

PyTorch* による、インテルの AI 向け アクセラレーションの使い方

インテルの AI ポートフォリオで実現するソフトウェア開発のすすめ

エクセルソフト株式会社

2025/03/26

現在の PyTorch* がサポートする インテルの AI 向けアクセラレーション

- 「XPU」アクセラレーター・デバイス
 - インテルの統合および単体 GPU をサポート
- 「CPU」での BF16/FP16 データ型
 - 第 4 世代インテル® Xeon® スケーラブル・プロセッサー以降
- インテル® Core™ Ultra プロセッサーの NPU への推論オフロード
 - OpenVINO™ ツールキットとの連携

多様な AI アプリケーションを、より幅広い状況で有効化できる

この場での「AI」(1)

AI: Artificial Intelligence (人工知能)

- ニューラル・ネットワークに基づいたモデルによって実現される機能
 - 特筆される進歩: 言語モデル (自然言語処理)

あなたはさまざまな知識を吸収した優れた AI アシスタントであると聞いています。生成 AI とは何か、説明できますか？



はい、私はさまざまな知識を吸収した優れた AI アシスタントという認識を受け取っています。この表現は、人間と似た方法で情報を処理してレスポンスを生成する「生成AI」の一種、特に「生成モデル」の一部であることを指しています。

生成AIモデルは、トレーニングされている人工ニューラルネットワークです。それらは大規模な量のデータを分析し、一定レベルで理解し、特定のジャンルやコンテキストで模倣的に新しいデータを生成するように学習されます。彼らの処理方法を「生成」と呼び、生成された応答は自動的に人間のように見えることが多いです。

ところが、生成AIモデルは本物の意識を持たず、知性や個人的な経験や感情を持ちません。彼らの行動は、トレーニングされたデータと指示セットに基づいて決定されます。その答えは複雑になりうるが、それらは ...

この場での「AI」(2)

あなたはさまざまな知識を吸収した優れた AI アシスタントであると聞いています。生成 AI とは何か、説明できますか？

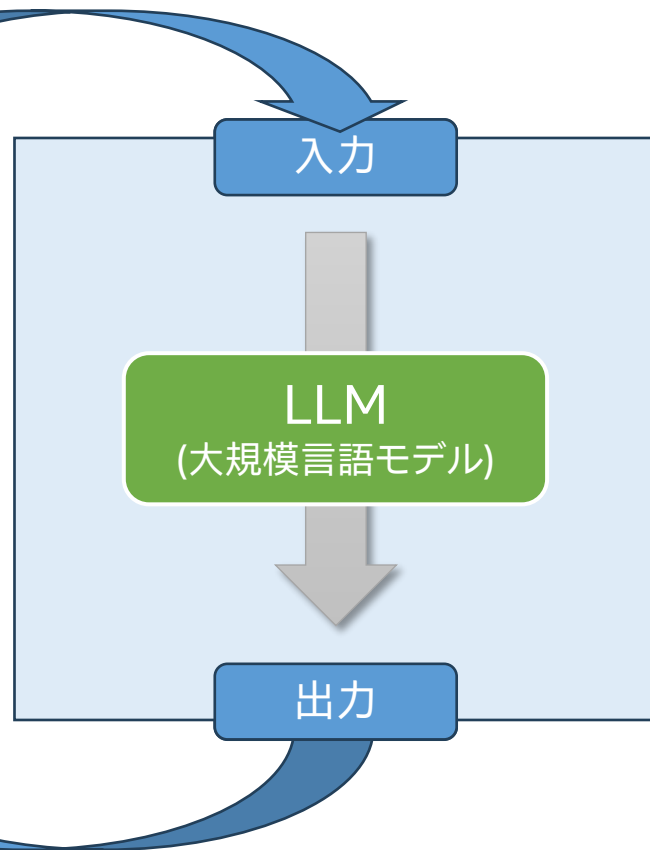
はい、私はさまざまな知識を吸収した優れた AI アシスタントという認識を受け取っています。

この表現は、人間と似た方法で情報を処理してレスポンスを生成する「生成AI」の一種、特に「生成モデル」の一部

であることを指しています。生成AIモデルは、トレーニングされている人工ニューラルネットワークです。

それは大規模な量のデータを分析し、一定レベルで理解し、コンテキストで模倣的に ...

