インテル® Tiber™ デベロッパー・クラウドで PyTorch* 2.4 を使って AI ワークロードを高速化

この記事は、インテルのブログで公開されている「Accelerate AI Workloads with a PyTorch* 2.4 Workshop on the Intel® Tiber™ Developer Clouds」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。

インテル® Tiber™ デベロッパー・クラウドとは?

インテル® Tiber™ デベロッパー・クラウドは、クラウドベースのプラットフォームであり、開発者、AI/ML 研究者、エコシステム・パートナー、AI スタートアップ、およびエンタープライズ顧客向けに、最先端のインテルのハードウェアとソフトウェア・ソリューションへのアクセスを提供することで、AI とハイパフォーマンス・コンピューティング・アプリケーションを低コスト、低オーバーヘッドで構築、テスト、実行、最適化できるようにします。インテル® Tiber™ デベロッパー・クラウドは、開発者に、インテルの CPU、GPU、および AI PC 上で小規模または大規模なワークロードを使用してイノベーションを行う簡単なパスを提供します。これには、oneAPI などの AI 向けに最適化されたソフトウェア (英語) へのアクセスが含まれます。

プラットフォームとハードウェアの機能をテストし、インテルで何ができるのかを知りたい開発者やエンタープライズ顧客向けには、無料の共有環境と Jupyter* Notebook が用意されています。

この記事では、インテル® GPU 上で PyTorch* 2.4 を体験できるワークショップについて説明します。XPU を活用してパフォーマンスを加速し、インテル® Tiber™ デベロッパー・クラウドで生成 AI ワークロードを開発してデプロイする方法を学ぶことができます。

PyTorch* 2.4

PyTorch* は、コンピューター・ビジョンや自然言語処理 (NLP) アプリケーションでよく使用される、ディープ・ニューラル・ネットワークの構築とトレーニング向けの人気あるオープンソース・ディープラーニング・フレームワークです。PyTorch* 2.4 (英語) は、インテル® データセンター GPU マックス・シリーズの初期サポートを提供し、インテル® ハードウェア上で AI ワークロードをさらに加速する、SYCL* ソフトウェア・スタックと Unified Acceleration Foundation* (UXL) (英語) のマルチベンダー・ソフトウェア・エコシステムを活用しています。

このサポートにより、最小限のコーディング作業でインテル® GPU 上でワークロードを実行およびデプロイできるようになり、ユビキタス・ハードウェアへの PyTorch* の展開が容易になります。これにより、異なるハードウェア・バックエンドの統合も容易となり、ワークロードの開発とデプロイの可能性が拡大します。

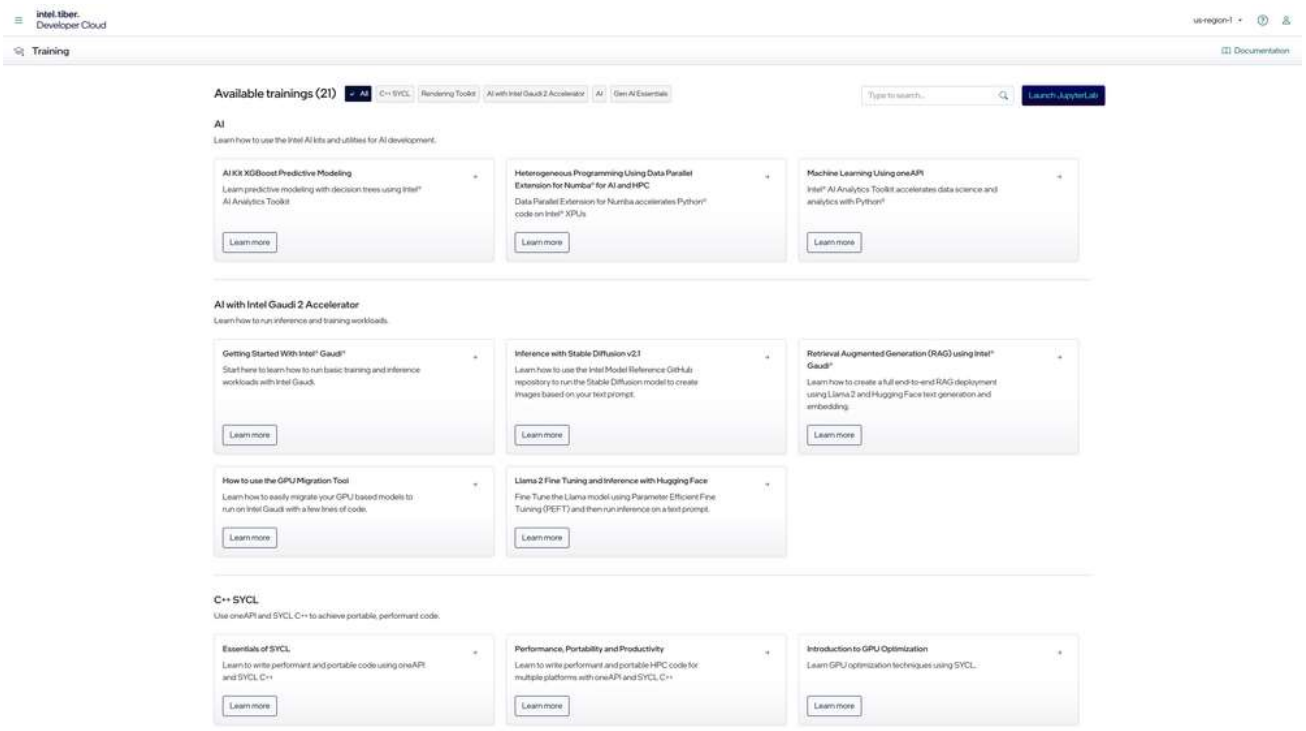
GPU サポートとより優れたパフォーマンスを得るため、PyTorch* 向けインテル® エクステンション (英語) も利用できます。これは、インテル® プロセッサー上で大規模言語モデル (LLM) や生成 AI (GenAI) など、さまざまなアプリケーションのディープラーニング・トレーニングと推論のパフォーマンスを最適化します。

