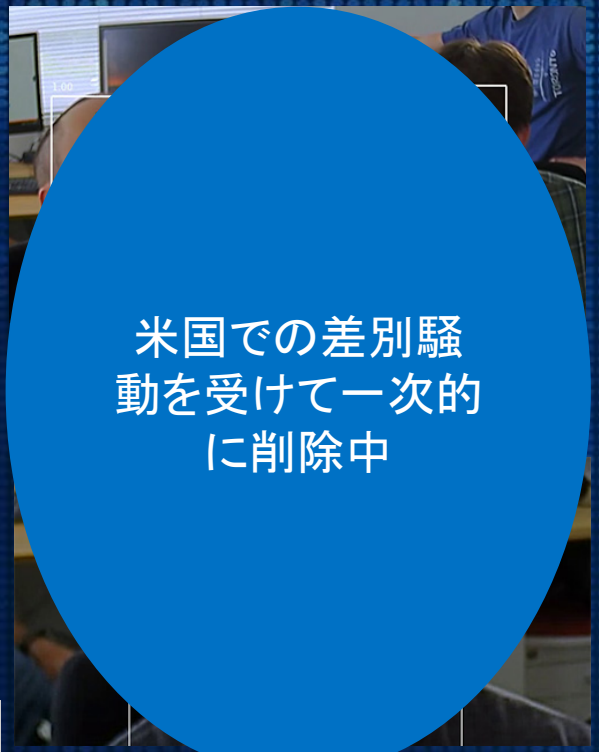
- 顔検出バイナリー
- 歩行者検出バイナリー
- 車両検出バイナリー
- ResNet50 バイナリー

OpenVINO™ツールキットに含まれる すぐに利用可能なインテルの学習済みモデルの例（無償）

性別・年齢

IMAGE INPUT	RESULT
	Female, 18.97
	Male, 26.52
	Male, 33.41

顔認識



米国での差別騒動を受けて一次的に削除中

人物検出

行動検出

テキスト認識

超解像

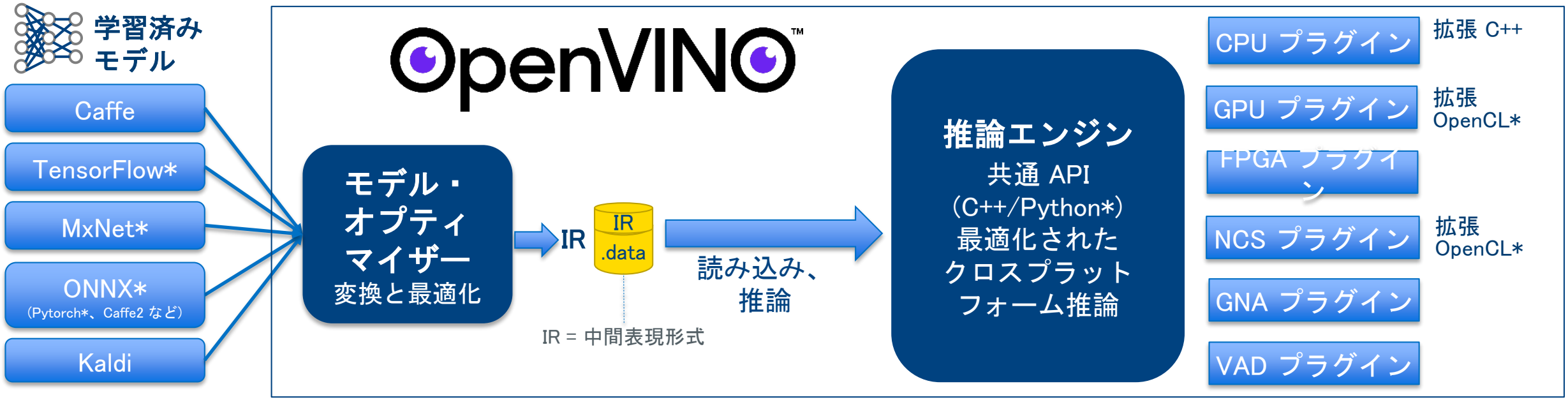
人・車両検出

人のポーズ推論

オープンな開発者コミュニティーで公開されているモデルに加えて、インテルが開発した学習済みモデル 200 種類以上を無償で提供

https://software.intel.com/en-us/openvino-toolkit/documentation/pretrained-models

特徴2:モデルコンパイラとしてのOpenVINO



モデル・オプティマイザー

- **概要**: 学習済みモデルをインポートし、中間表現に変換する Python* ベースのツール
- **重要な理由**: トポロジー変換に基づく抑制により、ハードウェアに適したデータ型に変換することで、パフォーマンスを最大化。

推論エンジン

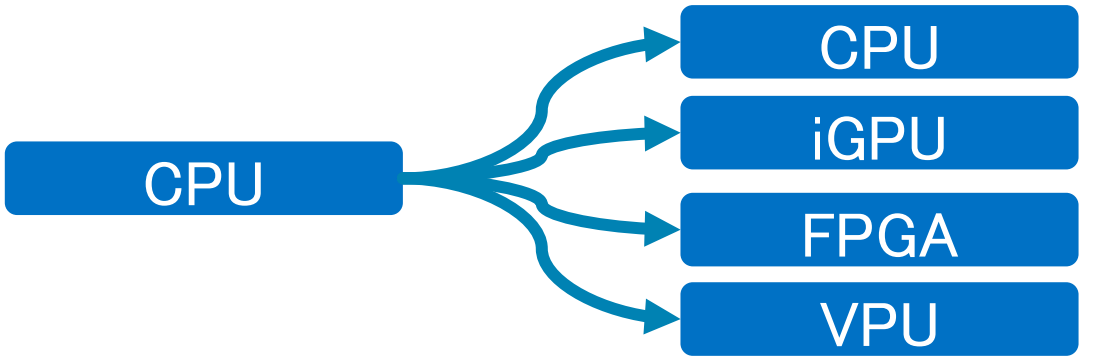
- **概要**: 高レベルの推論 API
- **重要な理由**: インターフェイスは、ハードウェアのタイプに応じた動的読み込みのプラグインとして実装。複数のコードを実装および管理することなく、タイプごとに最適なパフォーマンスを実現可能。

