OpenVINO™ ツールキットの重み圧縮を使用して LLM のフットプリントを削減する

この記事は、Medium に公開されている「Reduce LLM Footprint with OpenVINO™ Toolkit Weight Compression」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) により、会話型 AI が現実のものとなり、従業員の生産性を向上できる可能性を秘めた、強力なチャットボットやパーソナル・アシスタントが登場しています。しかし、LLM はサイズが非常に大きく、1,000 億以上のパラメーターが必要であり、その数はさらに増えています。この課題の解決策の 1 つは、OpenVINO™ ツールキットの重み圧縮 (英語) を使用することです。重み圧縮を使用することにより、LLM のデータストレージとメモリー/GPU ビデオ・ランダムアクセス・メモリー (vRAM) のフットプリントを元のサイズの 1/8 に削減できます。多くの環境で実行できる軽量なモデルが実現し、レイテンシーが低くなり、システムリソースの負荷が軽減されます。

この記事では、Optimum Intel (英語) API を使用した Hugging Face LLM 向けの重み圧縮と、OpenVINO™ ニューラル・ネットワーク圧縮フレームワーク (NNCF) (英語) を使用した重み圧縮の、2 つの手法について説明します。この記事は、OpenVINO™ を使用した LLM の最適化に関するソリューション・ホワイト・ペーパー (英語) から抜粋して要約したものです。

LLM のサイズ

サイズの課題に対処することは、クライアント・デバイスへのデプロイを含め、AI のデプロイをより簡単にして、日常的な利用をより実用的にするための鍵となります。LLM の合計ファイルサイズは、小規模モデルでは約 2GB、大規模モデルでは 300GB 以上になります。Llama 2–70B のような 700 億パラメーター・モデルでは、約 140GB のストレージ領域が必要です。モデルをすべてメモリーにロードできるようにするには、LLM を実行するシステムにモデルファイルのサイズと同じサイズのランダムアクセス・メモリー (RAM) が必要です。

RAM が少ない場合、システムはディスクストレージをメモリーのスワップ領域として使用するため、推論の実行速度が低下したり、メモリーが不足したときにシステムがクラッシュする可能性があります。システム GPU にも、モデルのサイズと同じサイズの vRAM と、ほかのオペレーティング・システム (OS) タスクを実行するための追加のメモリーパディングが必要です。当然、これらの高いハードウェア要件により、LLM をデプロイするための参入障壁は高くなり、強力なサーバーと、メモリーおよび GPU への多額の投資が必要になります。

重み圧縮でモデルのサイズを縮小

幸いなことに、OpenVINO™ の主要機能である重み圧縮を使用して、これらの要件を大幅に軽減できます。モデルの重みを FP32 から int8 に圧縮すると、モデルのサイズが元の 1/4 に縮小され、モデルを int4 形式に変換すると、モデルのサイズが元の 1/8 に縮小されます。例えば、FP32 形式の Zephyr 7B ベータモデルは 28GB ですが、OpenVINO™ NNCF を使用してモデルを int4 に圧縮すると、同様の精度を維持しながらサイズが 4GB に縮小されます。

モデルのサイズを縮小すると、モデルの実行に必要な RAM と GPU vRAM の量も減ります。ミッドエンドからハイエンドのプロセッサーと 16GB の RAM を搭載したクライアント・デバイスで、Llama-7B や Zephyr 7B ベータなどの OpenVINO™ 量子化 7B モデルを問題なく実行できます。これは、複雑で高性能な LLM を提供する際の、エンドユーザーのハードウェア制限を考慮している開発者やプロバイダーにとって朗報と言えます。

OpenVINO™ とは

重み圧縮は、AI モデルで「1 つのコードをどこでもデプロイ」できるようにするオープンソースの AI ツールキット、OpenVINO™ で実現できる機能の 1 つに過ぎません。OpenVINO™ は、LLM に柔軟で効率の良いランタイム環境を提供し、デプロイのサイズ、速度、さまざまなハードウェアで実行できる柔軟性を利点として提供します。インテルの公式サポートにより、ユーザーは長い間、更新と最適化の恩恵を受けることができます。

