Spectrum Technology

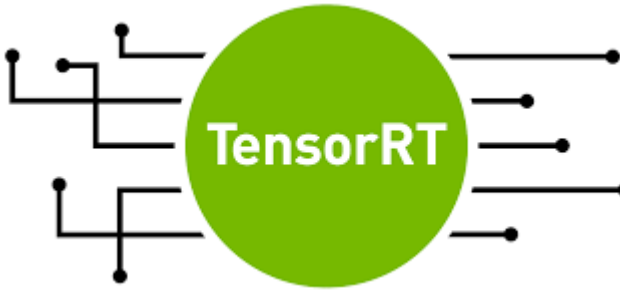
V1.0 2022/1/6

抜粋版

はじめてのTensorRT学習キット

ubuntuでTensorRTの基礎マスター、体験

学習編(ubuntu版）



スペクトラム・テクノロジー株式会社

https://spectrum-tech.co.jp

sales@spectrum-tech.co.jp

学習キット　目次

- ubuntu運用マニュアル　ページ
  - ubuntuについて　4
  - Linux基本コマンド　4
  - ubuntu基本操作　6
  - 日常運用（ウイルススキャン、更新）　7
- 学習キット　全体像　9
- ハード概要　10
- ソフト概要　11
- TensorRT
  1. 概要　13
  2. TensorRT(python)事例
     - ① EfficientDet　17
     - ② EfficientNet　23
     - ③ end_to_end_tensorflow_mnist　30
     - ④ engine_refit_mnist　31
     - ⑤ engine_refit_onnx_bidaf　32
     - ⑥ int8_caffe_mnist　33
     - ⑦ introductory_parser_samples　34
     - ⑧ network_api_pytorch_mnist　35
     - ⑨ onnx_packnet　36
     - ⑩ tensorflow_object_detection_api　37
     - ⑪ yolov3_onnx　43

抜粋版のためページと本文は一致しません

学習キット　目次

- TensorRT
  1. 概要 13
  2. TensorRT(python)事例 17
  3. TensorRT(C++)事例
     - ① Samplemnist 44
     - ② SamplemnistAPI 45
     - ③ sampleOnnxMNIST 46
     - ④ sampleUffMNIST 47
     - ⑤ sampleAlgorithmSelector 48
     - ⑥ sampleDynamicReshape 49
     - ⑦ sampleFasterRCNN 50
     - ⑧ sampleUffFasterRCNN 51
     - ⑨ sampleCharRNN 53
     - ⑩ sampleGoogleNet 54
     - ⑪ Sampleint8 55
     - ⑫ sampleINT8API 56
     - ⑬ sampleIOFormats 57
     - ⑭ sampleNMT 58
     - ⑮ sampleSSD 60
     - ⑯ sampleUffSSD 61
     - ⑰ sampleUffPluginV2Ext 62
     - ⑱ Trtexec 63
  4. Demo事例
     - ① BERT 64
     - ② HuggingFace 65
  5. Torch2trt
     - ① 画像セグメンテーション 67
     - ② 画像分類 68
     - ③ 量子化 69

Ubuntu運用マニュアル

1. Ubuntuについて

   Linuxの中でも一番シェアの高いOSです。2004年にDebian系から派生。

2. Linux基本コマンド

   ① システム関係

   - 起動：電源を入れると自動で起動します。
   - 再起動：$ reboot

     又は、左上のメニューの「ゲストを再起動
   - 終了： $ shutdown

     又は、左上のメニューの「ゲストをシャットダウン」
   - ログアウト $ exit

     ルートからログアウトします
   - 日本語／英語の入力切替：半角／全角のボタン（ESCボタンの下）

Ubuntu運用マニュアル

2. Linux基本コマンド

② ディレクトリ操作、コピー、移動、削除

masa@ubuntu:~$cd /home/masa/Documents　　ディレクトリの切り替え
masa@ubuntu :/home/masa/Documents$ ls　ファイルとディレクトリの表示（表示したら操作したいファイルを右クリックでコピペして操作します
masa@ubuntu:~$ cp ファイル名　ディレクトリ　配下のディレクトリのファイルを別のディレクトリへコピー
masa@ubuntu:~$ mv ファイル名　ディレクトリ　　配下のディレクトリのファイルを別のディレクトリへ移動
masa@ubuntu:~$ rm ファイル名　　ファイルの削除
便利な機能　　rm –help　　コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべてのコマンド共通(マイナスを2個とhelp)

③ ユーザ権限、プロセス他

masa@ubuntu :~ $ su -　　スーパーユーザ（root)に切り替え、パスワードを入力
masa@Ubuntu:~$ sudo　　ルート権限で各種コマンドを実施します。
masa@ubuntu:~$ ps　a　　現状の動いているプロセスを表示
masa@ubuntu:~$ kill　　特定のプロセスを強制終了
masa@ubuntu:~$ apt-get install pkg　　パッケージのインストールなどに使用
masa@ubuntu:~$ date　　日付、時間の設定を行います。
masa@ubuntu:~$ leafpad /etc/network/interfaces　　インタフェースに記述してい内容を変更します。Viよりも使いやすいです。

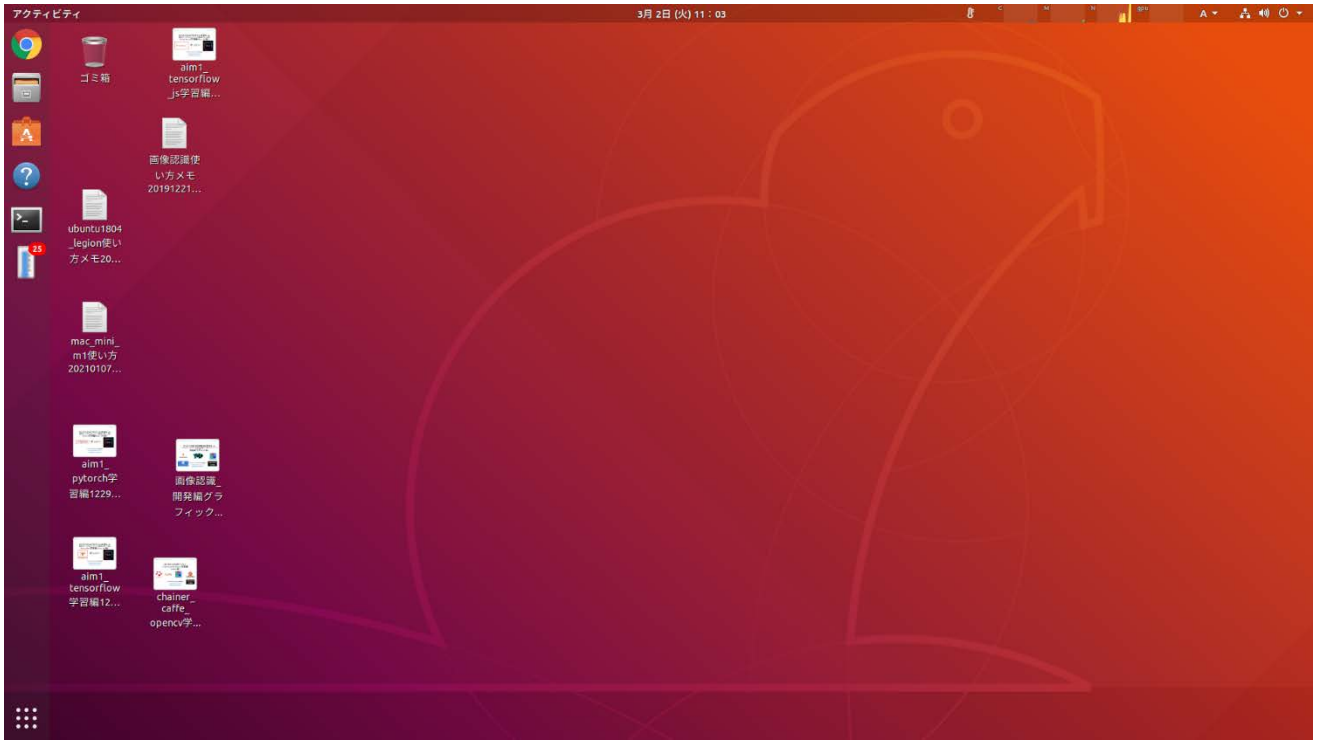
④ モジュール、usb、メモリ、HDDなどの表示

masa@ubuntu:~$ lsmod　　linuxのモジュールリスト表示
masa@ubuntu:~$ lsusb　　usbのデバイス表示
masa@ubuntu:~$ free –mt　　メモリ使用状態表示
masa@ubuntu:~$ df　　HDD（マイクロSD）の使用状態表示