インテル® GPU 上の PyTorch* ワークショップ

このトレーニングでは、インテル® GPU で PyTorch* 2.4 を使用する方法を学びます。PyTorch* の機能を活用しながら、テンソル操作の例の演習、サンプル・ワークロードの実行、パフォーマンスを向上するモデルの最適化を行います。

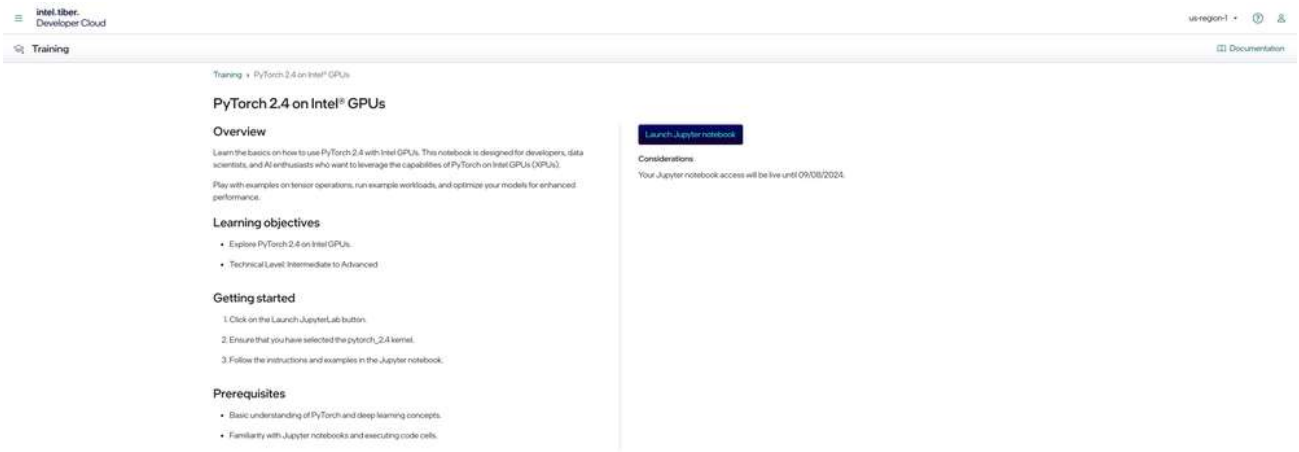
最初に、cloud.intel.com (英語) にサインインします。まだアカウントをお持ちでない場合は、こちらの手順に従ってアカウント登録を行ってください。

