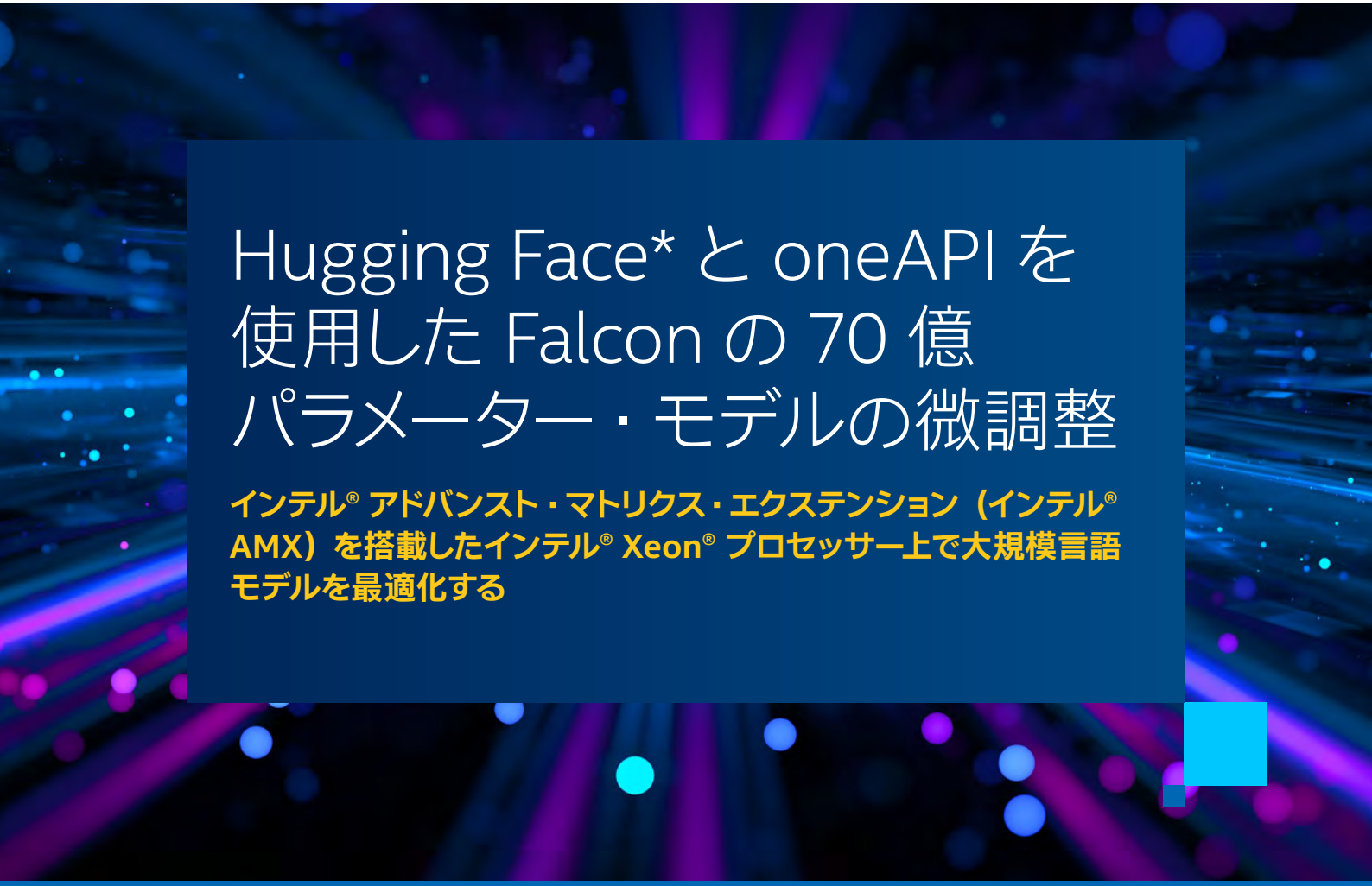

Hugging Face* と oneAPI を使用した Falcon の 70 億パラメーター・モデルの微調整

インテル® アドバンスト・マトリクス・エクステンション（インテル® AMX）を搭載したインテル® Xeon® プロセッサー上で大規模言語モデルを最適化する

Eduardo Alvarez インテル コーポレーション シニア AI ソリューション・エンジニア

大規模言語モデル（LLM）のオープンソース化は、あらゆる場所から AI テクノロジーにアクセスできるようにするのに大いに役立ちます。次の AI 研究のブレークスルーが大規模に分散されたアクセラレーターのクラスターにアクセスできない誰かによってもたらされる可能性はゼロではありませんが、その可能性は限りなく低いでしょう。しかし、AI アプリケーション開発では話がまったく異なり、製品開発インフラストラクチャーを選択する際の柔軟性が高くなることから、CPU の可用性とスケーラビリティーの交点および Falcon LLM（英語）の背後にある真のオープンソース・ライセンスが、AI の可能性における重要な要素となります。

この記事では、Hugging Face* の教師あり微調整トレーナー（SFTTrainer）、PyTorch* 向けインテル® エクステンション（IPEX）、インテル® アドバンスト・マトリクス・エクステンション（インテル® AMX）、自動混合精度（AMP）、Bfloat16 を使用して、インテル® Xeon® プロセッサー上で最先端の Falcon 70 億言語モデル（Falcon-7B）を微調整するというエキサイティングな挑戦について説明します。

環境のセットアップ

次のように環境をセットアップします。

miniconda（英語）をインストールします。

1. conda 環境を作成します。
   ```
   conda create -n falconft python==3.8.10
   ```
2. 依存関係をインストールします。
   ```
   pip install -r requirements.txt
   ```
   `requirements.txt` ファイルには、以下の依存関係がリストされています。

