OpenVINO™ を使用した AI PC への Llama3 のデプロイ

この記事は、Medium に公開されている「Deployment of Llama3 on Your AI PC with OpenVINO™」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。

大規模言語モデル (LLM) の開発速度は驚異的です。2024 年 4 月 18 日に、Meta は Llama シリーズの新世代モデル Llama3 を正式に発表し、この分野における新しいマイルストーンを打ち立てました。Llama3 は、以前のモデルの強力な機能を継承するだけでなく、技術の革新により、マルチモーダルの理解、長文テキスト処理、言語生成において質的に飛躍的な進歩を遂げています。Llama3 のオープン性と柔軟性は、開発者に前例のない利便性も提供します。モデルを微調整する場合でも、既存のシステムに統合する場合でも、Llama3 は優れた適応性と使いやすさを発揮します。

さらに、Llama3 モデルのデプロイでは、クラウドデプロイは別として、モデルのローカルデプロイにより、開発者はクラウドの計算リソースに依存することなく、データ処理と大規模モデル推論で高い効率とプライバシーを実現できます。OpenVINO™ を使用して Llama3 をローカル (AI PC など) にデプロイすると、応答時間が短縮され、運用コストが削減されるだけでなく、データ・セキュリティーが効果的に保護され、機密情報の漏洩を防ぐことができます。

この記事では、Llama3 モデルについて簡単に紹介し、OpenVINO™ を使用して推論を最適化、高速化、AI PC にデプロイして、AI 推論を高速かつスマートにする方法を説明します。

Llama3 の概要

Llama3 は、8B および 70B パラメーター・モデルなど、さまざまなパラメーター・スケールのモデルを提供します。コアの機能と主な利点は、次のようにまとめることができます。

● 高度な機能とパフォーマンス: 推論、言語生成、コード実行において最先端のパフォーマンスを提供し、LLM の新しい業界標準を確立します。

● 効率の向上: グループ・クエリー・アテンション (GQA) を備えたデコーダー専用トランスフォーマー・アーキテクチャーを活用し、言語エンコードの効率と計算リソースの使用率の両方を最適化して、大規模な AI タスクに適するようにします。

● 包括的なトレーニングと微調整: 15 兆を超えるトークンで事前トレーニングされ、SFT や PPO などの革新的な命令微調整手法で強化された Llama3 は、複雑な多言語タスクや多様な AI アプリケーションの処理に優れています。

● オープンソース・コミュニティーへの取り組み: Meta のオープンソース・イニシアチブの一環としてリリースされた Llama3 は、コミュニティーの関与とイノベーションを促進し、開発者が簡単にアクセスして開発に貢献できるエコシステムをサポートしています。

OpenVINO™ を使用した最適化、高速化、AI PC へのデプロイ

前述のとおり、Llama3 モデルをローカル AI PC にデプロイすると、応答時間が短縮され、運用コストが削減されるだけでなく、データ・セキュリティーも効果的に保護されます。これは、ヘルスケア、金融、パーソナル・アシスタントなど、機密性の高いデータを処理する必要があるアプリケーションでは特に重要です。

Llama-3–8B-Instruct を最適化し、推論を高速化して、AI PC にデプロイするプロセスには、次の手順が含まれます。一般的に使用される OpenVINO™ ノートブックの GitHub* リポジトリー(英語)の llm-chatbot サンプルコードを使用して具体的に説明します。詳細情報と完全なソースコードは、こちら(英語)を参照してください。

前提条件パッケージのインストールから始める

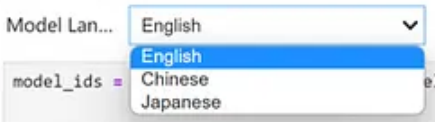
OpenVINO™ ノートブックのリポジトリーを実行するための詳細なインストール・ガイドは、こちら(英語)を参照してください。llm-chatbot サンプルコードを実行するには、まず次の前提条件パッケージをインストールする必要があります。

```
%pip uninstall -q -y openvino-dev openvino openvino-nightly optimum optimum-intel
%pip install -q --extra-index-url https://download.pytorch.org/whl/cpu\
"git+https://github.com/huggingface/optimum-intel.git"\
"git+https://github.com/openvinotoolkit/nncf.git"\
"torch>=2.1"\
"datasets" \
"accelerate"\
"openvino-nightly"\
"gradio>=4.19"\
"onnx" "einops" "transformers_stream_generator" "tiktoken" "transformers>=4.38.1" "bitsandbytes"
```