GPU = グラフィックス・プロセシング・ユニット / インテル® プロセッサー・グラフィックスが統合されたインテル® CPU

VAD = ビジョン・アクセラレーター・デザイン・プロダクト。FPGA バージョンと 8 つの Myriad™ X バージョン

特徴3：ヘテロジニアス・オーケストレータとしてのOpenVINO

複数種類のチップが混在するヘテロジニアス環境においても容易に各チップを切り替え、または、併用可能です。

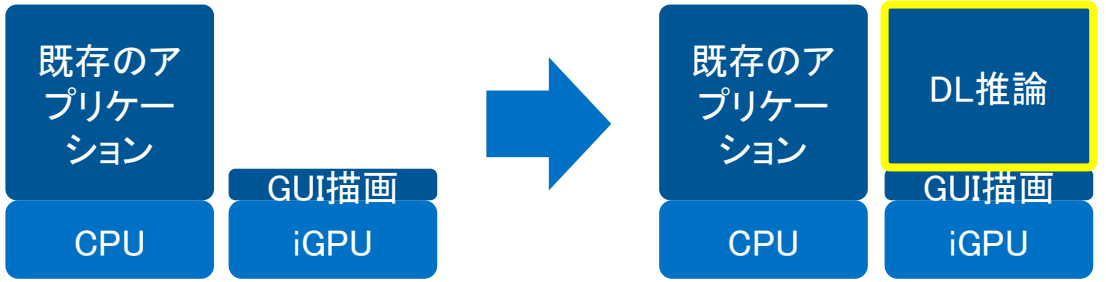


手元のPCで評価、開発着手

必要パフォーマンスが分かった時点でデバイス切り替え

- 厳密な負荷見積もり、ハードウエア選定なしに、すぐに評価、開発着手可能

CPUの処理をiGPUへオフロード



複数チップ併用時にロードバランシング

お使いのPCで始められるOpenVINOでのDL推論

OpenVINO™

Windows 10
Ubuntu, CentOS, Yocto
Mac OS

事例：理化学研究所 様
胸部疾患の自動診断用モデルの推論性能向上

Before Optimization←　→ After Optimization

約2.2万枚のテスト画像を
バッチ処理で推論

744 sec

on
他社アクセラレータ

11,177 sec
(Baseline)

on
Xeon 6252 x2

最適化

- モデルをONNXに変換
- OpenVINOのモデルオプティマイザーでONNX→IRへ変換
- OpenVINOの推論エンジン上で同期実行

x10.0 → 1,116 sec

量子化

- OpenVINOの量子化ツールにてIR内一部のレイヤーの数値表現をINT8へ変換（ツールのカスタマイズ含む）
- OpenVINOの推論エンジン上で同期実行（VNNI利用）

x3.1 → 359 sec

並列化

- OpenVINOの推論エンジン上で非同期実行（8並列で推論処理を実行）

x1.4 → 251 sec

x 44.5
against Baseline

上記対応内容は下記Githubを参照
https://github.com/taneishi/CheXNet
（計算科学研究機構 種石 様のレポジトリ）

既存システムがディープラーニングで大変身

OpenVINO

既存のインテル・アーキテクチャー搭載システムに OpenVINO™ツールキットを使用して付加価値を追加

JLK INSPECTION ヘルスケア

インテル® NUC を利用して胸部X線画像のAIスクリーニングをレントゲン・バス内で実現。OpenVINO™ツールキットにより判定時間が**4.23秒から2.81秒に短縮。** 医療のサービスレベル向上に貢献。

ZEROFOX インターネット・セキュリティー

ソーシャル・メディア等におけるデジタル・アタックの検出。ディープフェイクやブランドを毀損する情報の検出。OpenVINO™ツールキットの利用により**2.3倍の性能向上、50%の遅延低減**を達成。

dc water is life® 下水管の点検・分析

下水管の点検用動画の診断をマニュアルからAI化。インテル® Xeon スケーラブル・プロセッサーとOpenVINO™ツールキットを利用して診断時間を**80%削減**。正確性も**20 %向上。**

Retail Business Services 無人店舗

既存のインテルCPU搭載POSシステムを拡張して、自動で商品棚の在庫確認、顧客の行動分析、キャッシュレス・チェックアウトの無人店舗ソリューションを実現。

Success Stories ▶ https://intel.com/openvino-success-stories

OpenVINOのダウンロード&インストール

- ダウンロード
  - https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/choose-download.html

- インストール
  - Linux版
  - https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_linux.html
  - Windows 10版
  - https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_windows.html

OpenVINOの簡単なインストール方法

- Docker版
  - 公式レポジトリ
  - https://hub.docker.com/u/openvino

  - 非公式レポジトリ（by大内山）
  - https://hub.docker.com/u/hiouchiy

Preparation

ハンズオンの説明と準備

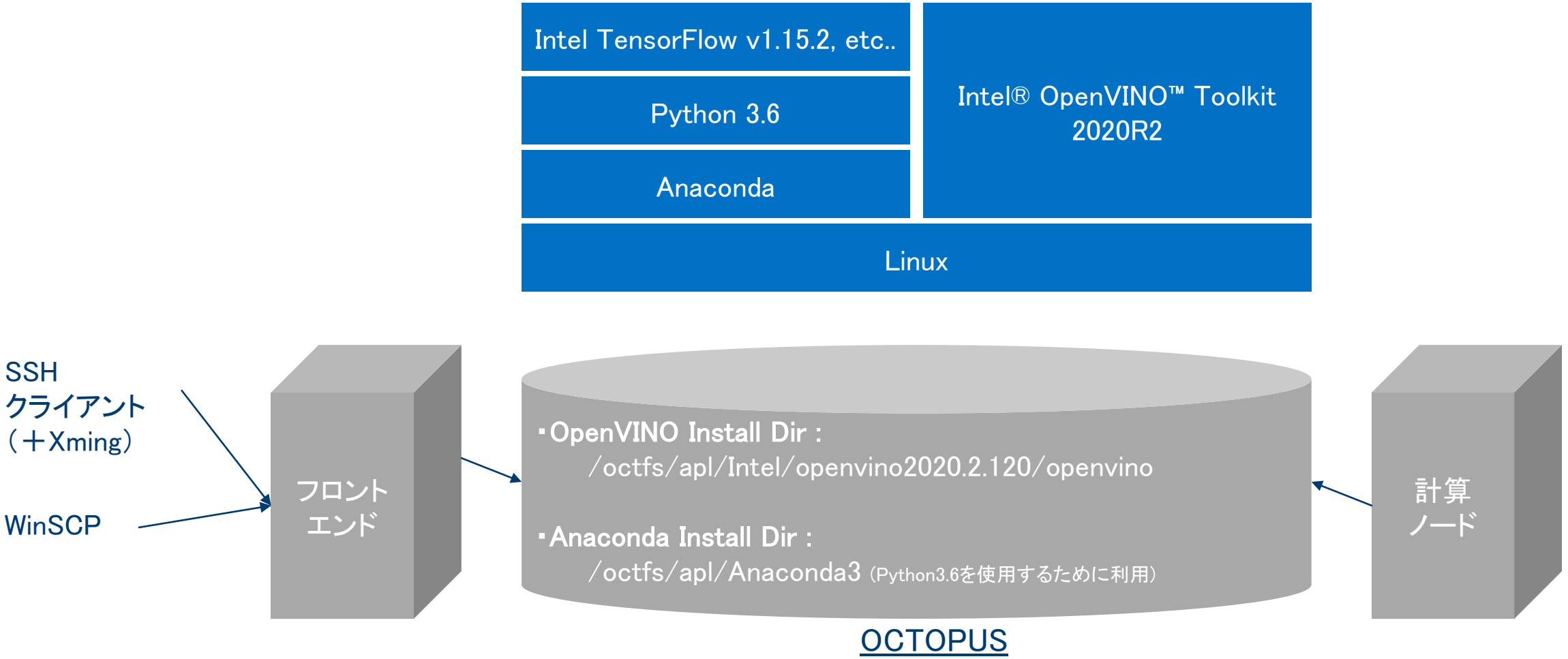
ハンズオンについて

ハンズオンは2つのパートに分けて実施します。

Part1では、AIパーツとしてのOpenVINOに着目し、OpenVINOが持つ様々な事前学習済みモデルの使い方を体験いただきます。

Part2では、モデルコンパイラとしてのOpenVINOに着目し、実際にカスタムモデルを作成し、そのモデルをOpenVINOによってCPU上で高速化する手順を体験いただきます。