Ubuntu運用マニュアル

3. 基本操作

① 表示画面と内容



主に使用するもの
・ブラウザ：Chrome
・フォルダ：Documents内に必要なファイルがあります。
・コマンド：コマンド画面を立ち上げて、python3のプログラムを動作させます。

Spectrum Technology

学習キット　全体像(ubuntu版)

ハードウエア

CPU

intel AMD

i9,i7,i5 第7世代
以降
Ryzen

＋

GPU

Nvidia RTX30, 20,
GTX16シリーズ
に対応

USBメモリ

Jetson nano版AI開発
キット（TensorRT搭載）
はリリース済（2020/6）

OS

ubuntu

AI系

TensorRT

TensorFlow

PyTorch

ONNX

画像系

OpenCV

YOLO

プログラム言語

python

グラヒック

NVIDIA
CUDA

ハードウエア概要

①必要なハードウエア仕様

本PCにインストールするハードウエアの概要です。

区分		プロダクツ	メーカ	備考
USBメモリ（ubuntu版）		64GB　USB3.0 TensorRT用プログラム一式		
お客様準備品				
PC本体	cpu	i9-7900以上（第7世代以降） i7-7700以上（同上） i5-7400以上（同上） Ryzen		マザーボードによっては、GPUが搭載できない場合があります。
	GPU	GTX16シリーズ RTX30,20シリーズ	nvidia	
	メモリ	16GB以上		
オプション				
カメラ		HD用　USBカメラ		リアルタイムで画像認識を行う場合

Spectrum Technology

ソフトウエア概要
①ソフトウエアー覧

本PCにインストールするソフトウエアの概要です。

区分	ソフト名	バージョン	備考
OS	ubuntu	18.04.3 LTS	
GPU用	cuDNN	8.1.0+cuda11.2	Nvidia用,搭載するGPUに依存
プログラム言語	python3	3.6.9	仮想化については、各自実施
	C++	cmake:3.10.2 gcc:8.4.0	
AI用プログラム	tensorRT	8.2.1	Nvidia gpu用
	tensorflow	2.5.0	Google、搭載GPU依存
	Pytorch	1.8	Facebook開発
	onnx	1.8.1	
各種モジュール	Jupyter notebook、matplotlibなど多数のpipライブラリ		

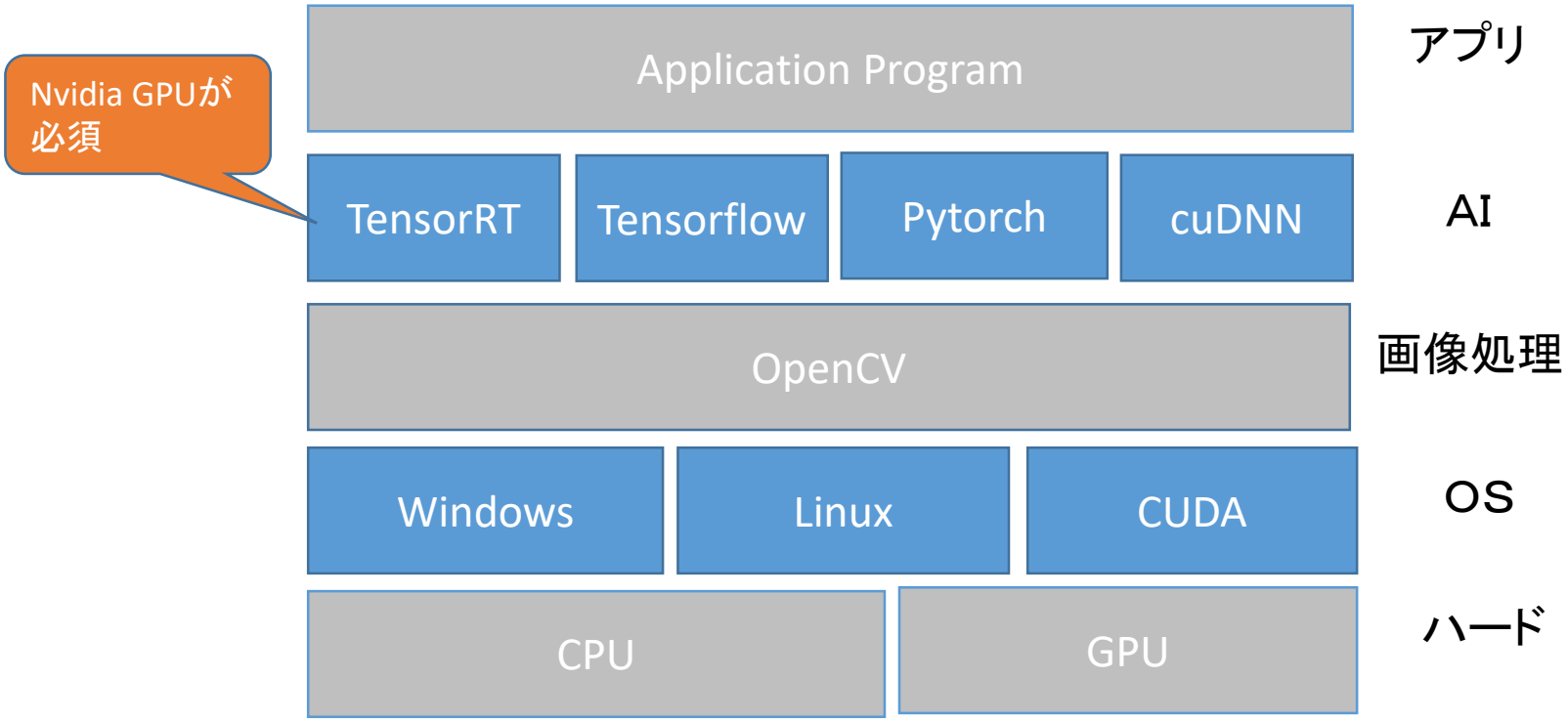
Opencv, yoloのインストールはお客様で実施してください。又は、弊社「顔認識・物体認識用AI開発キットTF2」を購入ください。