OpenVINO™ は、数ギガバイトの依存関係が必要な Hugging Face、PyTorch*、その他のフレームワークと比較して、依存関係がわずか数百メガバイトの自己完結型のパッケージです。OpenVINO™ を使用した LLM 推論の実行の詳細は、ソリューション・ホワイト・ペーパー (英語) を参照してください。

OpenVINO™ で重み圧縮を実行する利点

重みと活性化を量子化するフルモデルの量子化とは異なり、OpenVINO™ NNCF を使用した重み圧縮はモデルの重みのみが対象です。このアプローチでは、活性化が浮動小数点数のまま、モデルの精度を維持することができます。開発者は活性化値の範囲を調整する必要もないため、フルの量子化よりも簡単で効率的です。

重み圧縮のデータ型

OpenVINO™ NNCF は、8 ビット (int8)、4 ビット対称 (int4_SYM)、4 ビット非対称 (int4_ASYM) の 3 種類の重み圧縮をサポートしています。種類ごとに、長所と短所があります。

- **int8、8 ビット重み量子化:** このデフォルトの圧縮手法は、重みを 8 ビット整数のデータ型に量子化します。モデルのサイズの縮小と精度のバランスが取れた、幅広いアプリケーション向けの汎用的なオプションです。
- **int4_SYM、4 ビット対称重み量子化:** int4 対称モードは、重みを固定のゼロ点 8 (つまり、0 と 15 の中間点) を中心として対称的に符号なし 4 ビット整数に量子化します。推論は int8 精度のモデルより高速なため、精度に対する許容可能なトレードオフよりも速度が優先される状況に最適です。
- **int4_ASYM、4 ビット非対称重み量子化:** int4 非対称モードも重みを符号なし 4 ビット整数に量子化しますが、固定でないゼロ点を使用して非対称的に量子化します。このモードでは、対称モードよりも精度を高めるために速度がわずかに低下しますが、int8 よりもパフォーマンスは高くなります。

OpenVINO™ で重み圧縮を実行する

OpenVINO™ で重み圧縮を実行する主な手法は 2 つあります。

- Optimum Intel API を使用して Hugging Face LLM で重み圧縮を実行する。
- コマンドライン・インターフェイス (CLI) を使用して Hugging Face モデルを OpenVINO™ 中間表現 (IR) 形式に変換し、OpenVINO™ NNCF を使用して重み圧縮を実行する。

どちらの手法でも、開発者はリアルタイム・アプリケーションではなくオフラインで重み圧縮を実行する必要があります。LLM を開発環境で圧縮してエクスポートし、デプロイ環境で使用します。

要件

OpenVINO™ を使用するには、OpenVINO™ のインストール手順 (英語) に従って、OpenVINO™ 向けの Python* 仮想環境を設定します。環境を作成して有効にしたら、次のコマンドを実行して、Optimum Intel、OpenVINO™、NNCF および依存関係を Python* 環境にインストールします。

```
pip install optimum[openvino]
```

Optimum Intel を使用して Hugging Face モデルで重み圧縮を実行する

以下の例は、Hugging Face モデルで重み圧縮を実行する方法を示しています。この例では、Optimum Intel を使用して Zephyr 7B ベータモデルを Hugging Face からロードします。モデルがロードされると、NNCF を使用して指定した圧縮タイプに自動的に圧縮されます。

圧縮タイプは、OVModelForCausalLM.from_pretrained メソッドで `compression_option="<オプション>"` 引数を使用して指定します。このオプションを `none` に設定すると、10 億パラメーター以上のデコーダーモデルでは、int8 重み圧縮がデフォルトで有効になります。次のいずれかのオプションを利用できます。

- "int8" : NNCF を使用した int8 圧縮
- "int4_sym_g128" : グループサイズ 128 の対称 int4 圧縮
- "int4_asym_g128" : グループサイズ 128 の非対称 int4 圧縮