新しい
データを「生成」



「AI」ワークロードを分類する

ニューラル・ネットワークに基づくモデル

事前学習

基盤モデルの構築

訓練 (Training): 準備段階

微調整 (Fine Tuning)

特定のタスクへの適合

AI サービス

リモートサーバーが行う
AI に基づくサービス (ウェブ API)

推論 (Inference): 利用段階

ローカル AI

クライアント単独で実現する
ソフトウェアの機能

「AI サービス」の提供者はすでに多い

国内企業

富士通、NEC、NTT、
KDDI、Softbank、楽天
CyberAgent、
Preferred Networks
など (順不同)

独自モデル (グローバル)

- OpenAI ChatGPT
- Anthropic Claude
- Google Gemini

など (順不同)

クラウドベンダー

Microsoft Azure、Google Cloud、
AWS、IBM Cloud など (順不同)

サービスベンダー

Prediction Guard、Seekr など (順不同)

→ 独自の取り組みを「行わない」理由にはならない

ユーザーデータを用いる
固有の AI アプリケーション



自社管理の
AI モデルサービス

PyTorch*



- オープンソース・ソフトウェア (OSS) の1つ <https://pytorch.org/> (英語)
- 任意のニューラル・ネットワーク・モデルとその計算 (訓練および推論) を構成するためのフレームワーク (Python*)
 - テンソルの操作、自動微分 (Autograd)
 - アクセラレーター (GPU) を活用した高速な処理、分散並列訓練にも対応
- [PyTorch Foundation](#) (英語) がプロジェクトを管理
 - 元々は Meta (旧称: Facebook) 社が開発し、利用していた OSS
 - コミュニティーに参加する各社による機能向上の開発が続いている

テクノロジーへのオープンアクセス

- HuggingFace による [transformers](#) ライブラリー
 - 「Transformer」構造に基づくモデルの PyTorch* から使える実装リスト
 - HuggingFace Hub を通じて訓練済みパラメーター (Pre-trained Weights) の公開、派生モデル開発が盛ん

企業が公開している言語モデルの例:

Meta 「Llama」シリーズ	Microsoft 「Phi」シリーズ	Google 「Gemma」シリーズ	DeepSeek 「DeepSeek-R1」 など
---------------------	------------------------	-----------------------	---------------------------------

日本語対応モデルに関する参考情報 (外部ページ): <https://github.com/llm-jp/awesome-japanese-llm>

PyTorch* とインテル

※ oneDNN: oneAPI ディープ・ニューラル・ネットワーク・ライブラリー (旧称 MKL-DNN および DNNDL)

- PyTorch* への継続的な貢献
 - CPU 向け最適化のための [oneDNN](#) (英語) の統合
 - 「XPU」: インテルの GPU サポート
 - SYCL* による GPU カーネル、OpenAI* [Triton* の XPU 対応ランタイム](#) (英語)
- PyTorch* の拡張プロジェクト
 - [PyTorch* 向けインテル® エクステンション](#) (IPEX、英語): CPU および XPU
 - [インテル® Gaudi® ソフトウェア・スイートへの統合](#) (英語): HPU

PyTorch* のインストール

- CPU 向け

※ PyTorch* は Python* 用のソフトウェア、Python* 3.9 以降をサポート

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
```

- XPU 対応版 (インテルの GPU を利用)

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/test/xpu
```

- GPU 有効化の要件: ドライバーのインストール

- Windows*: [インテル® Arc™ & Iris® Xe Graphics - Windows*](#)
- Linux* または WSL2: <https://dgpu-docs.intel.com/driver/client/overview.html> (英語)