Spectrum Technology

ソフトウエア概要
②プロトコルスタック

各ソフトの位置付けです。

Python3、c++：プログラム言語



公式サイト
https://developer.nvidia.com/tensorrt

TensorRT

1. 概要

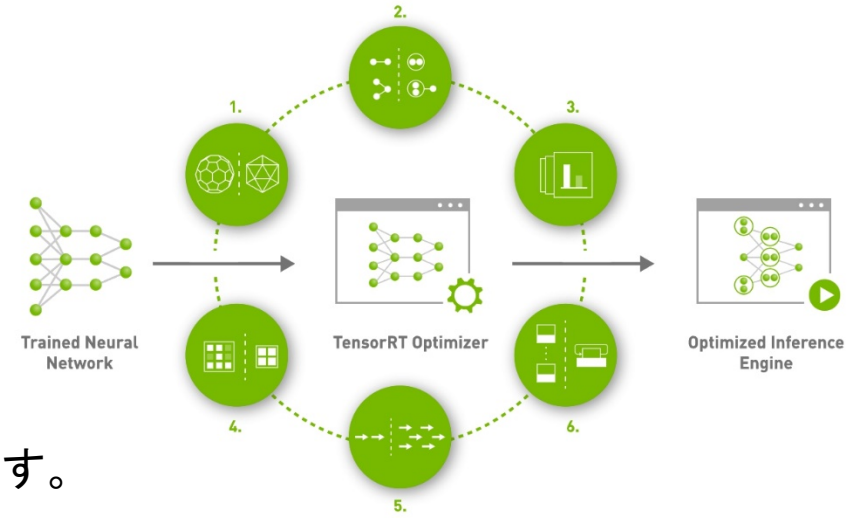
① 概要

- TensorRTベースのアプリケーションは、推論中にCPUのみのプラットフォームよりも最大40倍高速に実行されます。TensorRTを使用すると、すべての主要なフレームワークでトレーニングされたニューラルネットワークモデルを最適化し、低精度を高精度で調整し、ハイパースケールデータセンター、組み込み、または自動車製品プラットフォームに展開できます。
- NvidiaのGPUが必須になります。

② 利用方法

- TensorRTは、量子化対応トレーニングとトレーニング後の量子化を使用したINT8と、ビデオストリーミング、音声認識、推奨、不正検出、テキスト生成、自然言語処理などの深層学習推論アプリケーションの本番環境へのFP16最適化を提供します。精度の推論が低下すると、アプリケーションの遅延が大幅に減少します。これは、自律型および組み込みアプリケーションだけでなく、多くのリアルタイムサービスの要件です。
- TensorRTを使用すると、開発者は、推論展開のパフォーマンスチューニングではなく、新しいAIを利用したアプリケーションの作成に集中できます。
- Github：https://github.com/NVIDIA/TensorRT

TensorRT
1. 概要



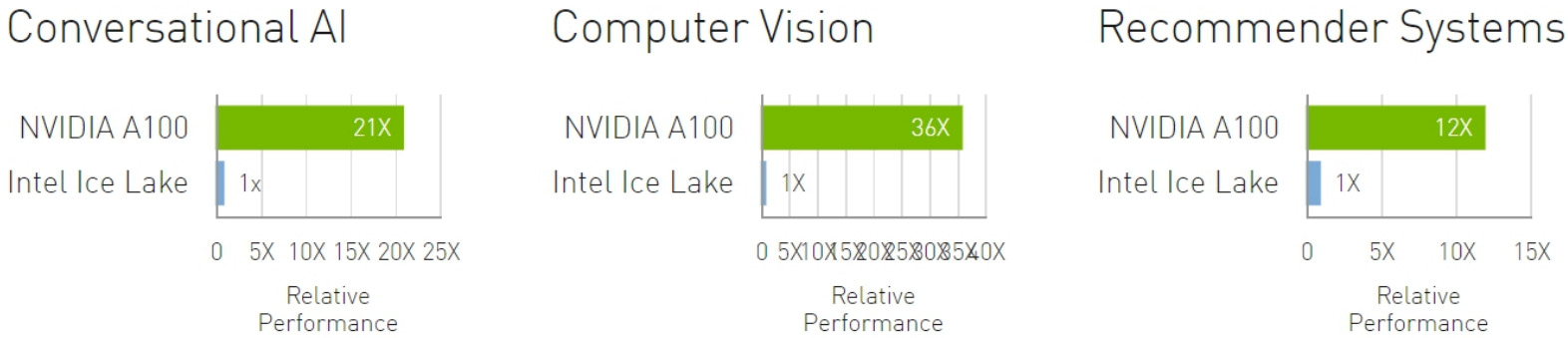
③ 概念図

- 図のとおり処理します。
- 訓練されたニューラル・ネットワーク
- TensorRT最適化：高速化が可能になります。
  1.精度の低下
     精度を維持しながらモデルを量子化することにより、FP16またはINT8でスループットを最大化します
  2.レイヤとテンソルの融合
     カーネル内のノードを融合することにより、GPUメモリと帯域幅の使用を最適化します
  3.カーネルの自動調整
     ターゲットGPUプラットフォームに基づいて最適なデータレイヤーとアルゴリズムを選択します
  4.動的テンソルメモリ
     メモリフットプリントを最小限に抑え、テンソルのメモリを効率的に再利用します
  5.マルチストリーム実行
     スケーラブルな設計を使用して、複数の入力ストリームを並行して処理します
  6.タイムフュージョン
     動的に生成されたカーネルを使用して、タイムステップにわたってリカレントニューラルネットワークを最適化します
- 最適化された推論エンジンを生成

TensorRT
1. 概要

④ パフォーマンス

- NvidiaのGPUとIntel cpuの処理比較です。10－30倍の処理能力？
- 弊社のテストのデータは別途提示しますが、pytorchとTensorRTでは、2-3倍。TensorflowとTensorRTでは、数十倍高速化されます。



⑤ お勧めの使い方

- Tensorflow, pytorchで作成したモデルを変換してtensorRTで高速化
- tensorRTの高速化モデルをjetson nanoなどのエッジデバイスで動作させる。
- 今回の学習キットは、汎用のデスクトップPC（NvidiaのRTX30，20のGPU搭載）でTensorRTの高速化モデルを作成することに使用