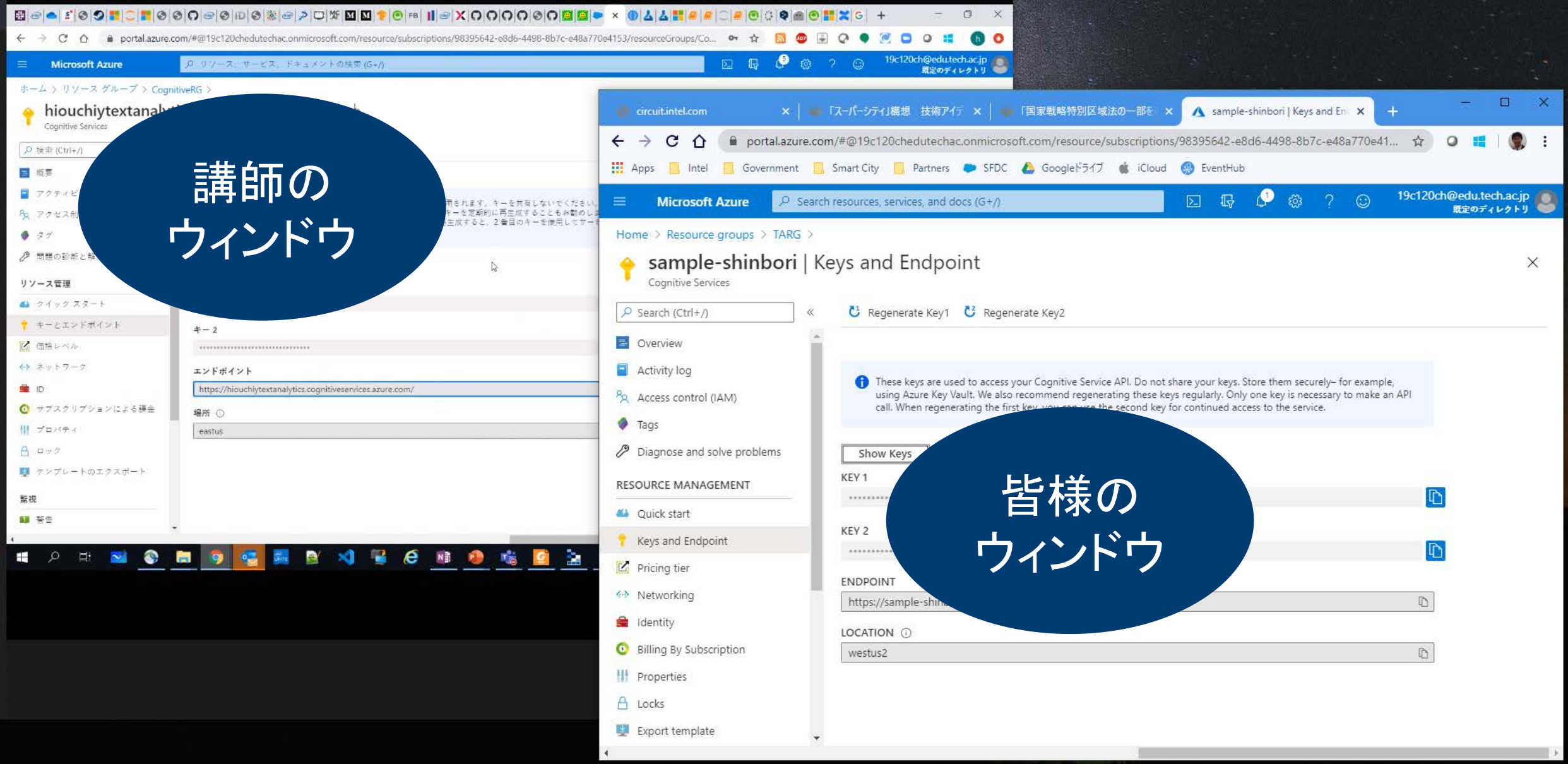
今回のハンズオン環境



セカンドモニターが無い方はこのようにウィンドウを配置ください

講師の
ウィンドウ

皆様の
ウィンドウ



OpenVINO Hands-on Part1

PART1：
事前学習済みモデルを用いた推論実行

intel

ハンズオンPart1の概要

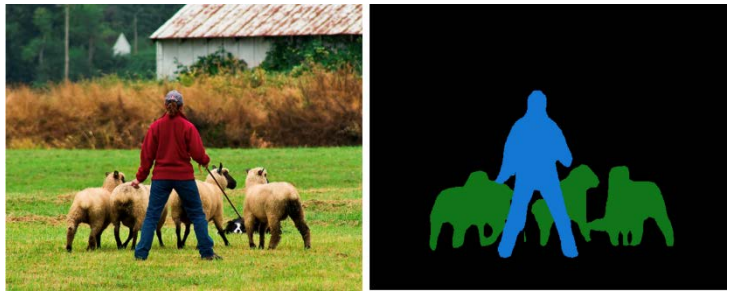
- OpenVINOに用意されている下記の事前学習済みモデルを動かします。
  - 画像セグメンテーション
  - 自動彩色（Colorization）
  - 手書き日本語認識
- 主な手順は下記の通りです。
  1. 事前学習済みモデルのダウンロード（OpenVINOモデル管理サーバーより）
  2. （必要に応じて）モデルをOpenVINOのIR形式に変換（最適化）
  3. IR形式のモデルをPythonスクリプトで実行

ステップ. 0 諸準備

- SSHクライアント（RLoginがおすすめ）でOCTOPUSへログイン
- 下記コマンドを実行する
  - source /octfs/apl/Anaconda3/bin/activate
    Anacondaを起動（Python3.6を使用するため）
  - source activate openvino
    本セミナー用の仮想環境に切り替え
  - source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh
    OpenVINOの各モジュールへのパスを環境変数に追加
- 下記コマンドを実行し、OCTOPUS上の画像表示用のウィンドウ（ImageMagick）を立ち上げておく
  - display &

- WinSCPをOctopusへ接続しておく

ステップ. 1 画像セグメンテーション



下記の順番でコマンドを実行してください

cd ~

cp -r /octfs/apl/Intel/openvino2020.2.120/openvino/inference_engine/demos/python_demos/instance_segmentation_demo/ .