推論のモデルを選択

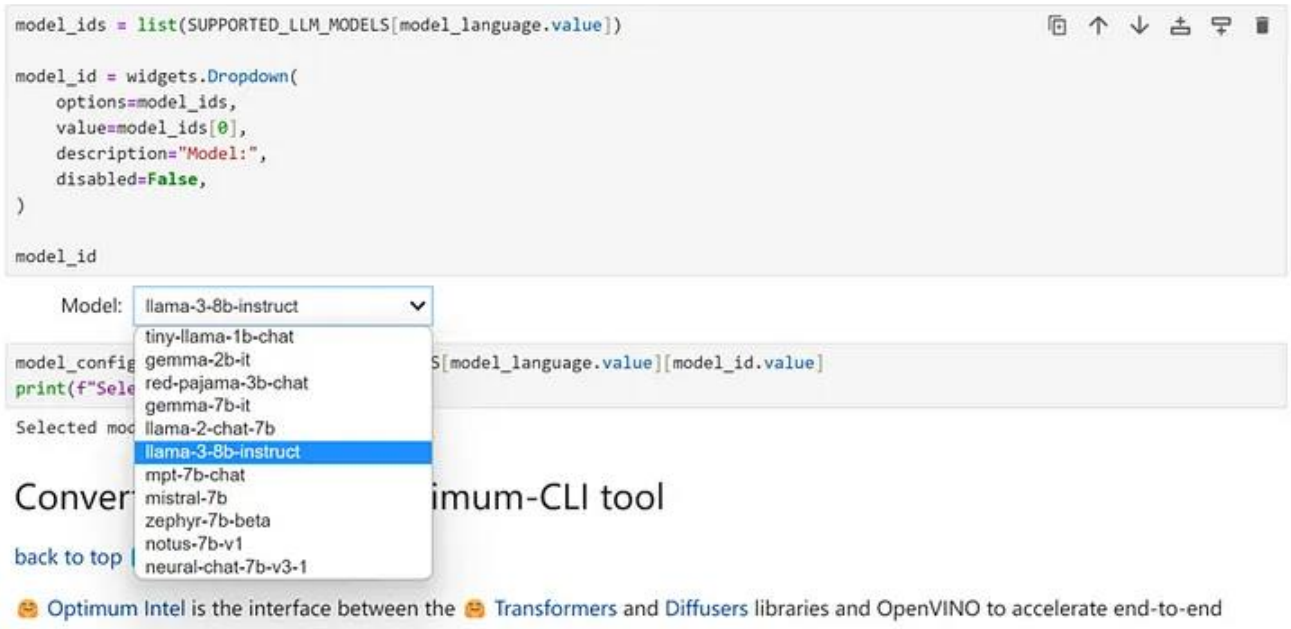
Jupyter* Notebook のデモでは、OpenVINO™ でサポートされている LLM のセットを複数の言語で提供しています。最初にドロップダウン・ボックスから言語を選択します。ここでは、[English] を選択しています。

```
model_languages = list(SUPPORTED_LLM_MODELS)

model_language = widgets.Dropdown(
    options=model_languages,
    value=model_languages[0],
    description="Model Language:",
    disabled=False,
)

model_language
```




```
model_ids =                                el_language.value])
```

次に、[llama-3–8b-instruct] を選択して、このモデルで残りの最適化と推論の高速化の手順を実行します。もちろん、ドロップダウン・ボックスにリストされているほかのモデルに切り替えることもできます。



Optimum-CLI を使用してモデルを変換

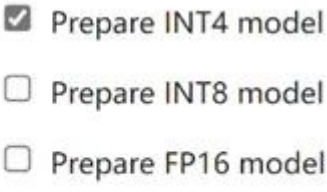
Optimum Intel(英語)は、Hugging Face Transformers および Diffusers ライブラリーと OpenVINO™ 間のインターフェイスとして機能し、インテル® アーキテクチャーでエンドツーエンドのパイプラインを高速化するように設計されていて、モデルを OpenVINO™ 中間表現 (IR) 形式にエクスポートするための使いやすい CLI (コマンドライン・インターフェイス)を提供します。モデルをエクスポートするには、次のコマンドを実行します。

```
optimum-cli export openvino --model <model_id_or_path> --task <task> <out_dir>
```

ここで、<model_id_or_path> は、HuggingFace Hub または(save_pretrained メソッドを使用して保存した)モデルがあるローカル・ディレクトリーのモデル ID、<task> は、エクスポートしたモデルが解決する必要があるサポートしているタスク(英語)の 1 つです。LLM の場合は、text-generation-with-past です。モデルの初期化にリモートコードの使用が必要な場合は、trust-remote-code オプションを追加で指定する必要があります。