TensorRT
1. 概要
⑥. 項目一覧

A:難しい、B:やや難しい、C:普通
-:バグ等で動作しない

項番	事例(Githubリンク)	言語	フォーマット	難易度(本マニュアルリンク)	学習時間(分)	内容
1	sampleAlgorithmSelector	C++	Caffe	C	30	アルゴリズム選択APIの使用法
2	sampleCharRNN	C++	INetwork	C	30	レイヤーごとのRNNネットワークの構築
3	sampleDynamicReshape	C++	ONNX	C	30	TensorRTの動的形状による数字認識
4	sampleFasterRCNN	C++	Caffe	B	60	Faster R-CNNによる物体検出
5	sampleGoogleNet	C++	Caffe	C	30	TensorRTでのGoogleNetの構築と実行
6	sampleINT8	C++	Caffe	C	30	カスタムキャリブレーションを使用したINT8での推論の実行
7	sampleINT8API	C++	Caffe	C	30	INT8Precisionで推論を実行する
8	sampleMNIST	C++	Caffe	C	30	“Hello World” For TensorRT
9	sampleMNISTAPI	C++	INetwork	C	30	レイヤーごとのシンプルなMNISTネットワークの構築
10	sampleNMT	C++	INetwork	A	90	seq2seqモデルを使用したニューラル機械翻訳
11	sampleOnnxMNIST	C++	ONNX	C	30	“Hello World” For TensorRT With ONNX
12	sampleOnnxMnistCoordConvAC	C++	ONNX	-	-	カスタムプラグインを使用したCoordConvの実装
13	sampleIOFormats	C++	Caffe	C	30	Specifying TensorRT I/O Formats
14	sampleSSD	C++	Caffe	B	60	SSDによる物体検出
15	sampleUffFasterRCNN	C++	UFF	A	90	TensorFlowFasterRCNNネットワークを使用した物体検出
16	sampleUffMNIST	C++	UFF	C	30	TensorFlowモデルをインポートして推論を実行する
17	sampleUffMaskRCNN	C++	UFF	-	-	MasKR-CNNネットワークによる物体検出とインスタンスセグメンテーション
18	sampleUffPluginV2Ext	C++	UFF	C	30	INT8 I/Oをサポートするカスタムレイヤーをネットワークに追加する
19	sampleUffSSD	C++	UFF	C	30	TensorFlowSSDネットワークを使用した物体検出
20	trtexec	C++	All	B	90	TensorRT Command-Line Wrapper: trtexec
21	efficientdet	Python	ONNX	A	120	TensorRTを使用したEfficientDet物体検出
22	efficientnet	Python	ONNX	A	120	TensorRTを使用したEfficientNetV1およびV2分類
23	end_to_end_tensorflow_mnist	Python	UFF	C	30	“Hello World” For TensorRT Using TensorFlow
24	engine_refit_mnist	Python	INetwork	C	30	TensorRTエンジンの取り付け
25	int8_caffe_mnist	Python	Caffe	C	30	INT8 Calibration
26	introductory_parser_samples	Python	Any	C	30	TensorRTパーサーを使用したモデルのインポートの概要
27	network_api_pytorch_mnist	Python	INetwork	C	30	“Hello World” For TensorRT
28	onnx_packnet	Python	ONNX	B	60	カスタムレイヤーを使用したONNXモデルのTensorRT推論
29	tensorflow_object_detection_api	Python	ONNX	A	120	TensorRTのTensorFlow物体検出APIモデル
30	uff_custom_plugin	Python	INetwork	-	-	TensorRTのTensorFlowネットワークにカスタムレイヤーを追加する
31	uff_ssd	Python	UFF	-	-	SSDによる物体検出
32	yolov3_onnx	Python	ONNX	B	90	TensorRTONNXバックエンドでYOLOv3を使用した物体検出

Spectrum Technology

TensorRT
2.TensorRT(python)事例

入力コマンド
```
$ pip3 install onnxruntime==1.8.1 tf2onnx==1.8.1
$ cd /usr/src/tensorrt/
$ git clone https://github.com/google/automl
cd /usr/src/tensorrt/data/efficientdet
$ cd /usr/src/tensorrt/automl/efficientdet
$ python3 model_inspect.py ¥
   --runmode saved_model ¥
   --model_name efficientdet-d0 ¥
   --ckpt_path
/usr/src/tensorrt/data/efficientdet/efficientdet-d0 ¥
   --saved_model_dir
/usr/src/tensorrt/data/efficientdet/saved_model
```

①　EfficientDet

- https://github.com/NVIDIA/TensorRT/tree/main/samples/python/efficientdet
- TensorRTを使用した物体検出
- 使用モデルGoogle EfficientDet
- 事前準備
  - 追加モジュール

  $ pip3 install onnxruntime==1.8.1　　tf2onnx==1.8.1
- モデル変換　　a)tensroflow—b)onnx—c)tensorRTで一度onnxを経由します。
  - Onnxは、open neural network exchangeの略でkeras, pytorchなどとの互換性と高速化を図り、tensorRTを使う上では必須になります。

a) Tensorflow saved model

- AutoMLのダウンロード,他にTFOD　zooのモデル例が上記githubに掲載（今回は省略

$ cd /usr/src/tensorrt/
$ git clone https://github.com/google/automl
$　cd /usr/src/tensorrt/data/efficientdet内に
https://github.com/google/automl/tree/master/efficientdet
　　上記URLの2. Pretrained EfficientDet CheckpointsのEfficientDet-D0のckptをダウンロードして解凍
上記のダウンロードしたモデルからSave_modelを生成

dataフォルダをusbから上書きコピペした場合は既に存在

```
$ cd /usr/src/tensorrt/automl/efficientdet
$ python3 model_inspect.py ¥
   --runmode saved_model ¥
   --model_name efficientdet-d0 ¥
   --ckpt_path /usr/src/tensorrt/data/efficientdet/efficientdet-d0 ¥
   --saved_model_dir /usr/src/tensorrt/data/efficientdet/saved_model
```

TensorRT
2.TensorRT(python)事例

入力コマンド
$ cd /usr/src/tensorrt/samples/python/efficientdet
$ python3 infer.py ¥
--engine engine.trt ¥
--input construction1.jpg ¥
--output output

① EfficientDet

- https://github.com/NVIDIA/TensorRT/tree/main/samples/python/efficientdet
- TensorRTを使用した物体検出
- 使用モデルGoogle EfficientDet
- tensorRTによる画像検出

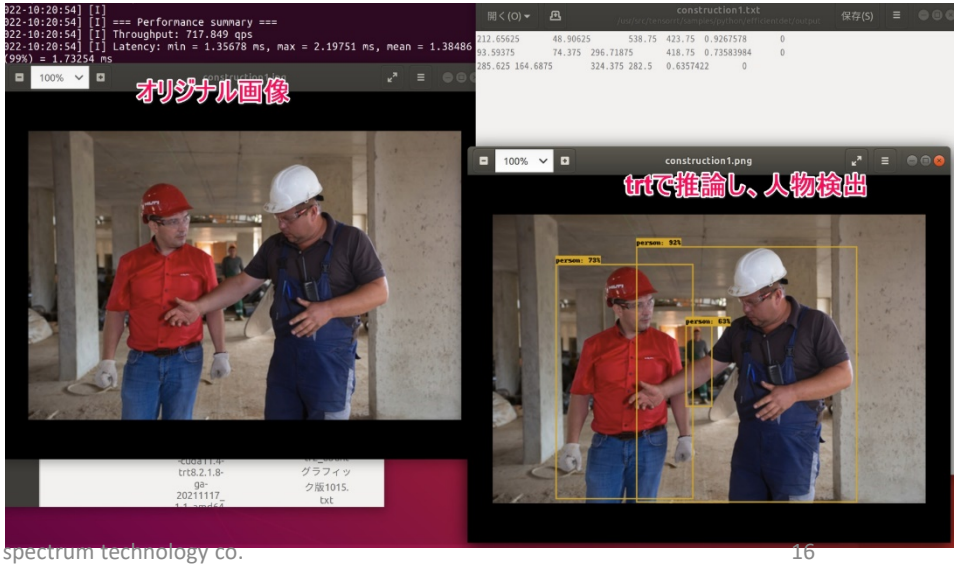
tensorrtエンジン：engine.trtを使って画像検出します。TFとTRTの比較がでます。

$ cd /usr/src/tensorrt/samples/python/efficientdet

construction1.jpgは、/usr/src/tensorrt/data/efficientdet/内にありますのでコピペで使用してください。

$ python3 infer.py ¥
--engine engine.trt ¥
--input construction1.jpg ¥
--output output