cd instance_segmentation_demo

python /octfs/apl/Intel/openvino2020.2.120/openvino/deployment_tools/tools/model_downloader/downloader.py --name instance-segmentation-security-1025

wget https://raw.githubusercontent.com/hiouchiy/IntelAI_and_Cloud/master/Azure/handson/my_instance_segmentation.py

wget https://isuta.jp/wp-content/uploads/2014/05/top3.jpg

ImageMagick上で top3.jpgを表示する。

下記のいずれかのコマンドを実行する。

- フロントエンド：python my_instance_segmentation.py -m intel/instance-segmentation-security-1025/FP32/instance-segmentation-security-1025.xml --label coco_labels.txt --no_keep_aspect_ratio -i top3.jpg --no_show
- 計算ノード：次ページ参照

ImageMagick上で after_top3.jpgを表示する。

適当な画像を用意してモデルにセグメンテーションさせてみてください。

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

python my_instance_segmentation.py -m intel/instance-segmentation-security-1025/FP32/instance-segmentation-security-1025.xml --label coco_labels.txt --no_keep_aspect_ratio -i top3.jpg --no_show
```

補足：Model DownloaderとOpen Model Zoo

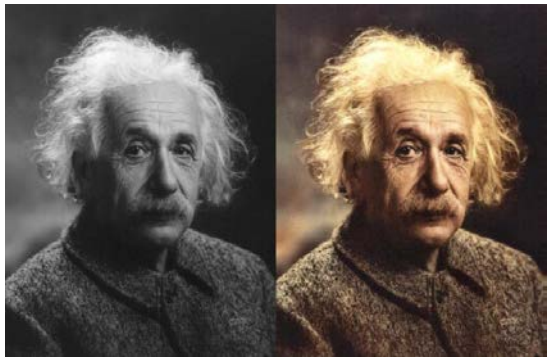
お好きなモデルをダウンロード可能

Model Downloader
$OPENVINO_INSTALL_DIR/deployment_tools/tools/model_downloader/downloader.py

python $OPENVINO_INSTALL_DIR/deployment_tools/tools/model_downloader/downloader.py –print_all
ですべての事前学習済みモデルの一覧が確認できます。

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

ステップ. 2 自動彩色（Colorization）



下記の順番でコマンドを実行してください

cd ~

cp -r /octfs/apl/Intel/openvino2020.2.120/openvino/inference_engine/demos/python_demos/colorization_demo/ .

cd colorization_demo

python /octfs/apl/Intel/openvino2020.2.120/openvino/deployment_tools/tools/model_downloader/downloader.py --list models.lst

python /octfs/apl/Intel/openvino2020.2.120/openvino/deployment_tools/tools/model_downloader/converter.py --name colorization-v2

wget https://raw.githubusercontent.com/hiouchiy/IntelAI_and_Cloud/master/Azure/handson/mycolorization.py

wget https://github.com/hiouchiy/IntelAI_and_Cloud/raw/master/Azure/handson/black_white.jpg

ImageMagick上で black_white.jpgを表示する

下記いずれかのコマンドを実行する

- フロントエンド：python mycolorization.py --coeffs public/colorization-v2/colorization-v2.npy -m public/colorization-v2/FP32/colorization-v2.xml -i black_white.jpg
- 計算ノード：次のページを参照

ImageMagick上で colorized_ black_white.jpgを表示する

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

python mycolorization.py --coeffs public/colorization-v2/colorization-v2.npy -m public/colorization-v2/FP32/colorization-v2.xml -i black_white.jpg
```

ステップ. 3 手書き日本語文字認識

下記の順番でコマンドを実行してください

cd ~

cp -r /octfs/apl/Intel/openvino2020.2.120/openvino/inference_engine/demos/python_demos/handwritten_japanese_recognition_demo/ .

cd handwritten_japanese_recognition_demo

python /octfs/apl/Intel/openvino2020.2.120/openvino/deployment_tools/tools/model_downloader/downloader.py --list models.lst

python handwritten_japanese_recognition_demo.py -m intel/handwritten-japanese-recognition-0001/FP32/handwritten-japanese-recognition-0001.xml -d CPU -cl data/kondate_nakayosi_char_list.txt -i data/test.png

wget https://github.com/hiouchiy/IntelAI_and_Cloud/raw/master/Azure/handson/myname.png

ImageMagick上でmyname.pngを表示する

下記いずれかのコマンドを実行する

- フロントエンド：python handwritten_japanese_recognition_demo.py -m intel/handwritten-japanese-recognition-0001/FP32/handwritten-japanese-recognition-0001.xml -d CPU -cl data/kondate_nakayosi_char_list.txt -i myname.png
- 計算ノード：次のスライドを参照

