インテルの GPU を使用して YOLOv8 で 1000fps 越えを達成するには

この記事は、Medium に公開されている「How to get YOLOv8 Over 1000 fps with Intel GPUs?」の日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。

著者: Raymond Lo インテル コーポレーション AI ソフトウェア・エバンジェリスト

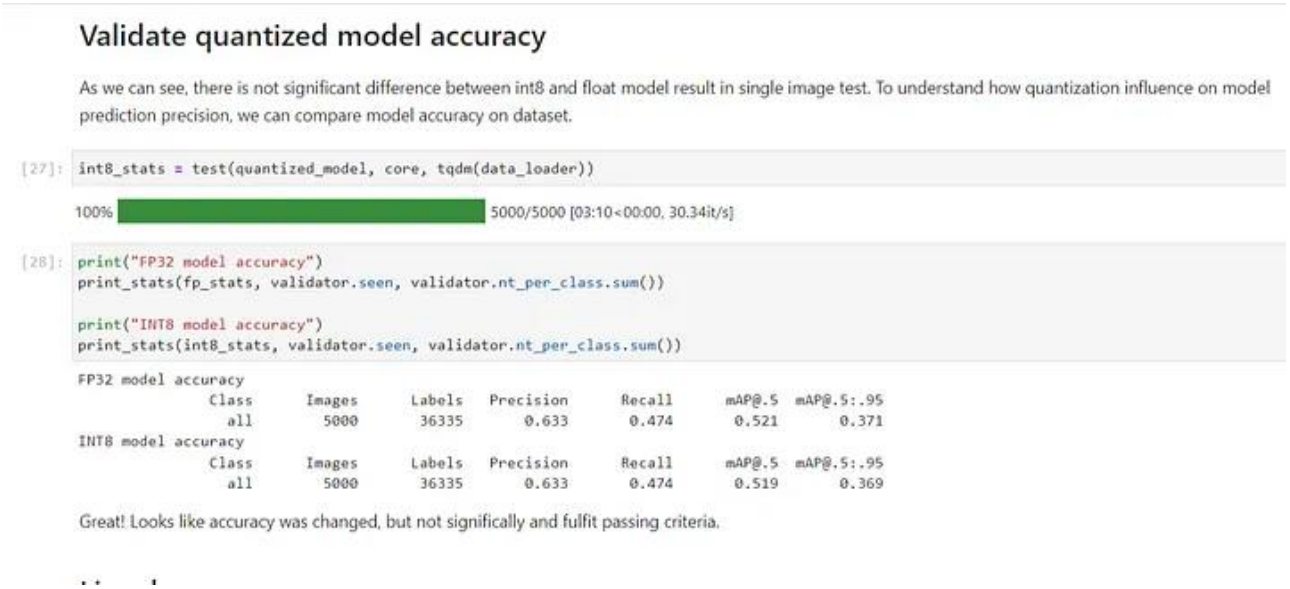
OpenVINO™ とインテル® Arc™ A770m グラフィックスがあれば YOLOv8 で 1000fps 越えを達成できます!

GPU で AI 推論を実行することは新しいトピックではありません。最近では、AI のトレーニングと推論に GPU を使用するアプリケーションも多くなりました。では、新しいインテル® Arc™ グラフィックスを使用して同じことができるでしょうか? どうすれば良いのでしょうか?

答えは OpenVINO™ を使用することです。私が (インテルに入社する前に) 初めて AI に取り組んだとき、TensorFlow* を使用して YOLO のような物体検出を実行しようとしました。そのときに気になったのは、最適化やハードウェア・アクセラレーション機能が備わっていない CPU のパフォーマンスでした。動作は非常に遅く、エンジンが 1 つのコアで動作していることがよく分かりました (非常に遅いわけです)。その当時、優れたリアルタイム・パフォーマンスを得るには高価なグラフィックス・カードが必要でした。現在では、YOLOv8 と OpenVINO™ (英語) により、状況は大きく変わりました。CPU、iGPU、dGPU を、同じコードベースでシームレスに連携できます。これは素晴らしいことです。