いろいろな画像でテストしてください。
最初のtensorflowのモデルで検出するものが変わります。



TensorRT
2.TensorRT(python)事例

① EfficientDet

- https://github.com/NVIDIA/TensorRT/tree/main/samples/python/efficientdet
- TensorRTを使用した物体検出
- 使用モデルGoogle EfficientDet
- tensorRTによる画像検出の比較

tensorrtエンジン：engine.trtを使って画像検出します。TFとTRTの比較がでます。

$ cd /usr/src/tensorrt/samples/python/efficientdet

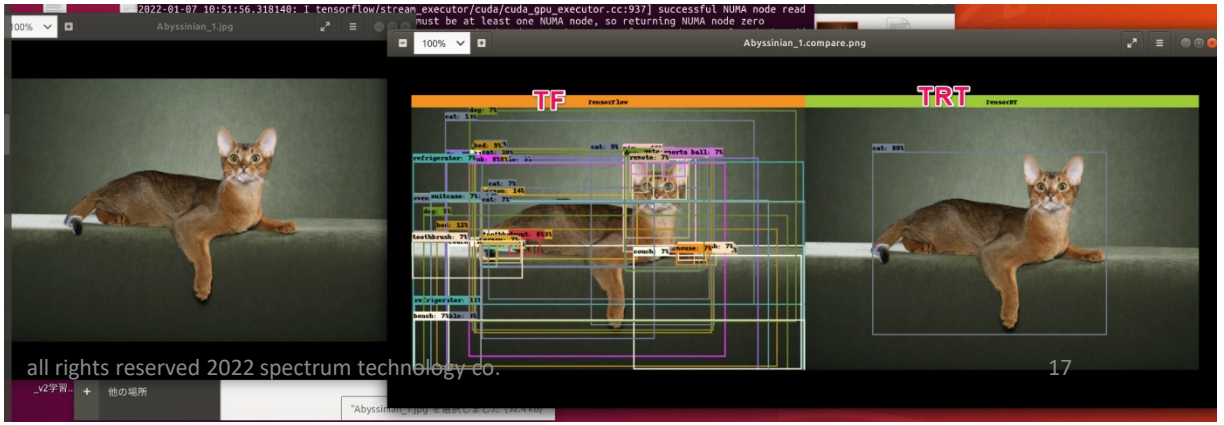
Abyssinian_1.jpgは、/usr/src/tensorrt/data/efficientdet/内にありますのでコピペで使用してください。

```
$ python3 compare_tf.py ¥
    --engine engine.trt ¥
    --saved_model /usr/src/tensorrt/data/ssd_mobilenet_v2_320x320_coco17_tpu-8/saved_model ¥
    --input  Abyssinian_1.jpg ¥
    --output output1
```

入力コマンド

```
$ cd /usr/src/tensorrt/samples/python/efficientdet
$ python3 compare_tf.py ¥
    --engine engine.trt ¥
    --saved_model /usr/src/tensorrt/data/ssd_mobilenet_v2_320x320_coco17_tpu-8/saved_model ¥
    --input Abyssinian_1.jpg ¥
    --output output1
```

高速化がわかりやすい



TensorRT 2.TensorRT(python)事例

② EfficientNet

- https://github.com/NVIDIA/TensorRT/tree/main/samples/python/efficientnet
- TensorRTを使用した分類
- 使用モデルEfficientNet V2
- tensorRT推論の検証

tensorrtエンジン：engine.trtを画像分類を行います。 ImageNet の ILSVRC2012 のデータを使用

出力結果の検証。

```
$ cd /usr/src/tensorrt/samples/python/efficientnet
$ python3 eval_gt.py ¥
   --engine /usr/src/tensorrt/data/effnetv2/effnetv2-s/checkpoint/engine.trt ¥
   --annotations /usr/src/tensorrt/data/effnetv2/val.txt¥
   --input /usr/src/tensorrt/data/effnetv2/imagenet/val¥
   --preprocessor V2
```

結果

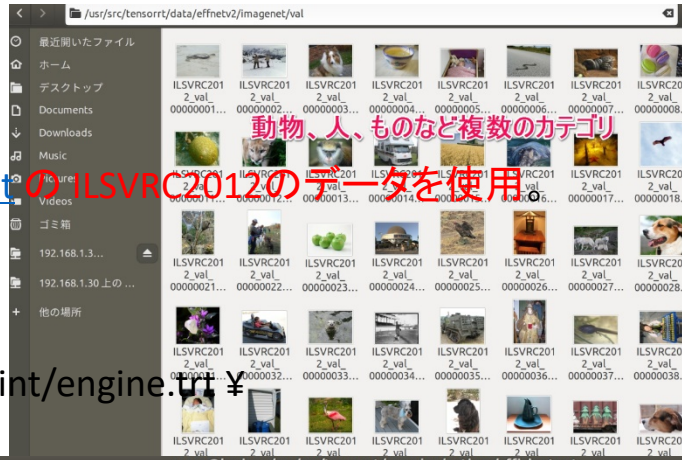
Processing 5000 / 5000 : Top-1 83.5% , Top-5: 96.9%

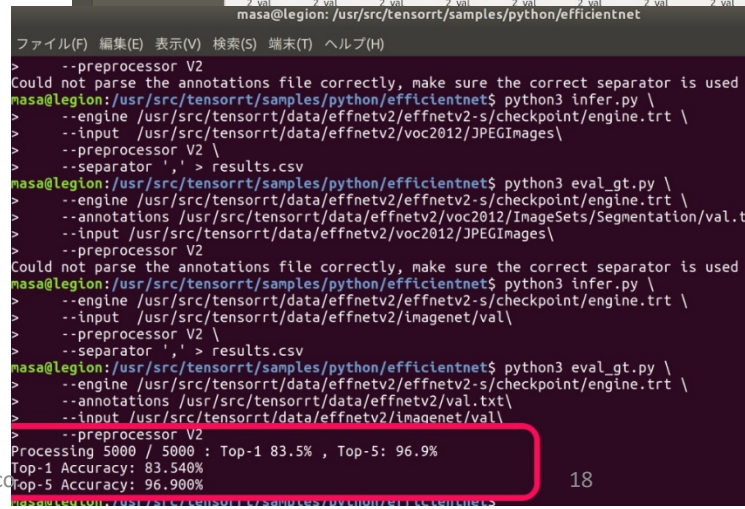
Top-1 Accuracy: 83.540%

Top-5 Accuracy: 96.900%

入力コマンド

```
$ cd /usr/src/tensorrt/samples/python/efficientnet
$ python3 eval_gt.py ¥
   --engine /usr/src/tensorrt/data/effnetv2/effnetv2-s/checkpoint/engine.trt ¥
   --annotations /usr/src/tensorrt/data/effnetv2/val.txt¥
   --input /usr/src/tensorrt/data/effnetv2/imagenet/val¥
   --preprocessor V2
```





TensorRT 2.TensorRT(python)事例

入力コマンド
$ cd /usr/src/tensorrt/samples/python/introductory_parser_samples
$ python3 onnx_resnet50.py -d /usr/src/tensorrt/data/effnetv2/effnetv2-s/checkpoint/model.onnx

⑦ introductory_parser_samples

https://github.com/NVIDIA/TensorRT/tree/main/samples/python/introductory_parser_samples

the UFF, Caffe and ONNXモデルを使って、TensorRT実行

- 使用モデルresnet50
- tensorRT推論実行

  $ cd /usr/src/tensorrt/samples/python/introductory_parser_samples

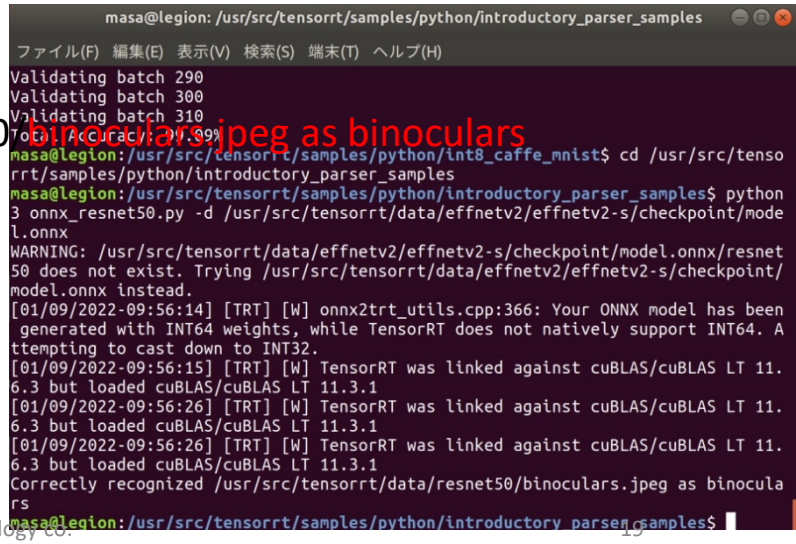
  /usr/src/tensorrt/data/effnetv2/を使用

  $ python3 onnx_resnet50.py -d /usr/src/tensorrt/data/effnetv2/effnetv2-s/checkpoint/model.onnx