適当な手書き文字を書いてモデルに認識させてみてください。

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

python handwritten_japanese_recognition_demo.py -m intel/handwritten-japanese-recognition-0001/FP32/handwritten-japanese-recognition-0001.xml -d CPU -cl data/kondate_nakayosi_char_list.txt -i myname.png
```

おまけ：その他のモデルも動かしてみましょう

OpenVINO samples & demo（公式ドキュメント）

- https://docs.openvinotoolkit.org/latest/_demos_README.html
- https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Samples_Overview.html
- https://github.com/opencv/open_model_zoo

OpenVINO samples & demo（非公式ドキュメント）

- https://github.com/yas-sim
- https://github.com/hiouchiy
- https://openvino.jp/

OpenVINO Hands-on Part2

カスタムモデルを用いた推論実行およびモデルの量子化

ハンズオンPart2の概要

- 独自のモデルを作成し、それをOpenVINOで動かすことでどの程度推論性能が向上するかを見ていきます。
- 主な手順は下記の通りです。

1. カスタムモデルの作成（Keras + TensorFlowでResNet50を転移学習）
2. TensorFlowで推論（通常およびCPU最適化の2パターン実行）
3. モデルをOpenVINOのIR形式に変換（最適化）し、OpenVINO上で実行
4. モデルをOpenVINOで量子化（INT8化）し、OpenVINO上で実行

ステップ. 0 諸準備

- SSHクライアント（RLoginがおすすめ）でOCTOPUSへログイン
- 下記コマンドを叩く
  - source /octfs/apl/Anaconda3/bin/activate
  - source activate openvino
  - source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

ステップ. 1 カスタムモデルの準備
37種類の犬猫を分類するモデル (ResNet50)の作成


TensorFlow　Keras

下記の順番でコマンドを実行してください

cd ~

mkdir model_opt_qtz

cd model_opt_qtz

wget https://hiouchiy.blob.core.windows.net/share/data.zip

mkdir train_data

mv data.zip train_data/

unzip train_data/data.zip -d train_data/

wget https://raw.githubusercontent.com/hiouchiy/IntelAI and Cloud/master/Azure/handson/train_model.py

python train_model.py　（注：SciPy not foundエラーが出た場合は、'pip install scipy --user'を実行下さい。）

学習が始まったら強制終了（Ctrl+C）してください。これはCPUが本気を出していいない性能です。以下のように環境変数を追加下さい

KMP_AFFINITY=granularity=fine,compact,1,0 KMP_BLOCKTIME=1 KMP_SETTINGS=1 OMP_NUM_THREADS=24 python train_model.py

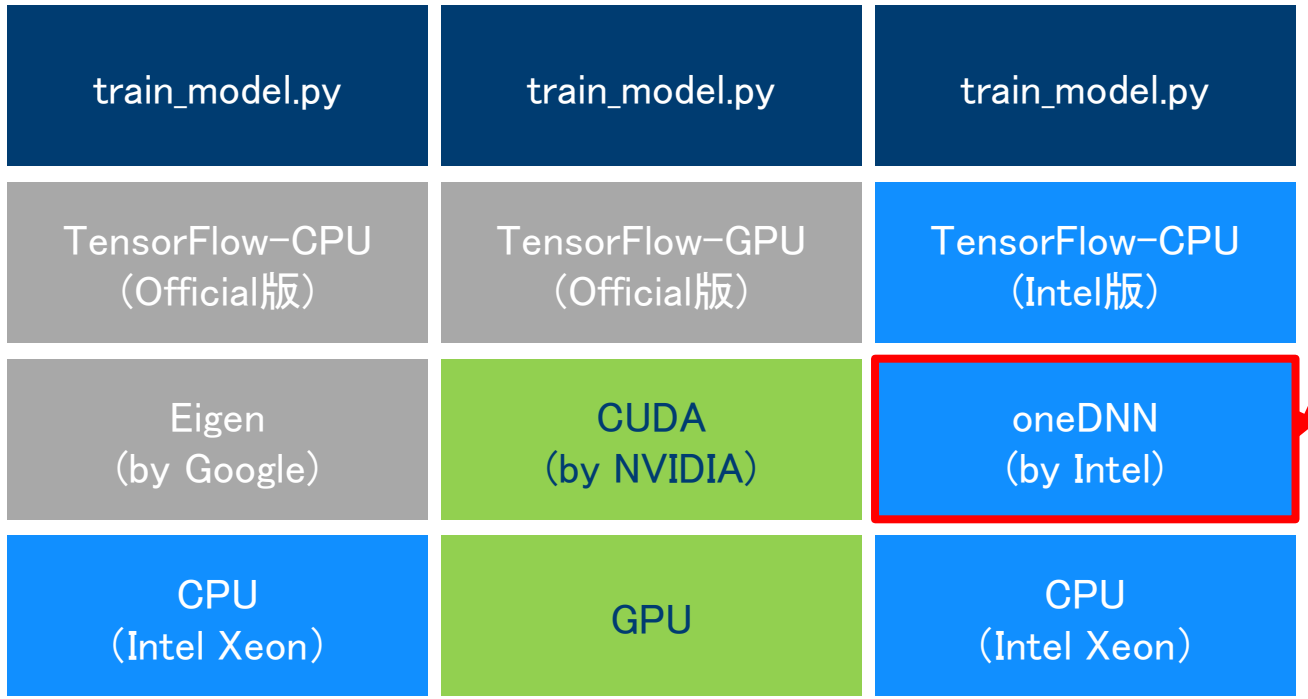
学習終了後、tf_model/mobilenet.pbというファイルが出来ていることを確認下さい。

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

KMP_AFFINITY=granularity=fine,compact,1,0 KMP_BLOCKTIME=1 KMP_SETTINGS=1 OMP_NUM_THREADS=24
python train_model.py
```

解説：謎の環境変数

4つの環境変数はoneDNNの中で動いているOpenMP用です。



OpenMP* パラメーターの推奨設定

KMP_AFFINITY=granularity=fine,compact,1,0

KMP_BLOCKTIME=1

KMP_SETTINGS=1

OMP_NUM_THREADS=物理コアの数

https://software.intel.com/content/www/us/en/develop/articles/maximize-tensorflow-performance-on-cpu-considerations-and-recommendations-for-inference.html

ステップ. 2 TensorFlowで推論実行

下記の順番でコマンドを実行してください

cd ~

cd model_opt_qtz

wget https://raw.githubusercontent.com/hiouchiy/IntelAI_and_Cloud/master/Azure/handson/infer_script.py

python infer_script.py --input_graph tf_model/resnet50.pb --num_images 300

これはCPUが本気を出していいない性能です。同じコマンドを下記の環境変数と共に実行してください。

- フロントエンド：KMP_AFFINITY=granularity=fine,compact,1,0 KMP_BLOCKTIME=1 KMP_SETTINGS=1 OMP_NUM_THREADS=24 python infer_script.py --input_graph tf_model/resnet50.pb --num_images 300
- 計算ノード：次のページを参照

推論性能が向上していることを確認してください。

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

KMP_AFFINITY=granularity=fine,compact,1,0 KMP_BLOCKTIME=1 KMP_SETTINGS=1 OMP_NUM_THREADS=24
python infer_script.py --input_graph tf_model/resnet50.pb --num_images 300
```

ステップ. 3 OpenVINOで推論実行

下記の順番でコマンドを実行してください

cd ~

cd model_opt_qtz

python /octfs/apl/Intel/openvino2020.2.120/openvino/deployment_tools/model_optimizer/mo.py --input_model=tf_model/resnet50.pb --input_shape=[1,224,224,3]

次のいずれかのコマンドで推論実行してください。

- フロントエンド：python infer_script.py --input_graph resnet50.xml --num_images 300 --openvino
- 計算ノード：次のページを参照

推論性能が向上していることを確認してください。

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

python infer_script.py --input_graph resnet50.xml --num_images 300 --openvino
```