この記事で覚えてほしいことは、Neural Network Compression Framework (ニューラル・ネットワーク圧縮フレームワーク、NNCF) (英語) でモデルを INT8 向けに圧縮および最適化して、VNNI (英語)、インテル® AMX、インテル® XMX (英語) のようなハードウェア・アクセラレーションに対応する方法です。

変換と最適化により、精度にほとんど影響を与えることなく、モデルは「軽量」で高速になります。



Validate quantized model accuracy

As we can see, there is not significant difference between int8 and float model result in single image test. To understand how quantization influence on model prediction precision, we can compare model accuracy on dataset.

```
[27]: int8_stats = test(quantized_model, core, tqdm(data_loader))
```

```
100% 5000/5000 [03:10<00:00, 30.34it/s]
```

```
[28]: print("FP32 model accuracy")
print_stats(fp_stats, validator.seen, validator.nt_per_class.sum())

print("INT8 model accuracy")
print_stats(int8_stats, validator.seen, validator.nt_per_class.sum())
```

```
FP32 model accuracy
               Class      Images      Labels   Precision      Recall      mAP@.5  mAP@.5:.95
                 all        5000       36335       0.633       0.474       0.521       0.371
INT8 model accuracy
               Class      Images      Labels   Precision      Recall      mAP@.5  mAP@.5:.95
                 all        5000       36335       0.633       0.474       0.519       0.369
```

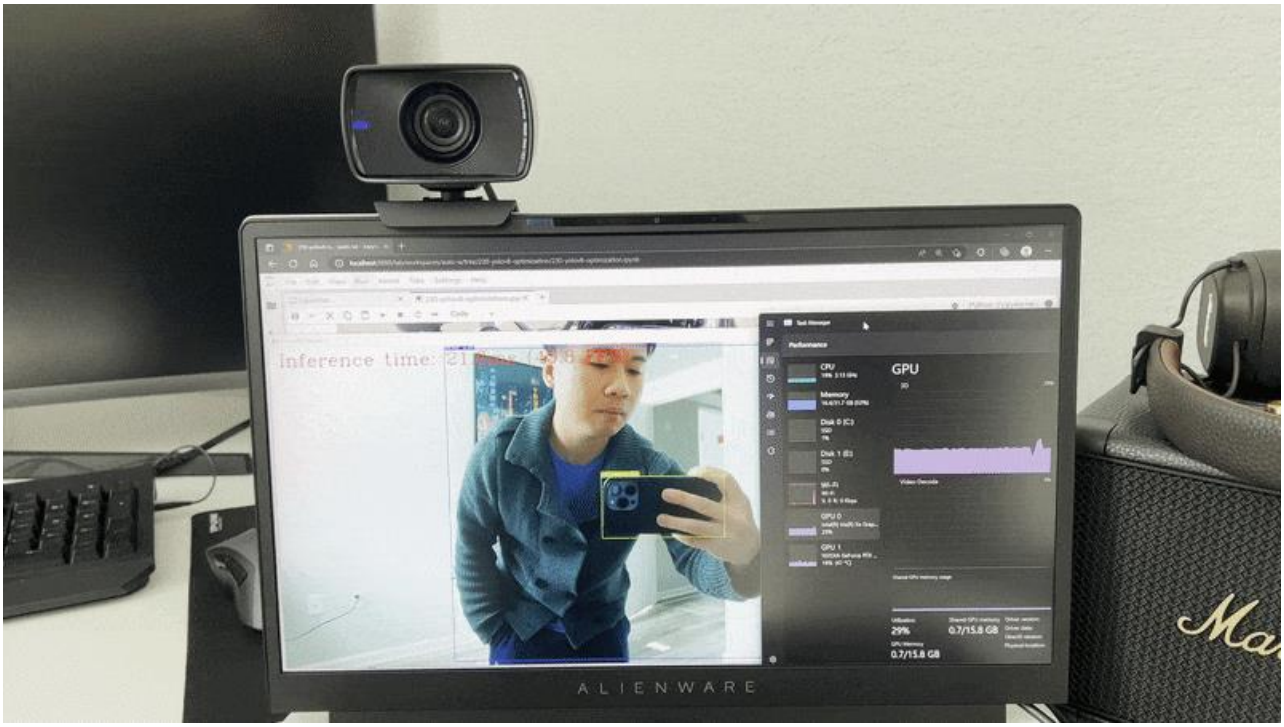
Great! Looks like accuracy was changed, but not significally and fulfit passing criteria.