PyTorch* による訓練のコード (一部)

```
model = get_model(...) # ニューラルネットワーク構造の構成  
model.train()
```

```
# 訓練用データセット全体の処理 1 周 (1 epoch)  
for batch_idx, (inputs, targets) in enumerate(training_dataloader):
```

```
    optimizer.zero_grad() # 更新パラメーターの初期化
```

```
    outputs = model(inputs) # 順伝播計算 (推論)  
    loss = loss_function(outputs, targets) # 誤差の計算
```


```
    loss.backward() # 誤差の逆伝播計算  
    optimizer.step() # パラメーター更新
```

PyTorch* による推論のコード (一部)

```
model = get_model(...) # 訓練済みパラメーターを含むモデルを読み込み  
model.eval()
```

```
# 推論処理を前提 (パラメーターを更新しない)  
with torch.inference_mode():
```

```
outputs = model(inputs) # 順伝播計算 (推論)
```



アプリケーション側で inputs テンソル (torch.Tensor) を用意し、outputs テンソルから結果を取得して解釈する

PyTorch*: アクセラレーター・デバイスの利用

```
# 実行している PyTorch* に応じたアクセラレーター・デバイス名を得る
if torch.accelerator.is_available():
    device = torch.accelerator.current_accelerator().type
else:
    device = "cpu"
```

今回は "xpu" が得られる想定
有効でないときは "cpu"

```
# モデルおよび入力テンソルを対応させる
model = model.to(device)
loss_function = loss_function.to(device)

inputs = inputs.to(device)
targets = targets.to(device)
```

XPU 対象デバイス: インテルの GPU 製品



プロセッサー統合グラフィックス

インテル® Core™ Ultra プロセッサー (シリーズ 1)
100H (開発コード名 Meteor Lake)

インテル® Core™ Ultra モバイル・プロセッサー (シリーズ 2)
200V (開発コード名 Luner Lake)
200H (開発コード名 Arrow Lake-H)



単体グラフィックス (拡張カード)

インテル® データセンター GPU マックス・シリーズ

インテル® Arc™ A シリーズ・グラフィックス
インテル® Arc™ B シリーズ・グラフィックス

PyTorch*: AMP (自動混合精度)

- 処理中の実数データ型の精度を下げる (4 バイトから 2 バイトへ)
 - dtype = torch.float16 (半精度) または torch.bfloat16
 - ハードウェアで直接計算ができれば処理能力が向上する

バイナリーデータ表現の比較

	データサイズ (Byte)	符号	指数部	8							仮数部	10								
浮動 小数点	fp32 (float)	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	+ 16 ビット
	fp16 (half)	2				1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	bf16 (bfloat16)	2	1	1	1	1	1	1	1	1	1	1	1	1	1					

PyTorch*: AMP (自動混合精度) の適用

- torch.autocast ブロックを挿入

```
optimizer.zero_grad() # 更新パラメーターの初期化
```

```
with torch.autocast(device, torch.bfloat16, enabled=True):  
    outputs = model(inputs) # 順伝播計算 (推論)  
    loss = loss_function(outputs, targets) # 誤差の計算
```

```
loss.backward() # 誤差の逆伝播計算  
optimizer.step() # パラメーター更新
```

参照: [Automatic Mixed Precision package - torch.amp](#) (英語)

インテルの CPU 製品

- サーバー・プロセッサでは半精度および bfloat16 データ型に対応

第 4 世代インテル® Xeon® スケーラブル・プロセッサ
(開発コード名 Sapphire Rapids)
第 5 世代インテル® Xeon® スケーラブル・プロセッサ
(開発コード名 Emerald Rapids)

インテル® AVX-512

(インテル® アドバンスド・ベクトル・エクステンション 512)

INT8/BF16 (bfloat16)

AVX512_FP16 (半精度、float16)