- "int4_sym_g64" : グループサイズ 64 の対称 int4 圧縮
- "int4_asym_g64" : グループサイズ 64 の非対称 int4 圧縮

グループサイズの詳細は、OpenVINO™ ドキュメントの「重み圧縮 (英語)」ページを参照してください。この例の、compression_option="int8" は、int8 量子化を実行することを示しています。

```
from nncf import compress_weights, CompressWeightsMode
from optimum.intel.openvino import OVModelForCausalLM
from transformers import AutoTokenizer, pipeline

# Load model from Hugging Face and compress to int8
model_id = HuggingFaceH4/zephyr-7b-beta"
model = OVModelForCausalLM.from_pretrained(model_id, export=True,
compression_option="int8")

# Inference
tokenizer = AutoTokenizer.from_pretrained(model_id)
pipe = pipeline("text-generation", model=model, tokenizer=tokenizer)
phrase = "The weather is"
results = pipe(phrase)
print(results)

# Save compressed model and tokenizer for later use
model.save_pretrained("zephyr-7b-beta-int8-sym-ov")
tokenizer.save_pretrained("zephyr-7b-beta-int8-sym-ov")
```

例の最後で、将来のセッションでインポートして使用できるように、圧縮したモデルとトークナイザーを保存していることに注意してください。圧縮したモデルを保存することにより、新しいセッションでモデルを使用するたびにモデルを圧縮する時間を節約できます。

NNCF で使用できるように Hugging Face モデルを OpenVINO™ IR 形式に変換する

NNCF を実行する前に、**optimum-cli** ツールを使用して Hugging Face モデルを IR 形式に変換する必要があります。このツールは、Python* スクリプトを使用しないでモデルを変換する場合に役立ちます。

この変換を実行するコマンドの構造を次に示します。

```
optimum-cli export openvino --model <MODEL_NAME> <NEW_MODEL_NAME>
```

-- model <MODEL_NAME>: コマンドのこの部分は、変換するモデルの名前を指定します。<MODEL_NAME> を Hugging Face の実際のモデル名に置き換えます。

<NEW_MODEL_NAME>: ここで、OpenVINO™ IR 形式の新しいモデルに付ける名前を指定します。<NEW_MODEL_NAME> を新しいモデルに付ける名前に置き換えます。

例えば、Hugging Face の Llama 2–7B モデル (正式名称は meta-llama/Llama-2–7b-chat-hf) を OpenVINO™ IR モデルに変換して、「ov_llama_2」という名前を付けるには、次のコマンドを使用します。

```
optimum-cli export openvino --model meta-llama/Llama-2-7b-chat-hf ov_llama_2
```

この例では、**meta-llama/Llama-2–7b-chat-hf** が Hugging Face のモデル名で、**ov_llama_2** が変換後の OpenVINO™ IR モデルの新しい名前です。

さらに、CLI を使用してモデルをエクスポートするときに、`--weight-format` 引数を指定して 8 ビットまたは 4 ビットの重み量子化を適用できます。モデル **gpt2** に 8 ビットの量子化を適用するコマンドの例を次に示します。

```
optimum-cli export openvino --model gpt2 --weight-format int8 ov_gpt2_model
```

NNCF で重み圧縮を実行する

OpenVINO™ IR モデルが用意できたら、NNCF を実行できます。この例では、ov.core.read_model を使用して OpenVINO™ IR 形式のモデルを読み取ります。nncf.compress_weights メソッドは、モデルの重みを指定したデータ型に量子化します。

nncf.compress_weights メソッドで使用する圧縮タイプは、`mode` 引数を使用して設定します。次の 3 つのオプションがあります。

- mode=CompressWeightsMode.int8
- mode=CompressWeightsMode.int4_SYM
- mode=CompressWeightsMode.int4_ASYM

この例では、int4_SYM モードを設定して 4 ビット対称量子化を使用しています。

```
from nncf import compress_weights, CompressWeightsMode
import openvino as ov

# Read an OpenVINO IR model
core = ov.Core()
model = core.read_model("model.xml")

# Compress to int4 Symmetric
model = compress_weights(model, mode=CompressWeightsMode.int4_SYM)

# Save compressed model for later use
ov.save_model(model, "model-int4-sym.xml")
```