シングルストリームの推論

第 12 世代インテル® Core™ プロセッサーを搭載したノートブックでは、(1 つの 1080p webcam で) 45fps でした。驚くべきことに、ノートブックは「フル稼働」状態ではなく、ほかの多くの推論タスクを実行する余裕があります。

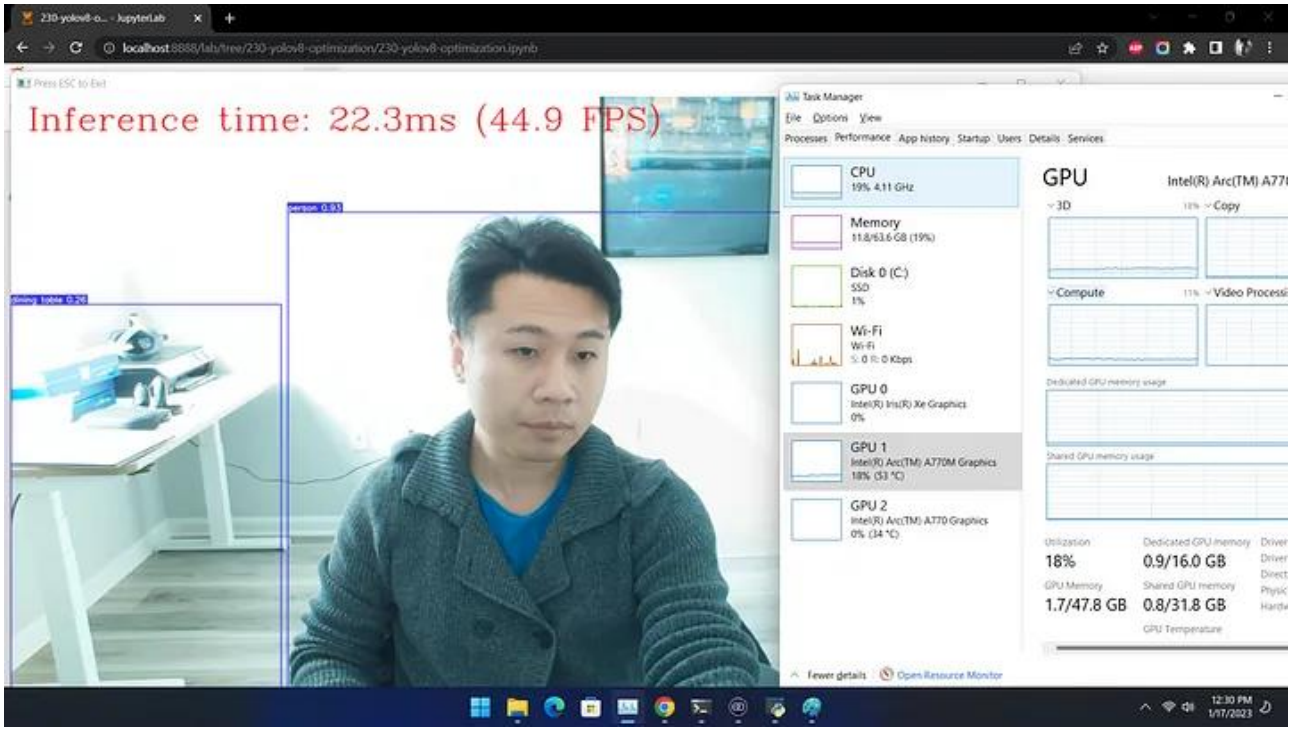
どんな魔法を使ったのでしょうか? その魔法とは、OpenVINO™ の GPU プラグイン (英語) によるデバイスの切り替え (*device = "GPU"*) でした。

```
run_object_detection(source=0, flip=True, use_popup=False, model=ov_model, device="GPU")
```