サインインしたら、コンソールの左上のメニューアイコンをクリックして **Training** (英語) を選択します。

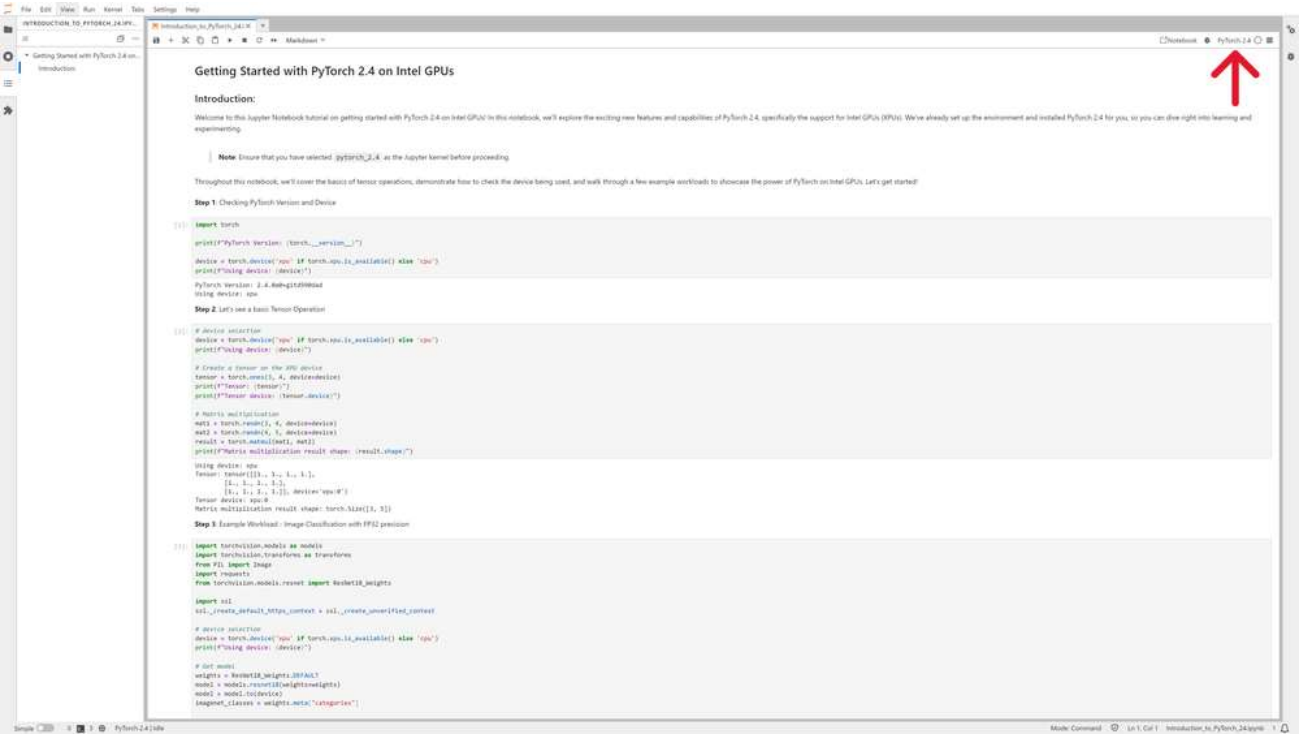


Training ページには、インテル® Tiber™ デベロッパー・クラウドで利用できる JupyterLab* ワークショップがいくつか用意されています。AI、インテル® Gaudi® 2 AI アクセラレーターを使用した AI、インテル® データセンター GPU マックス・シリーズを使用した AI、C++ SYCL*、レンダリング・ツールキットなどがあります。

このチュートリアルでは、**AI with Max Series GPU** トレーニングの PyTorch* on Intel GPUs ワークショップ (英語) を使用します。**[Launch]** をクリックして Jupyter* Notebook にアクセスします。



Jupyter* Notebook トレーニングが起動したら、右上の **Kernel** が PyTorch* 2.4 に設定されていることを確認します。



セルを実行し、ノートブックの手順に従って、インテル® GPU 上で PyTorch* の利点を示すいくつかのサンプル・ワークロードを確認してください。以下は、トレーニング・ノートブックに記載された手順の詳しい説明です。

注: ここでは参考用にコードの一部のみを表示しており、実行に必要な重要な行が含まれていません。完全なコードは、Jupyter* Notebook を参照してください。

ステップ 1: PyTorch* のバージョンとデバイスの確認

```
print(f"PyTorch Version: {torch.__version__}")
device = torch.device(&apos;xpu&apos; if torch.xpu.is_available() else
&apos;cpu&apos;)
print(f"Using device: {device}")
```

使用している PyTorch* のバージョン (PyTorch* 2.4) とインテル® Tiber™ デベロッパー・クラウド上で XPU または CPU のどちらが使用されているかを確認します。出力には、プラットフォーム上で動作している XPU デバイスが表示されるはずです。

ステップ 2: 基本的なテンソル操作

```
mat1 = torch.randn(3, 4, device=device)
mat2 = torch.randn(4, 5, device=device)
result = torch.matmul(mat1, mat2)
```

XPU デバイス上にテンソルを作成し、単純な行列乗算を実行します。結果のテンソルの形状が出力されます。PyTorch* を使用すると、GPU やその他の AI アクセラレーター上でテンソル操作を実行し、モデルの入力、出力、パラメーターを効率良くエンコードできます。