   ```
   torch==2.0.1
   transformers==4.30.1
   bitsandbytes==0.39.0
   peft==0.3.0
   accelerate==0.20.3
   datasets==2.12.0
   trl==0.4.4
   einops==0.6.1
   scipy==1.10.1
   intel_extension_for_pytorch==2.0.100
   ```

3. conda 環境をアクティベートします。
   ```
   conda activate falconft
   ```

因果関係言語モデルの微調整

因果関係言語モデルには、前のコンテキストに基づくシーケンス内の次の単語の予測が含まれていて、テキスト生成などのタスクを可能にしています。Falcon-7B などのモデルを特定のタスクに合わせて微調整するには、タスク固有のラベル付きデータを提供して、事前トレーニング済みモデルを適応させる必要があります。モデルはこのデータでさらにトレーニングされ、パラメーターを調整して新しいタスクのパフォーマンスを最適化します。このプロセスを通じて、Falcon-7B は特定の因果関係があるタスクのパターンと複雑さを徐々に学習し、その特定のユースケースに合わせて一貫した文脈的に適切なテキストを生成できるようにします。

会話ツリーの最も評価の高いパスのみを含む Open Assistant データセットのサブセット（合計 9,846 サンプル）を使用します。微調整と転移学習の詳細は、この記事（英語）を参照してください。

ディープラーニング・タスクのデフォルトの選択肢は GPU ですが、CPU で Falcon-7B を微調整すると、次のような利点が得られます。

- **可用性：** CPU は至る所に存在し、簡単にアクセスできるため、高価な GPU クラスターにアクセスできない研究者や実践者にとって魅力的な選択肢となります。
- **コスト：** 一般に、大規模なデプロイでは、CPU のほうが GPU よりも費用対効果は高くなります。
- **互換性：** CPU は広範なハードウェアやインフラストラクチャーと互換性があり、既存のシステムへのスムーズな統合が保証されます。