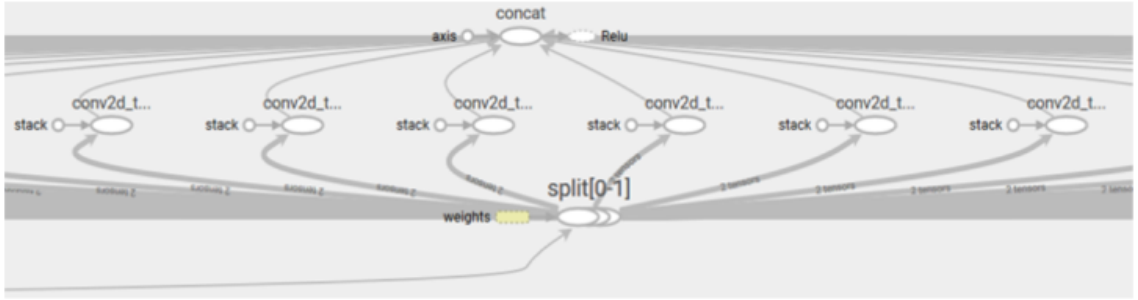
補足：Model Optimizerのモデル最適化技術

Linear Operation Fusing

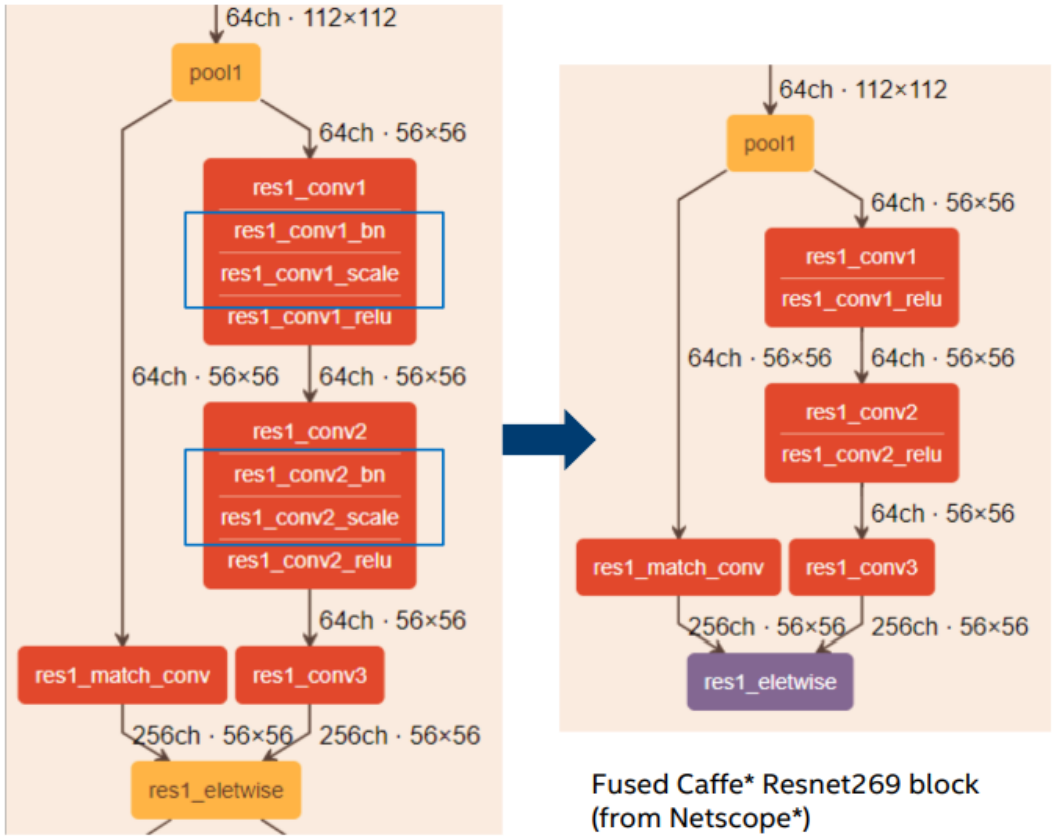
- **BatchNorm and ScaleShift decomposition:** *BN* layers decomposes to *Mul->Add->Mul->Add* sequence; ScaleShift layers decomposes to *Mul->Add* sequence.
- **Linear operations merge:** Merges sequences of Mul and Add operations to the **single** Mul->Add instance.
- **Linear operations fusion:** Fuses Mul and Add operations to Convolution or FullybConnected layers.

Grouped Convolutions Fusing

Specific optimization that applies for TensorFlow* topologies. (Xception*)



Split->Convolutions->Concat block from TensorBoard*



Caffe Resnet269 block (from Netscope)

Fused Caffe* Resnet269 block (from Netscope*)

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

ステップ. 4 OpenVINOで量子化＆推論実行

OpenVINO™

INT8

下記の順番でコマンドを実行してください

cd ~

cd model_opt_qtz

mkdir train_data/quantization

cp train_data/val/*/* train_data/quantization/

wget https://raw.githubusercontent.com/hiouchiy/IntelAI_and_Cloud/master/Azure/handson/resnet50_int8.json

wget https://raw.githubusercontent.com/hiouchiy/IntelAI_and_Cloud/master/Azure/handson/val.txt

pot –c resnet50_int8.json

以下のいずれかのコマンドを実行

- フロントエンド：python infer_script.py --input_graph "results/resnet50_int8_DefaultQuantization/YYYY-MM-DD_HH-MM-SS/optimized/resnet50_int8.xml" --num_images 300 --openvino
- 計算ノード：次のページを参照

推論性能が向上していますか？

現在OCTOPUSに搭載されいているXeonプロセッサーの世代（Skylake）では、1.3～1.4倍程度の向上が期待できます。

次のCascade Lakeでは、Deep Learning Boost（別名VNNI）という、量子化（ここではINT8化）されたモデルに特化した命令セットが搭載されているため、更なる性能向上が期待できます。

計算ノードで実行するためのスクリプト

```
#!/bin/bash

#PBS -q LECTURE

#PBS -y 326

#PBS -l elapstim_req=1:00:00

cd $PBS_O_WORKDIR

source /octfs/apl/Anaconda3/bin/activate

source activate openvino

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

python infer_script.py --input_graph "results/resnet50_int8_DefaultQuantization/YYYY-MM-DD_HH-MM-SS/optimized/resnet50_int8.xml"  --num_images 300 --openvino
```