モデルの重みを圧縮

「Llama-3–8B-Instruct」などの LLM は、人間のようなテキストの理解と生成においてさらに強力かつ複雑になっていますが、これらのモデルの管理とデプロイは、特に AI PC などのクライアント・デバイスでは、計算リソース、メモリー・フットプリント、推論速度が大きな課題となります。重み圧縮アルゴリズムは、モデルの重みを圧縮することを目的としており、LLM のように重みのサイズが活性化よりも比較的大きな大規模モデルの空間占有率とパフォーマンスを最適化するために使用できます。INT8 圧縮と比較して、INT4 圧縮は予測の質が若干低下しますが、モデルサイズをさらに削減してテキスト生成のパフォーマンスを向上できます。そのため、ここではモデルの重みを INT4 精度に圧縮することを選択します。



浮動小数点と圧縮モデルのバリアントを選択できるようになりました

Optimum-CLI を使用したモデル圧縮

Optimum-CLI ツールは、線形、畳み込み、埋め込み層に FP16、INT8、INT4 ビットの重み圧縮を適用するオプションを備えていて、モデルを容易にエクスポートできます。この機能は、モデルサイズと推論速度を最適化し、モデルの動作効率を向上するために重要です。適用方法は簡単です。weight-format を FP16、INT8、または INT4 に設定します。最適化により、メモリー使用量と推論のレイテンシーを減らすことができます。デフォルトでは、INT8/INT4 の量子化スキームは非対称量子化になります。対称圧縮を使用する必要がある場合は、sym オプションを追加します。

INT4 量子化の場合、次のように「Llama-3–8B-Instruct」のパラメーターを指定します。

```
compression_configs = {
        "llama-3-8b-instruct": {
            "sym": True,
            "group_size": 128,
            "ratio": 0.8,
        },
}
```

group_size パラメーターは、量子化に使用するグループサイズを定義します。

ratio パラメーターは、4 ビットと 8 ビットの量子化の比率を制御します。上記の場合、層の 80% を INT4 に量子化し、層の 20% を INT8 に量子化することを意味します。

Optimum-CLI を使用したモデル圧縮は、次のコードで実行できます。

```
optimum-cli export openvino --model "llama-3-8b-instruct" --task text-
generation-with-past --weight-format int4
--group-size 128 --ratio 0.8 --sym
```

モデル圧縮の後、この 8B パラメーター・モデルのモデルサイズが約 5GB に減っていることが分かります。

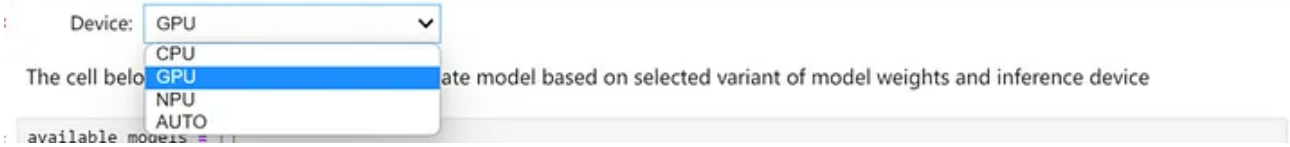
```
fp16_weights = fp16_model_dir / "openvino_model.bin"
int8_weights = int8_model_dir / "openvino_model.bin"
int4_weights = int4_model_dir / "openvino_model.bin"

if fp16_weights.exists():
    print(f"Size of FP16 model is {fp16_weights.stat().st_size / 1024 / 1024:.2f} MB")
for precision, compressed_weights in zip([8, 4], [int8_weights, int4_weights]):
    if compressed_weights.exists():
        print(f"Size of model with INT{precision} compressed weights is {compressed_weights.stat().st_size / 1024 / 1024:.2f}
    if compressed_weights.exists() and fp16_weights.exists():
        print(f"Compression rate for INT{precision} model: {fp16_weights.stat().st_size / compressed_weights.stat().st_size:.
```

```
Size of model with INT4 compressed weights is 5149.79 MB
```

推論用デバイスとモデルバリアントを選択

OpenVINO™ は広範なハードウェア・デバイスに簡単にデプロイできるため、推論を実行するデバイスを選択するドロップダウン・ボックスも用意されています。モデルサイズとパフォーマンス要件を考慮して、ここではインテル® Core™ Ultra 7 プロセッサー 155H を搭載した AI PC の GPU を推論デバイスとして選択します。

```
import openvino as ov

core = ov.Core()

device = widgets.Dropdown(
    options=core.available_devices + ["AUTO"],
    value="CPU",
    description="Device:",
    disabled=False,
)

device
```