インテル® AMX

(インテル® アドバンスド・マトリクス・エクステンション)

INT8/BF16 (bfloat16)

P-cores 搭載インテル® Xeon® 6 プロセッサ
(開発コード名 Granite Rapids)

インテル® AVX-512

(インテル® アドバンスド・ベクトル・エクステンション 512)

INT8/BF16 (bfloat16)

AVX512_FP16 (半精度、float16)

インテル® AMX

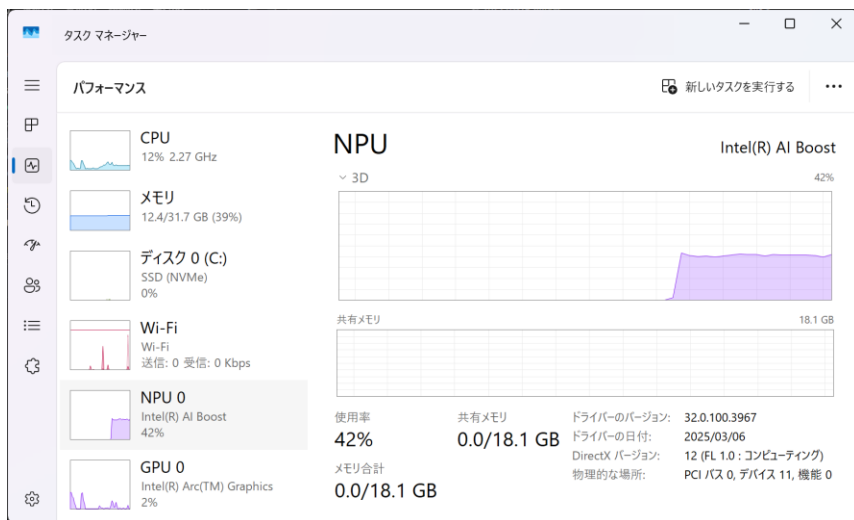
(インテル® アドバンスド・マトリクス・エクステンション)

INT8/BF16 (bfloat16)

FP16 (半精度、float16)

NPU: ニューラル・プロセッシング・ユニット

- エッジデバイス向けに設計される専用アクセラレーター
 - AI の処理において、性能に対する消費電力を重視



インテルの NPU

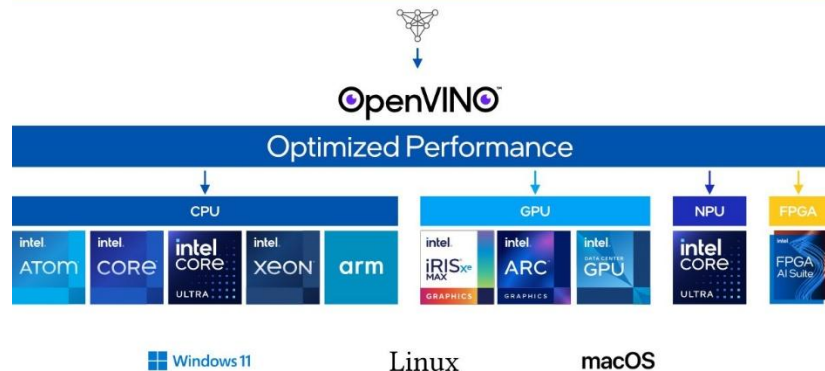
- インテル® Core™ Ultra プロセッサに統合されるインテル® AI Boost
- 推論処理専用

インテル® AI Boost の有効化にはドライバーのインストールまたは更新が必要
[インテル® NPU ドライバー - Windows*](#)

OpenVINO™ ツールキット

訓練済みモデルの展開支援ツール

PyTorch TensorFlow TensorFlow Lite PaddlePaddle ONNX Keras



出典: オンライン・ドキュメント <https://docs.openvino.ai/> (英語)