SFTTrainer、IPEX、インテル® AMX、AMP、Bfloat16 を組み合わせると、Falcon-7B の微調整がさらに効率良く効果的になります。SFTTrainer は、複雑なタスクに高レベルの抽象化を提供することにより、微調整プロセスを簡素化します。IPEX と AMP は、インテル® Xeon® プロセッサーの最新のハードウェア機能を活用します。この拡張により、オープンソースの PyTorch* にアップストリームされる前に、最新の最適化とデバイスのサポートが導入されます。AMP トレーニングと推論もサポートしており、必要に応じて完全な 32 ビット精度を維持したまま、パラメーターと操作を Bfloat16 に変換してインテル® AMX をさらに高速化します。

Falcon-7B は、アブダビの Technology Innovation Institute (TII) により開発された、70 億パラメーターのデコーダー専用モデルです。LLaMA、StableLM、RedPajama、MPT などのモデルよりもパフォーマンスが優れていて、FlashAttendant メソッドを利用して高速な推論を実現し、さまざまなタスクで速度が大幅に向上します（**図 1**）。

Model	Revision	Average	ARC (25-shot)	HellaSwag (10-shot)	MMLU (5-shot)
tiiuae/falcon-40b-instruct	main	63.2	61.6	84.4	54.1
timdettmers/guanaco-65b-merged	main	62.2	60.2	84.6	52.7
CalderaAI/30B-Lazarus	main	60.7	57.6	81.7	45.2
tiiuae/falcon-40b	main	60.4	61.9	85.3	52.7
timdettmers/guanaco-33b-merged	main	60	58.2	83.5	48.5
ausboss/llama-30b-supercot	main	59.8	58.5	82.9	44.3
pinkmanlove/llama-65b-hf	main	58.3	57.8	84.2	48.8
llama-65b	main	58.3	57.8	84.2	48.8
huggyllama/llama-65b	main	58.3	57.8	84.2	48.8
MetaIX/GPT4-X-Alpasta-30b	main	57.9	56.7	81.4	43.6

図 1. 2023 年 6 月 6 日時点の Hugging Face* LLM のリーダーボード（画像出典（英語 ））

以下のスクリプトを実行すると、Hugging Face* から「tiiuae/falcon-7b」モデルがロードされ、トークン化され、トレーニング・パラメーターが設定されて、微調整に SFTTrainer が使用されます。モデルの微調整にかかる時間は、設定した計算パラメーターとハイパーパラメーターによって異なります。スクリプトを実行する前に、次の環境変数を設定してインテル® AMX ISA が選択されるようにすることが重要です。

```
export ONEDNN_MAX_CPU_ISA=" AVX512_CORE_AMX"

# falcon-tune.py
import time
import argparse

from datasets import load_dataset
from trl import SFTTrainer
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    TrainingArguments)

def main(FLAGS):
```

```
    dataset = load_dataset("timdettmers/openassistant-guanaco", split="train")

    model_name = "tiiuae/falcon-7b"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    tokenizer.pad_token = tokenizer.eos_token
    model = AutoModelForCausalLM.from_pretrained(model_name, trust_remote_code=True)

    print('setting training arguments')

    training_arguments = TrainingArguments(
        output_dir="./results",
        bf16=FLAGS.bf16, #change for CPU
        use_ipex=FLAGS.use_ipex, #change for CPU IPEX
        no_cuda=True,
        fp16_full_eval=False,
    )

    print('Creating SFTTrainer')

    trainer = SFTTrainer(
        model=model,
        train_dataset=dataset,
        dataset_text_field="text",
        max_seq_length=FLAGS.max_seq_length,
        tokenizer=tokenizer,
        args=training_arguments,
        packing=True,
    )

    print('Starting Training')
    start = time.time()

    trainer.train()

    total = time.time() - start
    print(f'Time to tune {total}')

if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument('-bf16',
                        '--bf16',
                        type=bool,
                        default=True,
                        help="activate mix precision training with bf16")
    parser.add_argument('-ipex',
                        '--use_ipex',
                        type=bool,
                        default=True,
                        help="used to control the maximum length of the generated text in text
generation tasks")
    parser.add_argument('-msq',
                        '--max_seq_length',
                        type=int,
                        default=512,
                        help="specifies the number of highest probability tokens to consider at
```

```
each step")

    FLAGS = parser.parse_args()
    main(FLAGS)
```