iGPU と OpenVINO™ で YOLOv8 を実行。かなりシームレスです。

インテル® Arc™ A770m グラフィックスに切り替えると、10% の負荷では GPU はほとんど使用されていません。



dGPU は、使用率が比較的低い状態で多くの推論を並列で実行できます。

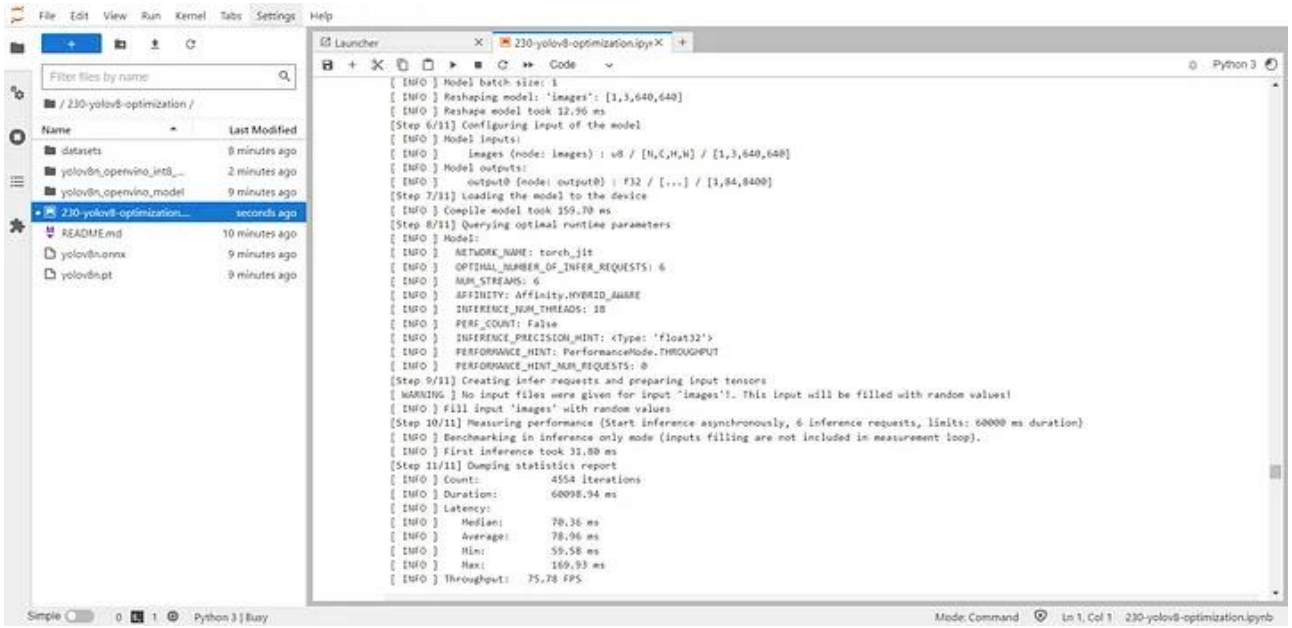
1000fps 越えを達成するには?

ここで、記事のタイトルに戻ります。どうすれば 1000fps 越えを達成できるのでしょうか? その秘訣は、スループット・モードでマルチストリームと複数の推論リクエストを使用する (つまり、複数を並列で実行する) ことです。例えば、"benchmark_app" を使用して、モデルのスループットを計算できます。ボトルネックが発生しないように前処理パイプラインと後処理パイプラインを微調整する必要があります。