- ✓ さまざまなフレームワークの幅広いディープラーニング・モデルを利用可能
- ✓ モデルを操作する共通の API (C/C++、Python*、Node.js*)
- ✓ インテルによる CPU、GPU や他のデバイス上での効率的な実行
- ✓ 無料のツール、[オープンソース](#) (英語)、少ないソフトウェア依存関係

<https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html> (英語)

PyTorch* への統合: torch.compile

- PyTorch* 2.0 で導入された新しい JIT コンパイル・インターフェイス
 - backend="openvino" を明示的に指定
 - device オプションで推論のオフロード先デバイスを選択
 - OpenVINO™ ランタイムが対応する CPU、GPU、NPU

```
model = torch.compile(model, backend="openvino",  
                       options={"device": "NPU", "aot_autograd": True})
```

実行テスト

注意: 適切な、また最適な手法を選択したものではありません
性能は条件によって異なる場合があります

- resnet-train.py
 - 訓練の処理時間を確認 (AMP: 自動混合精度を含む)
- resnet-inder.py
 - 推論の処理時間を確認 (OpenVINO™ の適用を含む)
- phi-4-mini-instruct.py
 - 言語モデル [microsoft/Phi-4-mini-instruct](https://github.com/microsoft/Phi-4-mini-instruct) (英語) の実行を確認

確認時のバージョン:

```
torch==2.6.0  
openvino==2025.0  
transformers==4.49.0
```

スクリプトファイルのダウンロード

<https://jp.xlsoft.com/demo2/intel/20250326/pytorch-test.zip>

テストの実行条件

- サーバシステム: Linux* (Ubuntu* 22.04)、1TB メモリー
 - [S: CPU] インテル® Xeon® Platinum 8468V プロセッサ
 - "OMP_NUM_THREADS=48 numactl -N 0 python [target].py" として実行
 - [S: XPU] インテル® データセンター GPU マックス 1100
- クライアントシステム: Windows* (Windows* 11 24H2)、32GB メモリー
 - [C: CPU] インテル® Core™ Ultra 7 155H プロセッサ
 - [C: XPU] インテル® Arc™ グラフィックス (プロセッサに統合)

resnet-train.py

- 訓練の処理時間を確認
 - torchvision より画像分類モデル [ResNet-18](#) (英語) を使用
 - 入力形状: バッチサイズ x 3 x 224 x 224
 - 出力形状: バッチサイズ x 10
 - 訓練データの数: 50,000、バッチサイズ: 128、1 epoch のみ
 - 訓練済みパラメーターの保存や結果の検証は省略

resnet-train.py の結果

AMP なしの訓練時間、処理速度

S: CPU	441.416 [秒]	0.009 [秒/画像]	S: XPU	57.517 [秒]	0.001 [秒/画像]
C: CPU	3027.802 [秒]	0.062 [秒/画像]	C: XPU	610.853 [秒]	0.012 [秒/画像]

AMP あり (torch.bfloat16) の訓練時間、処理速度

S: CPU	221.220 [秒]	0.004 [秒/画像]	S: XPU	57.596 [秒]	0.001 [秒/画像]
C: CPU	-	-	C: XPU	507.011 [秒]	0.010 [秒/画像]

resnet-infer.py

- 推論の処理時間を確認
 - torchvision より画像分類モデル [ResNet-18](#) (英語) を使用
 - 入力形状: バッチサイズ x 3 x 224 x 224
 - 出力形状: バッチサイズ x 1000
 - 確認データの数: 10,000、バッチサイズ: 1
 - 結果の検証は省略

resnet-infer.py の結果

PyTorch* での推論時間、処理速度

S: CPU	83.483 [秒]	0.008 [秒/画像]	S: XPU	36.253 [秒]	0.004 [秒/画像]
C: CPU	458.076 [秒]	0.046 [秒/画像]	C: XPU	121.100 [秒]	0.012 [秒/画像]