次のコマンドを使用してスクリプトを実行します。

```
python falcon-tune.py --bf16 True --use_ipex True --max_seq_length 512
```

トレーニング中は、プロセスが完了するまでの推定時間を示すプログレスバーが表示されます（**図 2**）。

```
 94%|          | 3480/3693 [43:39:31<2:40:20, 45.16s/it]
{'loss': 1.6137, 'learning_rate': 4.323043595992418e-05, 'epoch': 0.41}
{'loss': 1.5278, 'learning_rate': 3.646087191984837e-05, 'epoch': 0.81}
{'loss': 1.3579, 'learning_rate': 2.9691307879772547e-05, 'epoch': 1.28}
{'loss': 0.8744, 'learning_rate': 2.2921743839696726e-05, 'epoch': 1.68}
{'loss': 0.7795, 'learning_rate': 1.6152179799620905e-05, 'epoch': 2.15}
```

図 2. 微調整プロセスのトレーニング・ログ

トレーニングが完了すると、さまざまなチェックポイント・フォルダーを含む `results` ディレクトリーが作成されます。

```
(falconft) devcloud@a4bf01930766:~/falconft/results/checkpoint-3000$ ls
config.json                     pytorch_model-00003-of-00003.bin  tokenizer.json
generation_config.json          pytorch_model.bin.index.json      tokenizer_config.json
optimizer.pt                    rng_state.pth                     trainer_state.json
pytorch_model-00001-of-00003.bin  scheduler.pt                    training_args.bin
pytorch_model-00002-of-00003.bin  special_tokens_map.json
```

図 3. 最終的なチェックポイント・フォルダーの内容

数が最も大きなチェックポイント・フォルダー（`checkpoint-3000`）には、すべての構成ファイル、PyTorch* モデルのファイルなどが含まれます（**図 3**）。モデルをデプロイして推論リクエストを処理するには、このフォルダーのファイルが必要です。

微調整された Falcon-7B モデルによる推論

モデルが微調整されたら、Hugging Face* パイプラインを作成する次のスクリプトを使用して、サンプルプロンプトでモデルをテストします。

```
# falcon-tuned-inference.py

from transformers import AutoTokenizer, AutoModelForCausalLM, AutoConfig
import transformers
import torch
import argparse
import time

def main(FLAGS):

    model = AutoModelForCausalLM.from_pretrained(FLAGS.checkpoints, trust_remote_code=True)
    tokenizer = AutoTokenizer.from_pretrained(FLAGS.checkpoints, trust_remote_code=True)
    tokenizer.pad_token = tokenizer.eos_token

    generator = transformers.pipeline(
        “text-generation”,
        model=model,
        tokenizer=tokenizer,
        torch_dtype=torch.bfloat16,
        trust_remote_code=True,
        device_map=”auto”,
    )

    user_input = “start”

    while user_input != “stop”:

        user_input = input(f”Provide Input to tuned falcon: “)

        start = time.time()

        if user_input != “stop”:
            sequences = generator(
            f””” {user_input}”””,
            max_length=FLAGS.max_length,
            do_sample=False,
            top_k=FLAGS.top_k,
            num_return_sequences=1,
            eos_token_id=tokenizer.eos_token_id,)

        inference_time = time.time() - start

        for seq in sequences:
            print(f”Result: {seq[‘generated_text’]}”)

        print(f’Total Inference Time: {inference_time} seconds’)

if __name__ == “__main__”:
    parser = argparse.ArgumentParser()

    parser.add_argument(‘-c’,
                        ‘--checkpoints’,
                        type=str,
                        default=None,
                        help=”path to model checkpoint files”)
    parser.add_argument(‘-ml’,
                        ‘--max_length’,
                        type=int,
                        default=”200”,
```

```
                        help="used to control the maximum length of the generated text in text
generation tasks")
    parser.add_argument('-tk',
                        '--top_k',
                        type=int,
                        default="10",
                        help="specifies the number of highest probability tokens to consider at
each step")

    FLAGS = parser.parse_args()
    main(FLAGS)
```