ステップ 3: サンプル・ワークロード - FP32 精度での画像分類

```
# モデルを取得
weights = ResNet18_Weights.DEFAULT
model = models.resnet18(weights=weights)
model = model.to(device)
imagenet_classes = weights.meta["categories"]

# 入力画像を準備
image_url = https://upload.wikimedia.org/wikipedia/commons/thumb/4/4d/Cat_
November_2010-1a.jpg/1200px-Cat_November_2010-1a.jpg
input_image = Image.open(requests.get(image_url, stream=True).raw)
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0).to(device)

# 推論
model = model.eval()
with torch.no_grad():
    output = model(input_batch)

_, predicted = torch.max(output, 1)
class_index = predicted.item()
class_label = imagenet_classes[class_index]
```

この例では、サンプル ResNet18 モデルを取得し、Wikipedia* からサンプル入力画像を準備して、モデルを推論します。ResNet18 モデルは、画像認識のディープ残差学習に使用され、ImageNet データセットで事前トレーニングされています。コード出力には、モデルによって生成された予測クラスラベルが表示されます。ResNet18 モデルが正しく動作し、ファインチューニングの必要がない場合、クラスラベルは「Egyptian cat (エジプトの猫)」になります。

ステップ 4: サンプル・ワークロード – 感情分析推論

シンプルな感情分析モデルを定義（トレーニング済みのモデルが用意されていることを想定していますが、ここではこのモデルを例として使用します）class SentimentModel(nn.Module):
```
    def __init__(self, vocab_size, embed_dim, hidden_dim, output_dim):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.lstm = nn.LSTM(embed_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = self.embedding(x)
        _, (hidden, _) = self.lstm(x)
        out = self.fc(hidden.squeeze(0))
        return out

vocab_size = 10000
embed_dim = 100
hidden_dim = 512
output_dim = 2

model = SentimentModel(vocab_size, embed_dim, hidden_dim, output_dim)
model = model.to(device)
print(f"\nModel before compilation: \n{model}\n")
model = torch.compile(model)  # モデルをコンパイル
print("-"*72)
print(f"\nModel after compilation: \n{model}")
```

この例では、長短期記憶 (LSTM) モデルが、シンプルな感情分析モデルを定義しています。LSTM モデルは、シーケンシャル・データを処理し、非表示状態を保持できる再帰型ニューラル・ネットワーク (RNN) です。モデルのパラメーターは、torch.compile でコンパイルする前と後に表示されます。結果の感情スコアも表示されます。

ステップ 5: 転移学習 – 自動混合精度を使用したビジョン

```
# 評価
model = model.eval()
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        images, labels = data[0].to(device), data[1].to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
print(f"Accuracy on test images: {100 * correct / total:.2f}%")
```

`torch.float16` または `torch.bfloat16` を使用して ResNet18 ビジョンモデルで転移学習を行い、ストレージ要件を軽減し、自動混合精度のモデル速度を向上します。まず、多くの画像で構成される CIFAR10 データセットを使用して、ResNet18 モデルのトレーニング・セットとテストセットを作成します。モデルがトレーニングされ、テスト画像の精度に基づいて評価されます。トレーニングがすべての反復を完了すると、精度のパーセンテージが出力されます。この例では、高い精度が得られます。これは、適切なファインチューニングにより、さらに向上させることができます。

ノートブックの手順を実行すると、基本的なテンソル操作、デバイスチェック、およびいくつかのサンプル AI モデルのトレーニングと推論など、インテル® GPU 上で PyTorch* 2.4 の機能を理解できます。

インテル® Tiber™ AI クラウド (英語) で最新のシリコン・ハードウェアと最適化されたソフトウェアにアクセスし、革新的な AI プロジェクトの開発と強化に役立ててください。インテルの AI ツールとフレームワークの最適化 (英語) をチェックし、インテルの AI ソフトウェア・ポートフォリオ (英語) の基盤となる統一されたオープンな標準ベースの oneAPI プログラミング・モデルについて学びましょう。また、業界をリードする独立系ソフトウェア・ベンダー (ISV)、システム・インテグレーター (SIer)、OEM、エンタープライズ・ユーザーとのコラボレーション (英語) が、AI の導入をどのように加速させるかをご覧ください。

エクセルソフトでは、インテル® Tiber™ AI クラウドの評価用クーポンを配布しています。ご興味のある方は、こちらのフォームよりお問い合わせください。

関連情報

- インテルの AI 開発ツールとリソース (英語)
- oneAPI 統一プログラミング・モデル
- インテル® Tiber™ AI クラウド
- 記事: インテル® Tiber™ デベロッパー・クラウドで AI へのアクセスが容易に
- 記事: インテル® Tiber™ デベロッパー・クラウドで生成 AI と Jupyter* Notebooks を試してみる (英語)
- 記事: インテル® Tiber™ デベロッパー・クラウドで AI に PyTorch* と TensorFlow* を活用する (英語)