前置きはこれくらいにして、NUC の CPU -> iGPU -> dGPU のベンチマーク結果を示します。

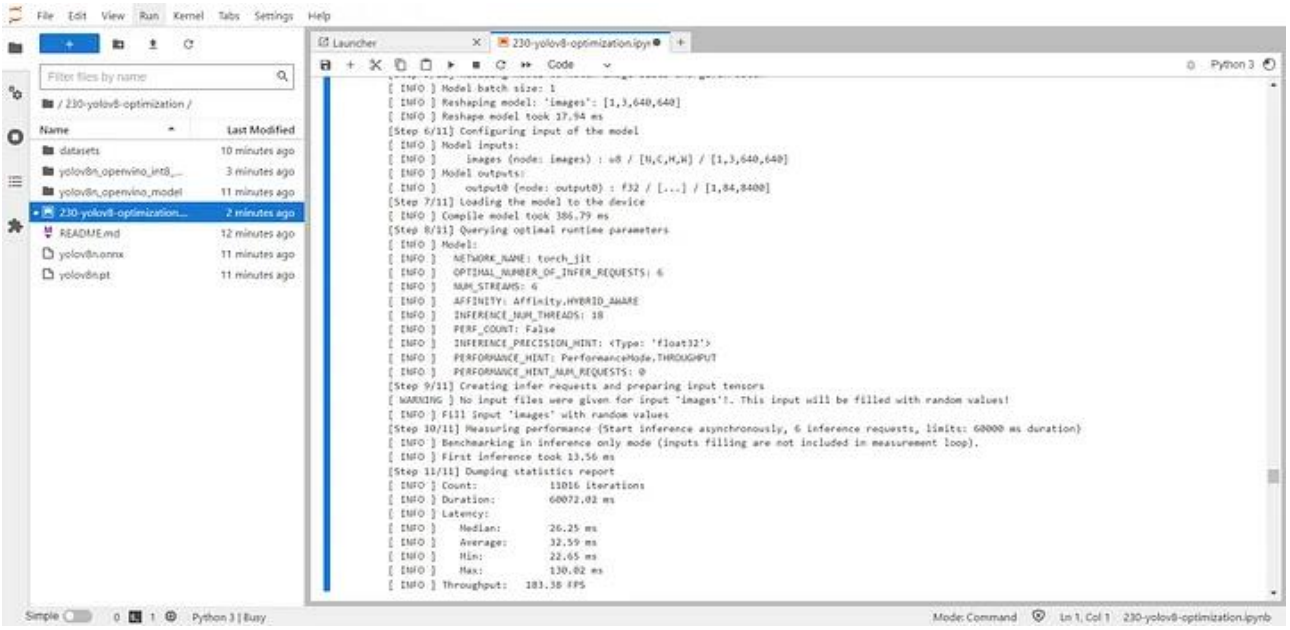
NUC のセットアップ

```
# Inference FP32 model (OpenVINO IR)
!benchmark_app -m $model_path -d CPU -api async -shape "[1,3,640,640]"
```



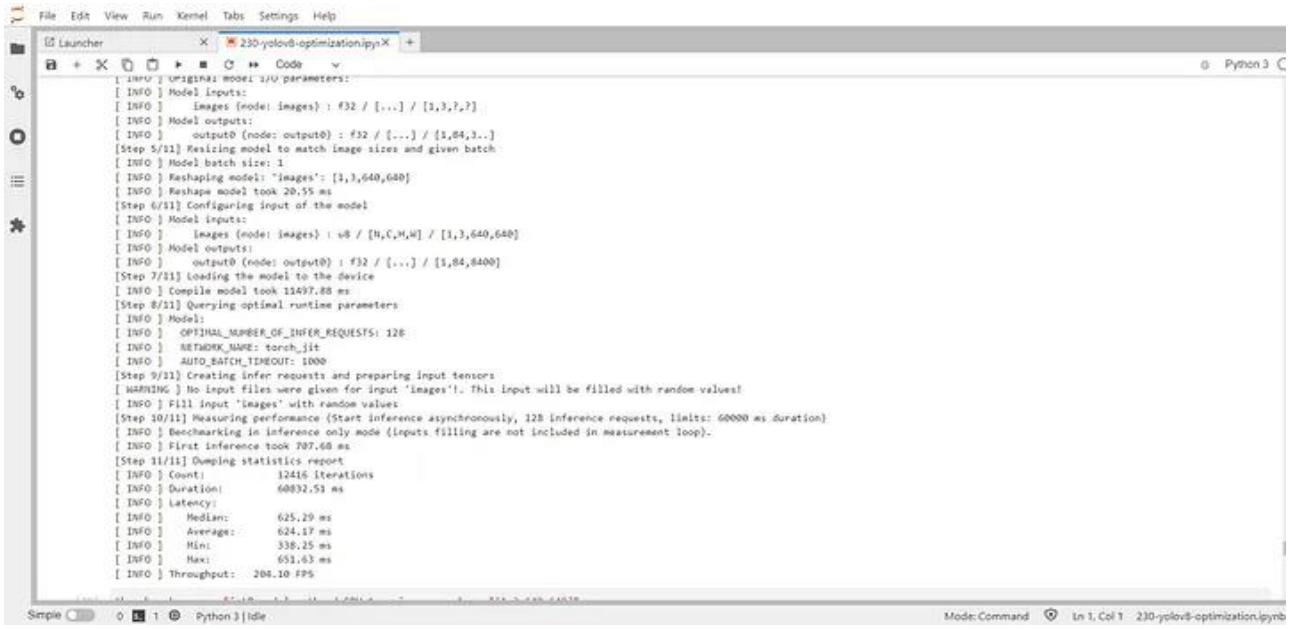
CPU 上の FP32 モデルでは ~75fps

```
# Inference INT8 model (OpenVINO IR)
!benchmark_app -m $int8_model_path -d CPU -api async -shape "[1,3,640,640]"
```