Post-training Optimization Tool（POT）

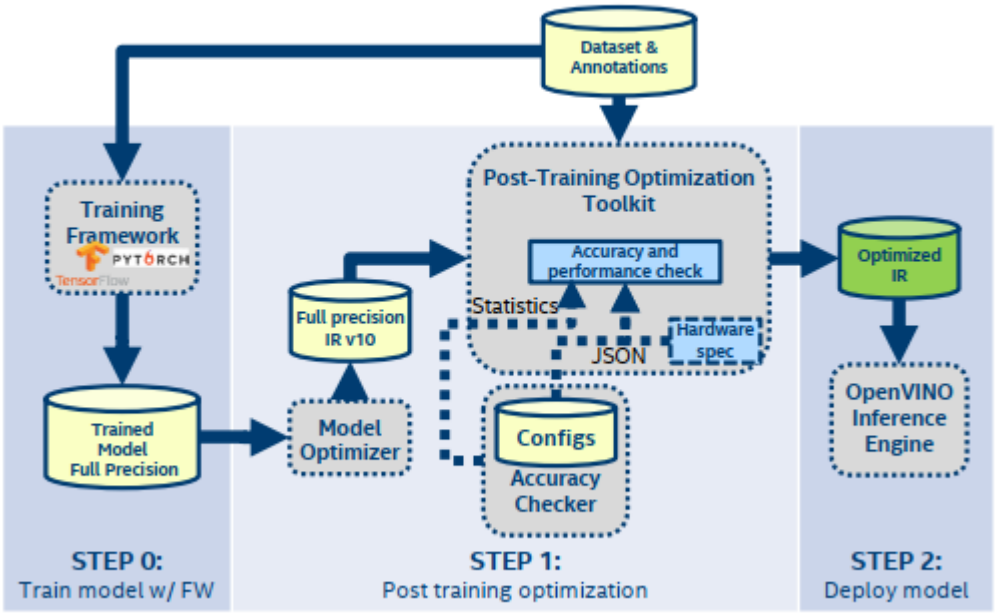
・トレーニング後のモデルに対する最適化ツール（再量子化、スパース化（まだ））

・使い方は JSONで書かれたconfigファイルを渡すとその設定に従ってINT8モデルを作ってくれます（calibration.pyはYAML）

・バリデーションデータセットにCOCOなども 指定できるようになった反面、configファイルの書式、設定項目が増えてちょっと 難しくなりました。

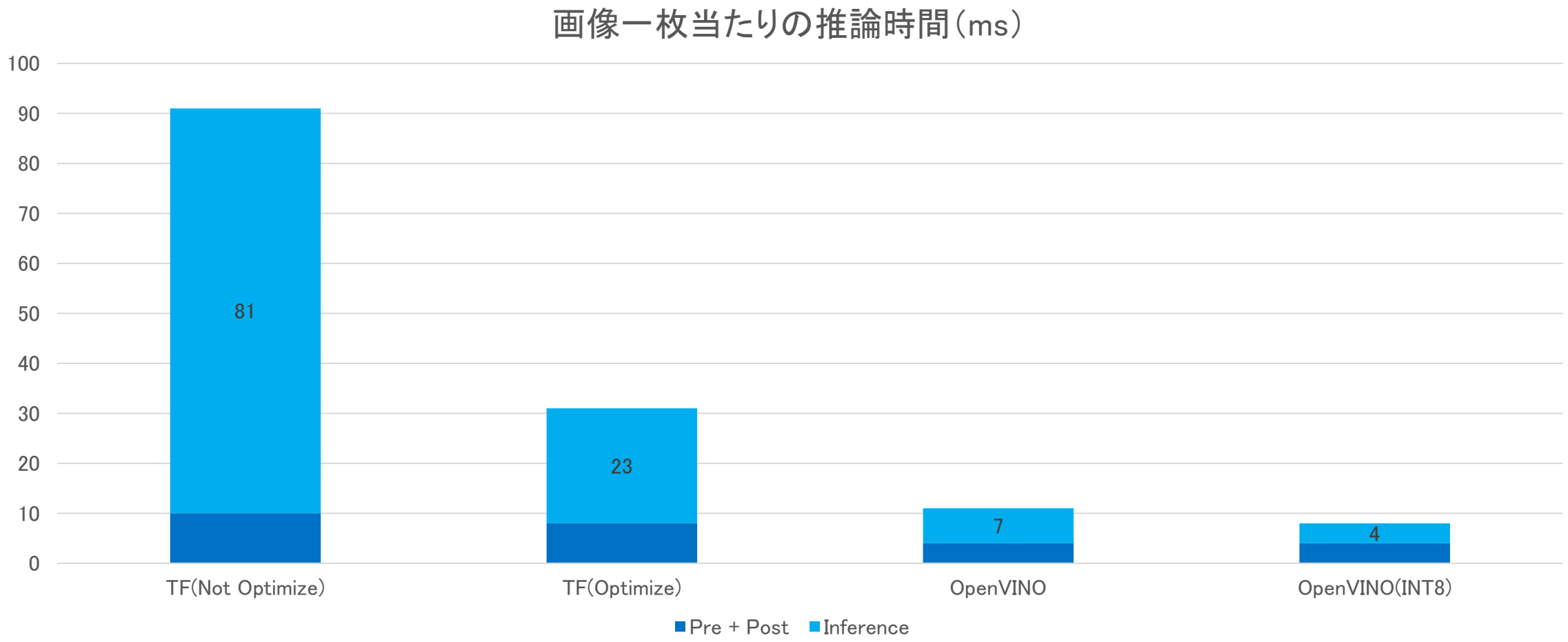
・Default algorithm(簡単)と Accuracy aware algorithm(面倒だが精度維持)が選べます

・DL workbenchからも 使えますので、そっちを使ったほうが楽ですが、DL workbenchは すべての設定が設定できるわけではないので注意。



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

性能比較（on OCTOPUS フロントエンド）



インテル® ディープラーニング・ブースト(VN

Skylake

FP32

Sign	Exponent								Mantissa																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Typical AVX-512 instruction to perform convolutions: **vfmadd231ps**

1st gen Intel® Xeon® Scalable processor with Intel® DL Boost

× 最大4倍

INT8

Sign	Mantissa							
07	06	05	04	03	02	01	00

Typical AVX-512 instructions to perform INT8 convolutions: **vpmaddubsw, vpmaddwd, vpaddd**

1st gen Intel® Xeon® Scalable processor with Intel® DL Boost

× 最大3倍

Cascade Lake

INT8

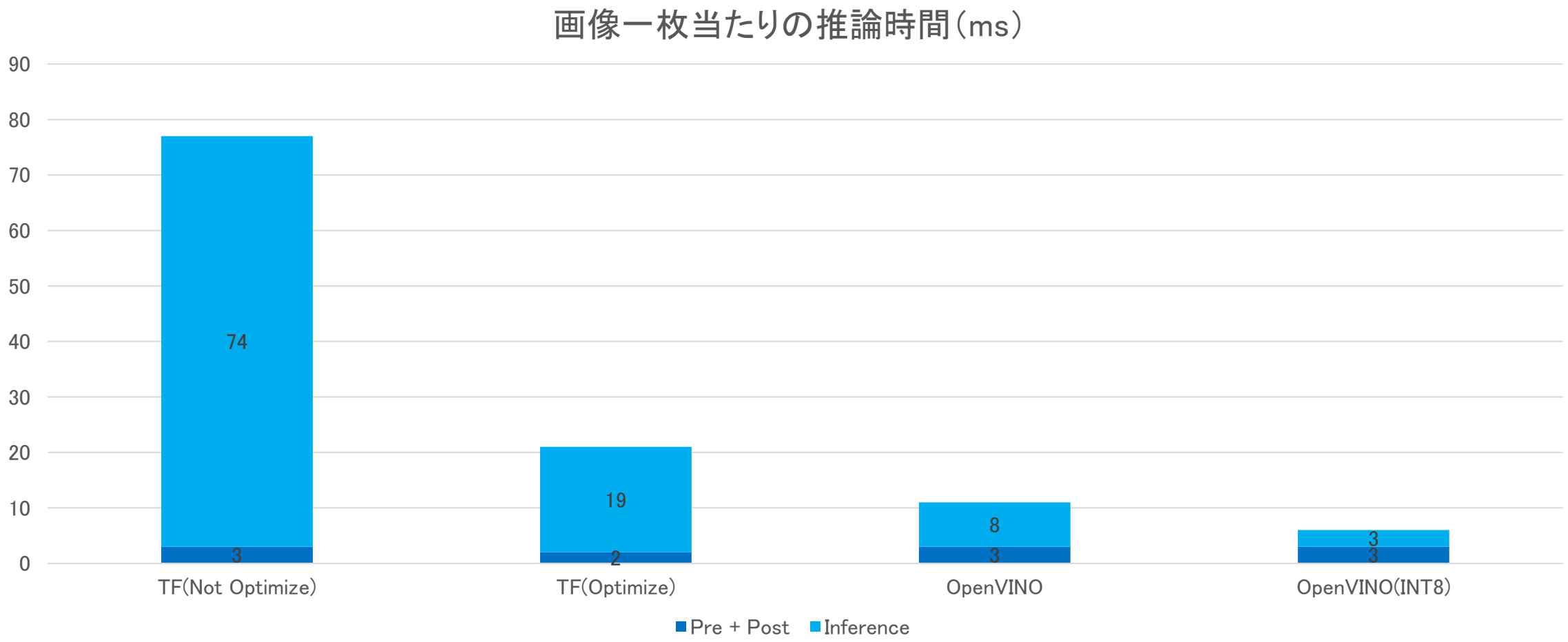
Sign	Mantissa							
07	06	05	04	03	02	01	00