  結果

  Correctly recognized /usr/src/tensorrt/data/resnet50/binoculars.jpeg as binoculars

他にuff,caffeのモデルもあります。



入力コマンド
$ cd /usr/src/tensorrt/samples/python/network_api_pytorch_mnist
$ python3 sample.py

TensorRT 2.TensorRT(python)事例

⑧ network_api_pytorch_mnist

https://github.com/NVIDIA/TensorRT/tree/main/samples/python/network_api_pytorch_mnist

Pytorchのmnistを使って、TensorRT実行

- 使用モデルmnist
- tensorRT推論実行

$ cd /usr/src/tensorrt/samples/python/network_api_pytorch_mnist

$ python3 sample.py

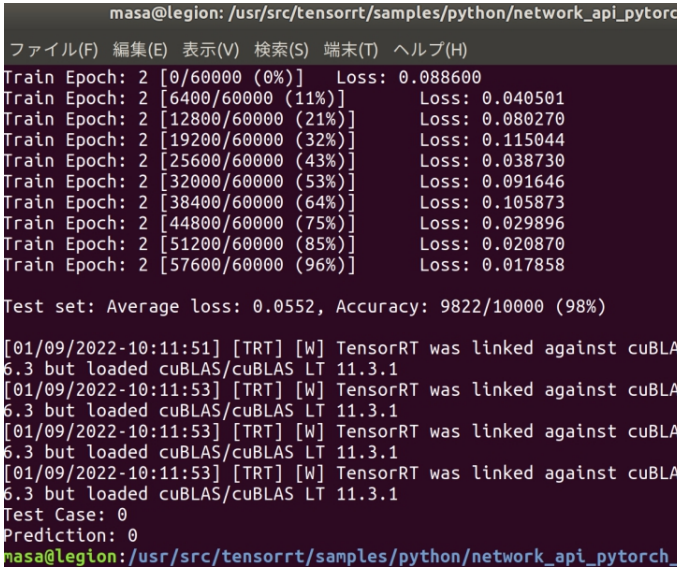
結果

Test set: Average loss: 0.0552, Accuracy: 9822/10000 (98%)

Test Case: 0

Prediction: 0

学習：6万回x2回実施して、98%の正解率：かなり高速で学習
テストデータも0で正解



Spectrum Technology

TensorRT
2.TensorRT(python)事例

入力コマンド
```
$ cd /usr/src/tensorrt/samples/python/tensorflow_object_detection_api
$ python3 compare_tf.py ¥
   --engine engine.trt ¥
   --saved_model /usr/src/tensorrt/data/export/saved_model ¥
   --input /usr/src/tensorrt/data/coco/val2017 ¥
   --annotations
/usr/src/tensorrt/data/coco/annotations/instances_val2017.json ¥
   --output output1 ¥
   --preprocessor fixed_shape_resizer ¥
   --labels labels_coco.txt
```

⑩ tensorflow_object_detection_api

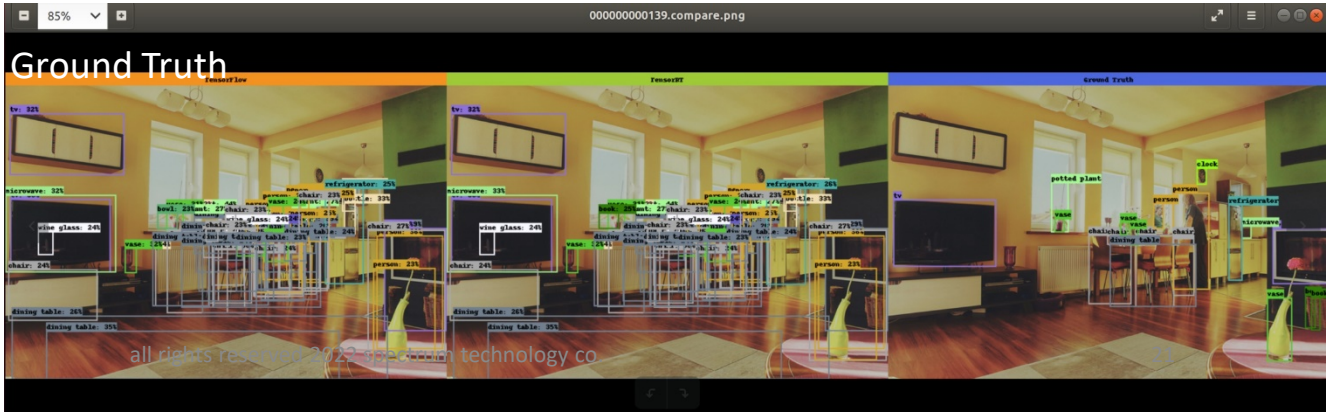
https://github.com/NVIDIA/TensorRT/tree/main/samples/python/tensorflow_object_detection_api

Tensorflowのobject detecition apiを使って、TensorRT実行

- 使用モデルssd_mobilenet_v2
- TF vs TRT比較
  - 全体の流れは、①efficientDetと同じ

```
$ cd /usr/src/tensorrt/samples/python/tensorflow_object_detection_api
$ python3 compare_tf.py ¥
   --engine engine.trt ¥
   --saved_model /usr/src/tensorrt/data/export/saved_model ¥
   --input /usr/src/tensorrt/data/coco/val2017 ¥
   --annotations /usr/src/tensorrt/data/coco/annotations/instances_val2017.json ¥
   --output output1 ¥
   --preprocessor fixed_shape_resizer ¥
   --labels labels_coco.txt
```

結果
100枚の画像でTF、TRT, Ground Truthの比較が見れます。



Spectrum Technology

TensorRT 2.TensorRT(python)事例

入力コマンド
$ cd /usr/src/tensorrt/samples/python/yolov3_onnx
$ python3 yolov3_to_onnx.py -d /usr/src/tensorrt/data/yolov3_onnx
$ python3 onnx_to_tensorrt.py -d /usr/src/tensorrt/data/yolov3_onnx

⑪ yolov3_onnx

https://github.com/NVIDIA/TensorRT/tree/main/samples/python/yolov3_onnx

Yolov3からonnxに変換し、TensorRT実行

- 使用モデルcoco dataset
- Yolov3_trt実行
  - 全体の流れは、①efficientDetと同じ

  $ cd /usr/src/tensorrt/samples/python/yolov3_onnx
  - Yolov3データ

  /usr/src/tensorrt/data/yolov3_onnx　配下にyolov3.cfg, yolov3.weihts, sample写真が入ってます。
  - Yolov3からonnx変換