CPU 上の INT8 モデルでは ~183fps

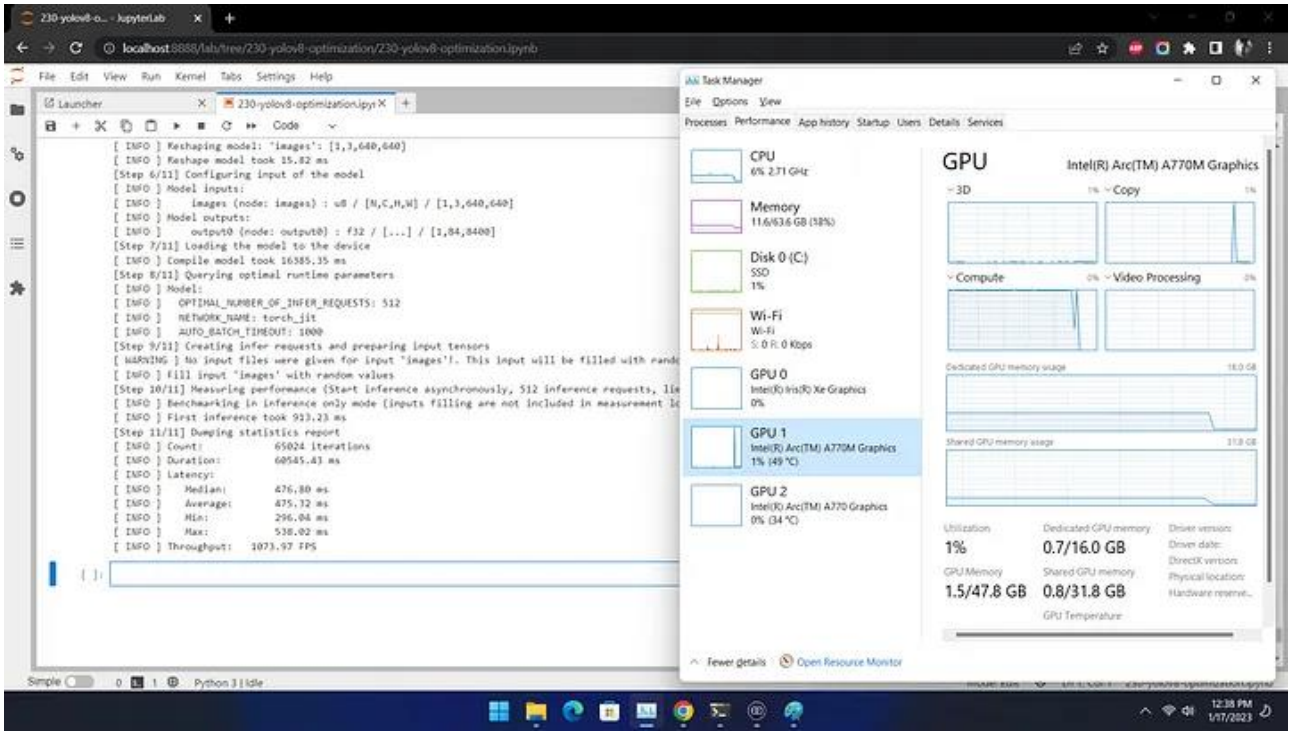
```
!benchmark_app -m $int8_model_path -d GPU.0 -api async -shape "[1,3,640,640]"
```



iGPU 上では ~204fps

```
!benchmark_app -m $int8_model_path -d GPU.1 -api async -shape "[1,3,640,640]"
```