Typical AVX-512 instructions to perform INT8 convolutions: **vpdpbusd**

2nd gen Intel® Xeon® Scalable processor with Intel® DL Boost

Vector Neural Network Instructions (VNNI) 対応

性能比較（Cascade Lake 6252 (24core x 1skt) ）



時間が余れば、、、

皆さんのお持ちのモデルをOpenVINOで走らせてみましょう。

おまけ：前処理（画像加工等）の高速化ヒント

libjpeg-turboの利用

https://github.com/hiouchiy/libjpeg-turbo-how-to

Optane Persistent Memoryの利用

https://tech.preferred.jp/ja/blog/mn-3-launch/

各種ガイドおよびオンデマンド動画のご紹介

深層学習 Deep Diveセッション
@Intel Data Centric Innovation Day
オンデマンド配信中

OpenVINO™ ツールキット
ハンズオン @Intel AI Park
オンデマンド配信中

TensorFlow* 量子化ガイド

- https://github.com/IntelAI/tools/releases/tag/v1.0.0

PyTorch* 量子化ガイド

- https://pytorch.org/docs/stable/quantization.html
- https://pytorch.org/blog/introduction-to-quantization-on-pytorch/
- https://pytorch.org/tutorials/advanced/static_quantization_tutorial.html
- https://pytorch.org/tutorials/advanced/dynamic_quantization_tutorial.html

OpenVINO™ ツールキット・モデル最適化ガイド（Model Optimizer の使い方）

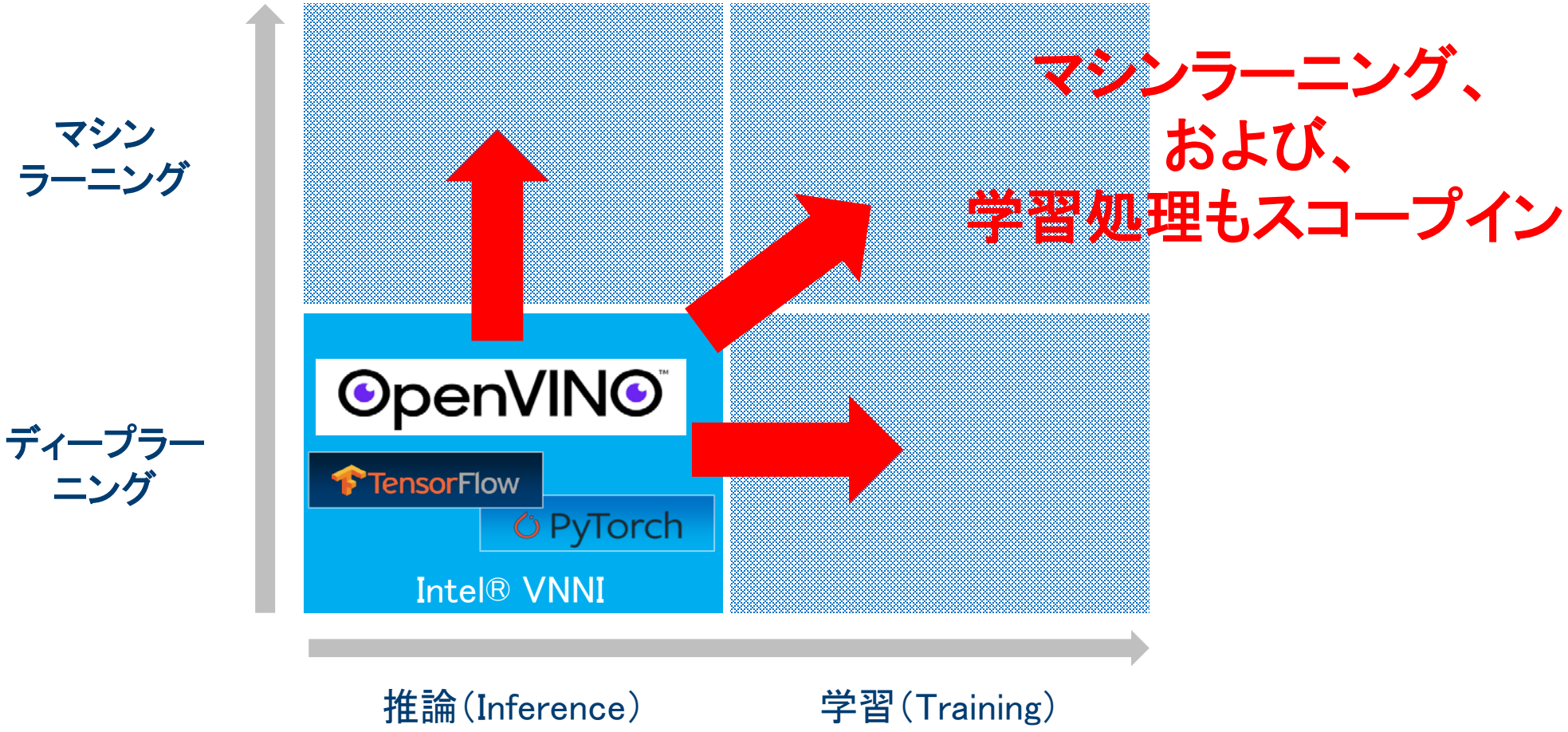
- https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html

OpenVINO™ ツールキット量子化ガイド （Post-Training Optimization Toolkit の使い方）

- https://docs.openvinotoolkit.org/latest/_README.html

ハンズオンは以上です

ここ最近のインテルAI
～より広いAIワークロードに対応～



One Intel - アナリティクス & AI ハードウェア製品

COMMUNITY

SOFTWARE

HARDWARE

Workload Breadth — AI Specific

CPU — GPU — FPGA — CUSTOM

Multi-Purpose, Foundation for Analytics & AI

Data-Parallel AI, HPC, Media & Graphics

Real-Time & Multi-Function DL Inference

Edge DL Inference

Data Center DL Inference

Data Center DL Training

STORE

インテル® 3D NAND SSD

CONNECT

Ethernet

Silicon Photonics

イベント／メディアなどのご紹介

公式Connpassグループ
https://intel.connpass.com/

公式Youtubeチャンネル
https://www.youtube.com/channel/UCymhjXNUpudGFa-vFRB-Yvw

GitHub

非公式ですが、、、面白サンプル多数公開中
https://github.com/yas-sim
https://github.com/hiouchiy

C++でプログラムを書く際の事前設定事項

下記URLに記載されております。

https://docs.openvinotoolkit.org/latest/openvino_docs_IE_DG_inference_engine_intro.html#common_workflow_for_using_the_inference_engine_api

事前準備としては、OpenVINOインストールディレクトリ内の下記スクリプトを実行ください。

source /octfs/apl/Intel/openvino2020.2.120/openvino/bin/setupvars.sh

その後、プログラムの記述およびコンパイル（ビルド）が可能となります。

ビルドに関しては下記URLもご参照ください。

https://docs.openvinotoolkit.org/latest/openvino_docs_IE_DG_Integrate_with_customer_application_new_API.html#build_your_application


intel®
experience
what's inside™