NNCF では、group_size と ratio 圧縮パラメーターを設定して、圧縮モデルのサイズと推論速度を微調整することもできます。詳細は、OpenVINO™ ドキュメントの「重み圧縮 (英語)」ページを参照してください。

圧縮後の LLM のベンチマーク

重み圧縮の後、圧縮がパフォーマンスに与える影響を確認するため、LLM のベンチマークを行うことを推奨します。OpenVINO™ GenAI (英語) リポジトリーには、Optimum Intel で提供されるパイプラインに基づいて LLM のパフォーマンスを推定する統一されたアプローチを提供する LLM ベンチマーク・ツール (英語) が含まれています。このツールを使用し、次の手順に従って PyTorch* モデルと OpenVINO™ モデルのパフォーマンスを推定できます。

`requirements.txt` を使用してベンチマークの依存関係をインストールします。

```
pip install -r requirements.txt
```

注: `pip install` コマンドで OpenVINO™ のバージョンを指定できます。

```
# e.g.
pip install openvino==2024.0.0
```

次のコマンドを使用して、LLM のパフォーマンスをテストします。

```
python benchmark.py -m <model> -d <device> -r <report_csv> -f <framework> -p
<prompt text> -n <num_iters>

# e.g.
python benchmark.py -m models/llama-2-7b-chat/pytorch/dldt/FP32 -n 2
python benchmark.py -m models/llama-2-7b-chat/pytorch/dldt/FP32 -p "What is
openvino?" -n 2
python benchmark.py -m models/llama-2-7b-chat/pytorch/dldt/FP32 -pf
prompts/llama-2-7b-chat_1.jsonl -n 2
```

コマンド・パラメーター:

- -m – モデルパス
- -d – 推論デバイス (デフォルト = cpu)
- -r – レポート csv
- -f – フレームワーク (デフォルト = ov)
- -p – インタラクティブ・プロンプト・テキスト
- -pf – インタラクティブ・プロンプトを含む JSONL ファイルのパス
- -n – ベンチマークの反復回数、値が 0 よりも大きい場合、最初の反復は除外されます (デフォルト = 0)

```
python ./benchmark.py -h # for more information
```

LLM の精度の測定

lm-evaluation-harness (英語) ツールは、LLM の精度を測定するためのサードパーティーのテストハーネスで、最近、OpenVINO™ のサポートが追加されました。モデルの精度の測定に使用する方法の詳細は、リポジトリーを参照してください。

ハードウェア要件や多くのパラメーター向けに調整できるように LLM を軽量にする

LLM 開発者とプロバイダーは、多くのデータとパラメーターを使用して、デプロイを簡素化しつつ、モデルの精度を向上する方法を探しています。重み圧縮は、モデルのサイズとメモリー・フットプリントを抑えながらこれらの目標をサポートする便利なツールです。OpenVINO™ は、速度、柔軟性、ハードウェア・サポートを含む、多くの利点を LLM 推論にもたらします。OpenVINO™ ツールキットを自分で試して、OpenVINO™ の旅を続けてください。

著者の帰属

この記事は、インテル AI エバンジェリストの Ria Cheruvu とインテル OpenVINO™ プロダクト・マネージャーの Ryan Loney によるソリューション・ホワイト・ペーパー「OpenVINO™ ツールキットを使用した大規模言語モデルの最適化 (英語)」の情報を抜粋して要約したものです。

追加のクレジット (敬称略): Ekaterina Aidova、Alexander Kozlov、Helena Kloosterman、Artur Paniukov、Dariusz Trawinski、Ilya Lavrenov、Nico Galoppo、Jan Iwaszkiewicz、Sergey Lyalin、Adrian Tobiszewski、Jason Burris、Ansley Dunn、Michael Hansen、Raymond Lo、Yury Gorbachev、Adam Tumialis、Milosz Zeglarski

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