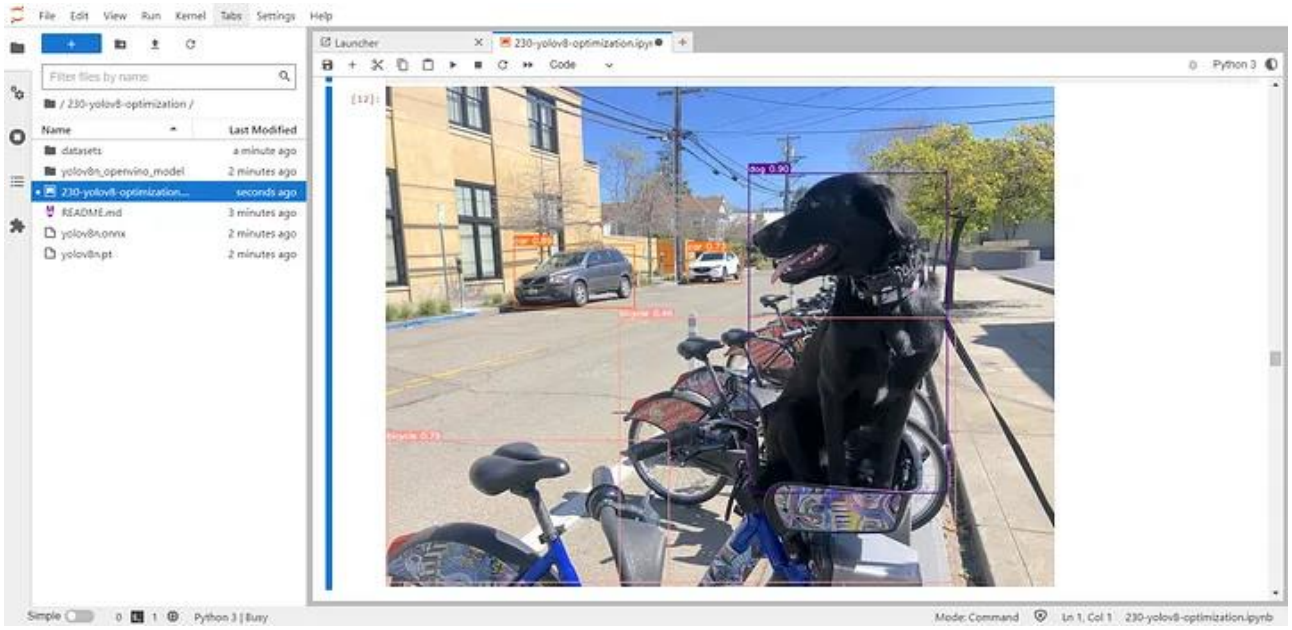
準備はいいですか?



なんと、インテル® Arc™ A770m グラフィックス上の INT8 モデルでは 1073.97fps になりました!

1000fps 越え達成です!

途中の過程を詳しく知りたい場合は、OpenVINO™ ノートブック (英語) のコードを確認してください。皆さんのコメントを歓迎します。



最適化ツールを実行し、結果をプレビューしてコーディング方法を学ぶこともできます。

試してみて、結果をお知らせください。コーディングを楽しみましょう!

#iamintel

OpenVINO™ ツールキットとは

AI を加速する無償のツールである OpenVINO™ ツールキットは、インテルが無償で提供しているインテル製の CPU や GPU、VPU、FPGA などのパフォーマンスを最大限に活用して、コンピューター・ビジョン、画像関係をはじめ、自然言語処理や音声処理など、幅広いディープラーニング・モデルで推論を最適化し高速化する推論エンジン/ツールスイートです。

OpenVINO™ ツールキット・ページでは、ツールの概要、利用方法、導入事例、トレーニング、ツール・ダウンロードまでさまざまな情報を提供しています。ぜひ特設サイトにアクセスしてみてください。

https://www.intel.co.jp/content/www/jp/ja/internet-of-things/openvino-toolkit.html

法務上の注意書き