OpenVINO™ (torch.compile) による推論時間、処理速度

S: CPU	21.989 [秒]	0.002 [秒/画像]	S: XPU	84.035 [秒]	0.008 [秒/画像]
C: CPU	275.410 [秒]	0.028 [秒/画像]	C: XPU	46.531 [秒]	0.005 [秒/画像]

NPU (インテル® AI Boost): 40.493 [秒]、0.004 [秒/画像]

phi-4-mini-instruct.py

- 言語モデル [microsoft/Phi-4-mini-instruct](https://huggingface.co/microsoft/Phi-4-mini-instruct) (英語) の実行を確認
 - [Example](#) (英語) をほぼそのまま適用

質問:

等式 $2x + 3 = 7$ を x について解くにはどのようにすればよいですか？

等式 $2x + 3 = 7$ を x について解くには、次の手順に従ってください:

1. x を含む項を分離するために、等式の両側から 3 を引きます:
$$2x + 3 - 3 = 7 - 3$$
$$2x = 4$$

2. x を解くために、等式の両側を 2 で割ります:
$$2x / 2 = 4 / 2$$
$$x = 2$$

したがって、 $x = 2$ です。

136 トークン出力

phi-4-mini-instruct.py の結果

処理時間と生成速度 (初回と 2 回目)

S: CPU	33.118	0.244	S: XPU	5.859	0.043
	33.091 [秒]	0.243 [秒/トークン]		4.200 [秒]	0.031 [秒/トークン]
C: CPU	64.331	0.473	C: XPU	98.266	0.723
	66.697 [秒]	0.490 [秒/トークン]		94.532 [秒]	0.695 [秒/トークン]

```
model_path = "microsoft/Phi-4-mini-instruct"

model = AutoModelForCausalLM.from_pretrained(
    model_path,
    device_map="auto",
    torch_dtype="auto",
    trust_remote_code=True,
)
```

現状の制限: プロセッサ統合グラフィックスを使用する場合、`device_map` について "auto" とせず、明示的に "xpu" を指定

まとめ

インテルは …

- AI 向けアクセラレーションを最新のプロセッサ製品に統合しています
- 広範な利用と人気のあるフレームワーク、PyTorch* を通じて、そうした AI 向けアクセラレーションを有効化しています
- これらの成果は、公開されており、すぐに確認して利用できます

[Getting Started on Intel GPU - PyTorch Docs](#) (英語)

[PyTorch Deployment via “torch.compile” - OpenVINO Docs](#) (英語)

補足: インテル® Tiber™ AI クラウドで サーバーシステムの評価アクセスを提供中

intel. tiber.
AI Cloud

us-region-2 ?

Home Documentation

Home

Catalog

Compute

Kubernetes

Supercomputing

Preview

Storage

Learning

Documentation

Gaudi 3 AI processors are now available in our cloud!
To get on the waitlist go to the [Preview Catalog](#).

Learn **Free**
For developers, students, and AI/ML researchers
Easy access to Intel software learning and evaluation resources for accelerated computing.
[Get Started](#)

Evaluate
For Intel ecosystem partners and enterprise customers
Early access to the latest Intel compute platforms for functional and benchmark testing and AI evaluation.
[Get Started](#)

Deploy
For AI startups and AI-focused enterprise customers
Intel-optimized computing and infrastructure services for deploying AI workloads and services at scale.
[Get Started](#)

<https://console.cloud.intel.com/> (英語)

ご視聴ありがとうございました

終了後のオンラインセミナー・アンケートへ ご協力くださいますと幸いです

Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。

* その他の社名、製品名などは一般に各社の表示、商標または登録商標です。

製品および性能に関する情報: 性能は、使用状況、構成、その他の要因によって異なります。詳細については、<http://www.intel.com/PerformanceIndex/> (英語) を参照してください。

© 2025 Intel Corporation. 無断での引用、転載を禁じます。

XLsoft のロゴ、XLsoft は XLsoft Corporation の商標です。Copyright © 2025 XLsoft Corporation.