  $ python3 yolov3_to_onnx.py -d /usr/src/tensorrt/data/yolov3_onnx

  結果

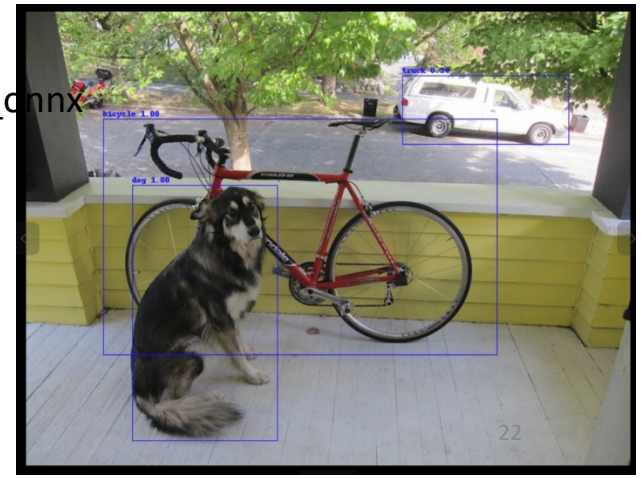
  Yolov3.onnx, Yolov3.trtが出力されます。
  - Trt実行

  $ python3 onnx_to_tensorrt.py -d /usr/src/tensorrt/data/yolov3_onnx

  結果

  dog_bboxes.pngが出力

Opencv, yoloのインストールはお客様で実施してください。又は、弊社「顔認識・物体認識用AI開発キットTF2」を購入ください。



TensorRT
3.TensorRT(c++)事例

入力コマンド
$ cd /home/masa/Documents/TensorRT/build/samples/sampleMNIST/out
$ ./sample_mnist --datadir=/usr/src/tensorrt/data/mnist --fp1

① Samplemnist

https://github.com/NVIDIA/TensorRT/tree/main/samples/sampleMNIST

“Hello World” For TensorRT

- 使用モデルmnist
- データ準備
  - usbのdataを/usr/src/tensorrt/dataに上書きするとmnistのデータがあり
- sample実行

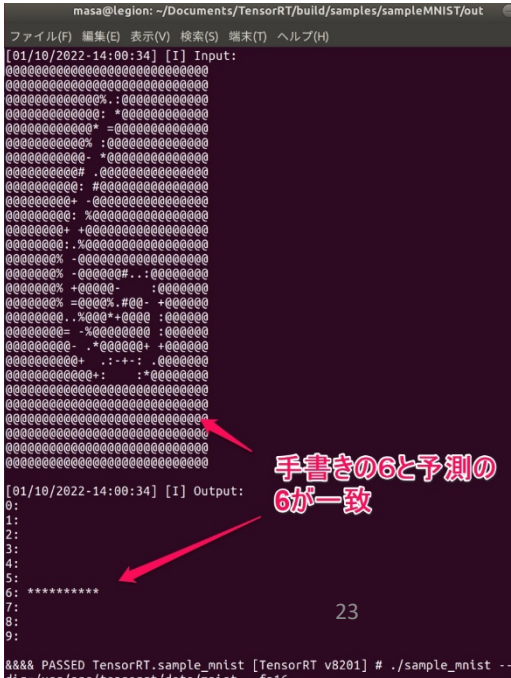
  $ cd /home/masa/Documents/TensorRT/build/samples/sampleMNIST/out

  $ ./sample_mnist --datadir=/usr/src/tensorrt/data/mnist --fp16

  結果

  手書きの6（入力）と予想の6が一致

設定編でc++を実行すると、各sampleフォルダに実行ファイルがoutフォルダに自動生成



Spectrum Technology

TensorRT
3.TensorRT(c++)事例

入力コマンド
$ cd /home/masa/Documents/TensorRT/build/samples/sampleUffFasterRCNN/out
$ ./sample_uff_fasterRCNN --datadir /home/masa/Documents/TensorRT/samples/sampleUffFasterRCNN/uff_faster_rcnn -W 480 -H 272 -I 000456.ppm

⑧ sampleUffFasterRCNN

https://github.com/NVIDIA/TensorRT/tree/main/samples/sampleUffFasterRCNN

TensorflowのFaster RCNNを変換してtensorRTで実行

- 使用モデルFaster RCNN
- データ準備
  - usbのdataを/usr/src/tensorrt/dataに上書きするとfaster-rcnnのデータがあります。
- tensorRT推論実行

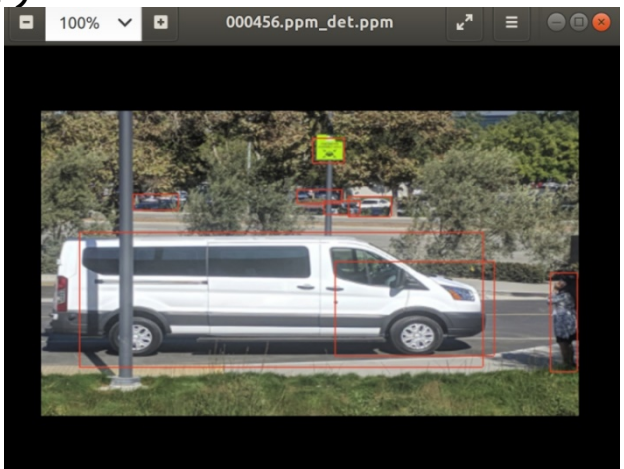
  $ cd /home/masa/Documents/TensorRT/build/samples/sampleUffFasterRCNN/out

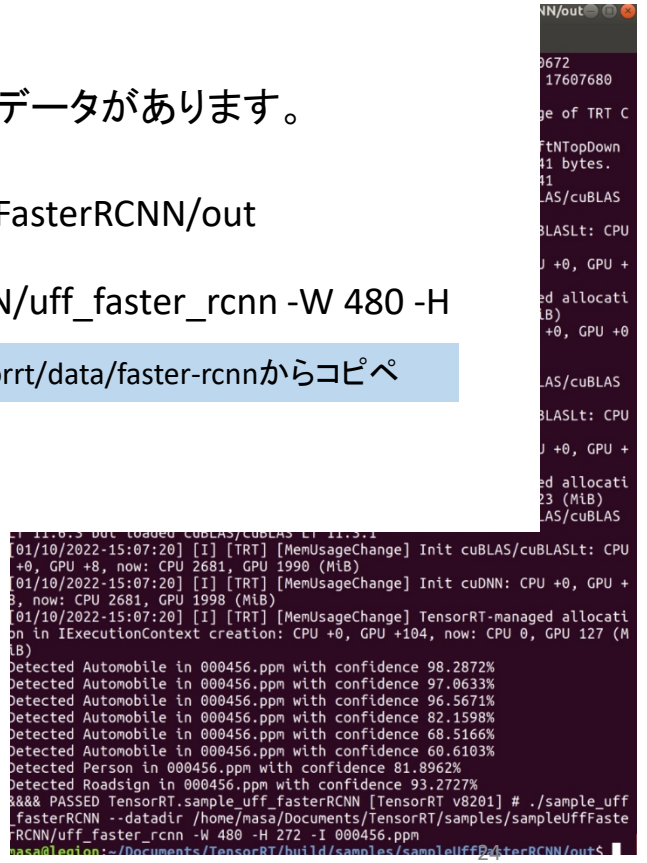
  $ ./sample_uff_fasterRCNN --datadir /home/masa/Documents/TensorRT/samples/sampleUffFasterRCNN/uff_faster_rcnn -W 480 -H 272 -I 000456.ppm