このスクリプトを実行するには、次のコマンドを実行します。

```
python falcon-tuned-inference.py --checkpoints <PATH-TO-CHECKPOINT> --max_length 200 --top_k 10
```

「Can you tell me three fun facts about space?（宇宙に関する面白い事実を 3 つ教えてもらえますか ?)」という質問に対する Falcon の返答を**図 4** に示します。土星に関する部分的に正しい事実（土星は環がある太陽系で唯一の惑星、と返答していますが、木星などにも環があることが観測されています）を除いて、モデルは事実に基づいて正確な返答を提供し、容易に解釈可能な形式に整理したようです。微調整により、モデルが改善された兆候が見受けられます。

```
Result:  Can you tell me a three fun facts about space?
Thanks!### Assistant: Sure! Here are three fun facts about space:

1. The Milky Way galaxy, which is home to the Earth and the rest of the Solar System, is estimated to contain a
pproximately 100 billion stars.

2. Saturn is the only planet in the Solar System that has a ring system, and the planet's rings are composed pr
imarily of water ice particles.

3. The Moon is the only natural satellite of the Earth, and it has been orbiting the Earth for over 4.5 billion
 years.
```

図 4. 微調整された Falcon-7B からの返答

微調整されたモデルと微調整されていない Falcon-7B モデルを比較するため、上記と同じプロンプトを使用して、微調整されていない Falcon-7B モデルをテストしました（**図 5**)。微調整されていないモデルでは、質問を理解して一貫した返答を定式化することが困難であることが分かります。これは、微調整によってモデルの理解力と全体的な返答の品質が向上したことを示す証拠です。改善の度合いを定量化するには因果関係言語モデリングのベンチマークを実行する必要がありますが、本題から外れるためここでは取り上げません。

```
Result:  Can you tell me three fun facts about space?
- What is the most important thing you have learned about space?
- What is the most important thing you have learned about yourself?
- What is the most important thing you have learned about your classmates?
- What is the most important thing you have learned about your teacher?
- What is the most important thing you have learned about yourself as a student?
- What is the most important thing you have learned about your classmates as
```

図 5. 微調整されていない Falcon-7B からの返答

要点と考察

これまでリリースされた中で最も強力な「真のオープンソース」LLM の 1 つの、微調整されたバージョンを利用できるようになりました。Hugging Face* の API、インテルの高速化された AI ツール、CPU ハードウェアのアクセシビリティー、および Falcon のオープンソース・ライセンスを組み合わせたこの実装は、さまざまな企業や AI アプリケーション開発者にとってアクセス可能なオプションとなります。

今回の目標は、パフォーマンスを分析することではなく、このワークロードを有効にすることであったため、ハードウェアのパフォーマンスと因果関係モデリングのメトリックは省略しました。開発者には、トレーニングのパフォーマンスを向上するため、Habana Gaudi*-1 および Gaudi*-2 アクセラレーター上でハイパーパラメーターの最適化、トランスフォーマー向けインテル® エクステンション（英語）、インテル® ニューラル・コンプレッサー（英語）、パラメーター効率に優れた微調整（PEFT）、LLM の低ランク適応（LoRA）、および Falcon の微調整を使用して、このワークフローの最適化を検討することを推奨します。