Optimum Intel を使用してモデルをインスタンス化

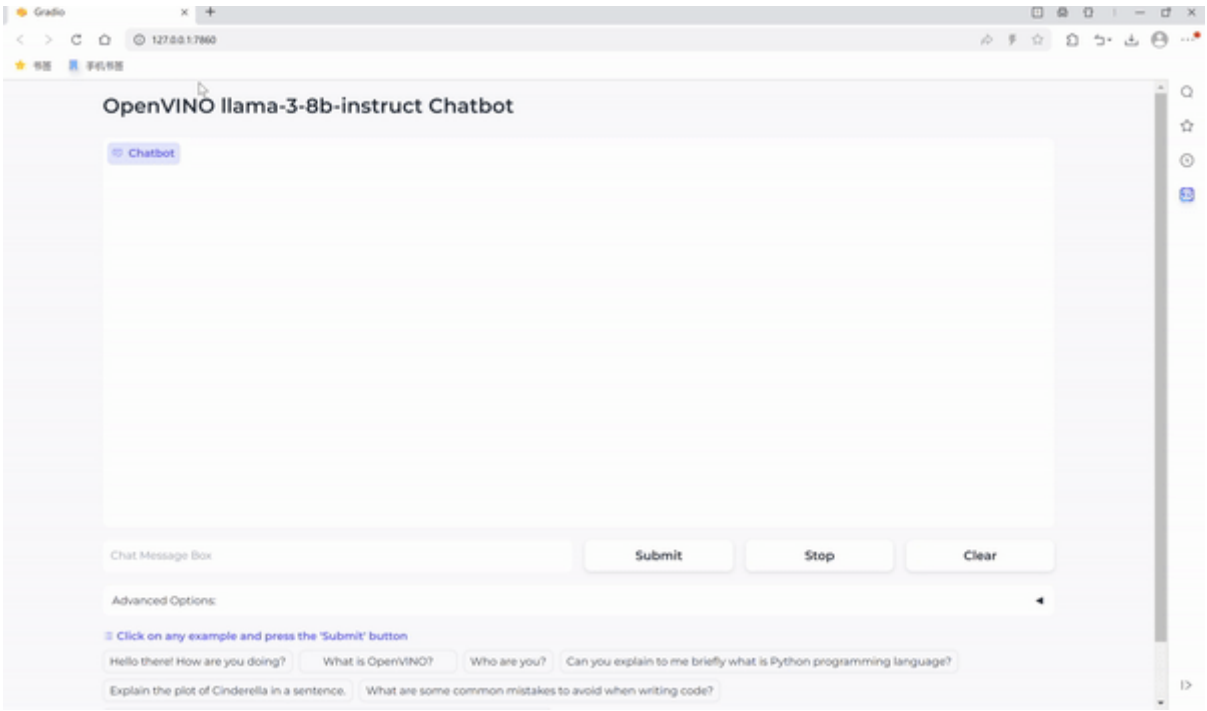
Optimum Intel を使用すると、ローカルにダウンロードされ、重み圧縮で最適化されたモデルをロードし、Hugging Face API で OpenVINO™ ランタイムを使用して推論を実行するパイプラインを作成できます。この場合、AutoModelForXxx クラスを対応する OVModelForXxx クラスに置き換えるだけで、「Llama-3–8B-Instruct」の推論パイプラインをセットアップして実行できます。

```
model_name = model_configuration["model_id"]
tok = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)

ov_model = OVModelForCausalLM.from_pretrained(
    model_dir,
    device=device.value,
    ov_config=ov_config,
    config=AutoConfig.from_pretrained(model_dir, trust_remote_code=True),
    trust_remote_code=True,
)
```

```
Loading model from llama-3-8b-instruct\INT4_compressed_weights
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
The argument `trust_remote_code` is to be used along with export=True. It will be ignored.
Compiling the model to GPU ...
```

OpenVINO™ を使用して Llama3 でチャットボットを実行

これで準備はすべて完了です。この Llama3 ベースのチャットボットの使いやすさを向上する、Gradio をベースにしたユーザーフレンドリーなインターフェイスも提供しています。では、チャットを始めましょう。



関連情報(英語)

OpenVINO™ ドキュメント

OpenVINO™ ノートブック

フィードバックの提供 & 問題の報告

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