  結果

  推論結果が出力されます

00456.ppmの写真は、/usr/src/tensorrt/data/faster-rcnnからコピペ





Spectrum Technology

TensorRT
3.TensorRT(c++)事例

入力コマンド
$ cd /home/masa/Documents/TensorRT/build/samples/sampleINT8API/out
$ ./sample_int8_api --model=/usr/src/tensorrt/data/resnet50/ResNet50.onnx --image=/usr/src/tensorrt/data/int8_api/airliner.ppm --reference=/usr/src/tensorrt/data/int8_api/reference_labels.txt --ranges=/usr/src/tensorrt/data/int8_api/resnet50_per_tensor_dynamic_range.txt

⑫ sampleINT8API

https://github.com/NVIDIA/TensorRT/tree/main/samples/sampleINT8API

int8補正なしでint8推論、tensorRT実行

- 使用モデルresnet50
- データ準備
  - Usbのdataを/usr/src/tensorrt/dataに上書きするとresnet50のデータがあります。
- TensorRT実行

  $ cd /home/masa/Documents/TensorRT/build/samples/sampleINT8API/out

  $ ./sample_int8_api --model=/usr/src/tensorrt/data/resnet50/ResNet50.onnx --image=/usr/src/tensorrt/data/int8_api/airliner.ppm --reference=/usr/src/tensorrt/data/int8_api/reference_labels.txt --ranges=/usr/src/tensorrt/data/int8_api/resnet50_per_tensor_dynamic_range.txt

  結果

  [01/11/2022-07:49:04] [I] SampleINT8API result: Detected:
  [01/11/2022-07:49:04] [I] [1] airliner
  [01/11/2022-07:49:04] [I] [2] warplane
  [01/11/2022-07:49:04] [I] [3] projectile
  [01/11/2022-07:49:04] [I] [4] space shuttle
  [01/11/2022-07:49:04] [I] [5] wing
  &&&& PASSED TensorRT.sample_int8_api [TensorRT v8201]

Spectrum Technology

入力コマンド
$ cd
/home/masa/Documents/TensorRT/build/samples/sampleSSD/out
$ ./sample_ssd --datadir /usr/src/tensorrt/data/ssd --fp16

TensorRT
3.TensorRT(c++)事例

⑮ sampleSSD

https://github.com/NVIDIA/TensorRT/tree/main/samples/sampleSSD

Pascal vocを使って、tensorRT実行

- 使用モデルvoc0712_ssd_300x300
- データ準備
  - Usbのdataを/usr/src/tensorrt/dataに上書きするとssdのデータがあります。
- TensorRT実行

  $ cd /home/masa/Documents/TensorRT/build/samples/sampleSSD/out

  $ ./sample_ssd --datadir /usr/src/tensorrt/data/ssd --fp16

  結果

  Image name:/usr/src/tensorrt/data/ssd/bus.ppm, Label: car, confidence: 96.0449 xmin: 4.02832 ymin: 117.48 xmax: 244.043 ymax: 241.699

  &&&& PASSED TensorRT.sample_ssd [TensorRT v8201]

Spectrum Technology

TensorRT 5.torch2trt

入力コマンド
$ cd
/home/masa/Documents/torch2trt/examples/image_segmentation
$ jupyter notebook

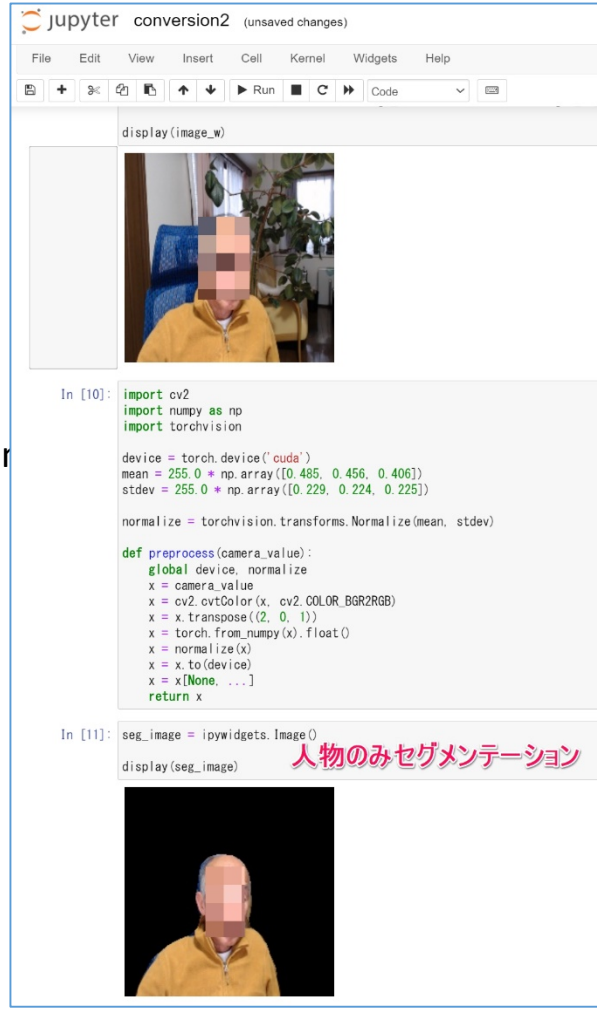
① 画像セグメンテーション

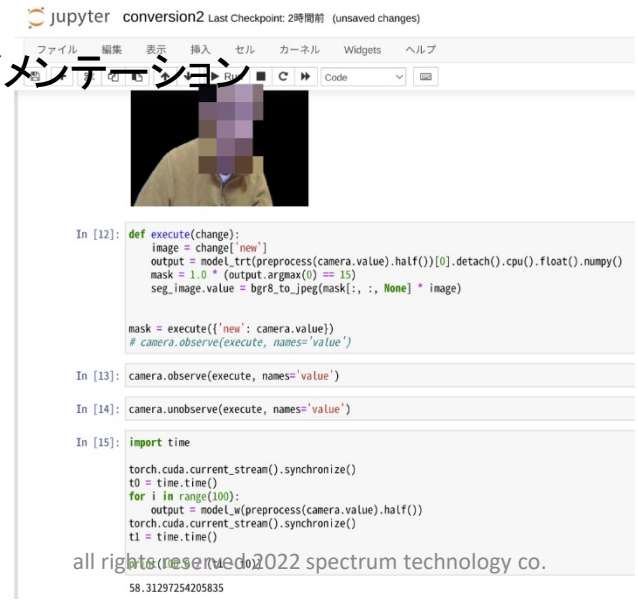
https://github.com/NVIDIA-AI-IOT/torch2trt

Pytorchのモデルをtensorrtに変換しで高速処理

- usbカメラを使って画像のセグメンテーションを行います
- TensorRT実行

  $ cd /home/masa/Documents/torch2trt/examples/image_segmentation
  $ jupyter notebook
  conversion2.ipynbを起動
  結果
  背景が消えて、人物がセグメンテーション





ライブ画像がでない場合は、
$ pip3 install jupyterlab

Spectrum Technology

TensorRT 5.torch2trt

入力コマンド
$ cd
/home/masa/Documents/torch2trt/examples/image_classification
$ jupyter notebook

② 画像分類

https://github.com/NVIDIA-AI-IOT/torch2trt

Pytorchのモデルをtensorrtに変換しで高速処理

- usbカメラを使って画像の分類を行います
- TensorRT実行

  $ cd /home/masa/Documents/torch2trt/examples/image_classification

  $ jupyter notebook

  live_demo2.ipynbを起動

  結果

  検出したものをリアルタイムで、文字で表示。

  他のプログラムでopencvで画像上に重ねて表示可能