Stable Diffusion ControlNet パイプラインを使用して LoRA の重みを有効にする

この記事は、Medium に公開されている「Enable LoRA weights with Stable Diffusion Controlnet Pipeline」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。

Low-Rank Adaptation (LoRA) は、ディフューザーと大規模言語モデル (LLM) の微調整の問題に対処するために導入された新しい手法です。Stable Diffusion の微調整では、潜在画像表現の cross-attention 層に LoRA を適用できます。モデルの微調整の基本的な概念と手法を理解するには、Hugging Face のディフューザー (英語) を参照してください。

この記事は、OpenVINO™ の最適化を使用して Stable Diffusion + ControlNet のパイプラインを構築する方法を紹介し、Stable Diffusion の Unet モデルで LoRA の重みを有効にして異なるスタイルの画像を生成することを目的としています。デモのソースコードのベースは、https://github.com/FionaZZ92/OpenVINO_sample/tree/master/SD_controlnet (英語) です。

Stable Diffusion ControlNet パイプライン

ステップ 1: 環境の準備

まず、以下の手順に従って開発環境を準備してください。ランタイムのエクスペリエンスが向上するように、Hugging Face のダウンロード・モデルを使用します。ここでは、canny イメージタスクに ControlNet を選択しました。

```
$ mkdir ControlNet && cd ControlNet

$ wget
https://huggingface.co/lllyasviel/ControlNet/resolve/main/annotator/ckpts/bod
y_pose_model.pth

$ conda create -n SD python==3.10
$ conda activate SD

$ pip install opencv-contrib-python
$ pip install -q "diffusers>=0.14.0"
"git+https://github.com/huggingface/accelerate.git" controlnet-aux gradio
$ pip install openvino openvino-dev onnx
$ pip install torch==1.13.1 #important

$ git lfs install
$ git clone https://huggingface.co/lllyasviel/sd-controlnet-canny
$ git clone https://huggingface.co/runwayml/stable-diffusion-v1-5
$ git clone https://huggingface.co/openai/clip-vit-large-patch14
```

```
$ wget
https://huggingface.co/takuma104/controlnet_dev/blob/main/gen_compare/control
_images/vermeer_512x512.png
```

* インストールされている torch のバージョンが 2.0 以上の場合、ディフューザーは `torch.nn.functional.scaled_dot_product_attention` を使用しますが、ONNX* は「Aten::scaled_dot_product_attention」の op 変換をサポートしていないことに注意してください。
「torch.onnx.export」によるモデル変換中のエラーを回避するため、torch のバージョン 1.13.1 を使用していることを確認してください。

ステップ 2: モデルの変換

デモでは 2 つのプログラムを提供します。モデルを OpenVINO™ IR に変換するには「get_model.py」を使用してください。次のようにコマンドを入力して、このスクリプトのオプションを確認してください。

```
$ python get_model.py -h
usage: get_model.py [-h] -b BATCH -sd SD_WEIGHTS [-lt LORA_TYPE] [-lw
LORA_WEIGHTS]

Options:
  -h, --help            Show this help message and exit.
  -b BATCH, --batch BATCH
                        Required. batch_size for solving single/multiple
prompt->image generation.
  -sd SD_WEIGHTS, --sd_weights SD_WEIGHTS
                        Specify the path of the stable diffusion model
  -lt LORA_TYPE, --lora_type LORA_TYPE
                        Specify the type of lora weights, you can choose
"safetensors" or "bin"
  -lw LORA_WEIGHTS, --lora_weights LORA_WEIGHTS
                        Add lora weights to Stable diffusion.
```

ここでは、複数のイメージを生成するために複数のバッチサイズを選択します。ビジョン生成の一般的なアプリケーションには、バッチの 2 つの概念があります。

1. `batch_size`: 入力プロンプトまたはネガティブプロンプトの長さを指定します。この方法は、N 個のプロンプトで N 個のイメージを生成するために使用します。
2. `num_images_per_prompt`: 各プロンプトが生成する画像の数を指定します。この方法は、1 つのプロンプトで M 個のイメージを生成するために使用します。

つまり、一般的なユーザー・アプリケーションでは、ディフューザーでこれら 2 つの属性を使用して、ランダムシード値を増やした N 個のプロンプトで N*M 個の画像を生成できます。例えば、基本シードが 42 で N(2)*M(2) 個の画像を生成する場合、実際の生成は次のようになります。

- N=1、M=1: prompt_list[0]、シード=42
- N=1、M=2: prompt_list[0]、シード=43
- N=2、M=1: prompt_list[1]、シード=42
- N=2、M=2: prompt_list[1]、シード=43

ここでは、デモンストレーションの簡単な例として N=2、M=1、つまり `— batch 2` を使用します。このスクリプトはデフォルトで静的形状モデルを生成します。N と M に異なる値を使用する場合は、`— dynamic` と指定してください。

```
$ python get_model.py -b 2 -sd stable-diffusion-v1-5/
```

現在のパスを確認し、以下のモデルが生成されていることを確認してください。スペースを節約するため、ほかの ONNX* ファイルは削除できます。

- controlnet-canny.<xml|bin>
- text_encoder.<xml|bin>
- unet_controlnet.<xml|bin>
- vae_decoder.<xml|bin>

* ローカルパスにすでに ONNX* または IR モデルが存在する場合、スクリプトは ONNX/IR を生成します。PyTorch* モデルを更新する場合、または異なる形状のモデルを生成したい場合は、既存の ONNX* および IR モデルを削除することを忘れないでください。

ステップ 3: ランタイム・パイプラインのテスト

デモプログラム `run_pipe.py` は、`diffusers.StableDiffusionControlNetPipeline` のソースを参照する Stable Diffusion ControlNet パイプラインを手動で構築します。

https://github.com/huggingface/diffusers/blob/main/src/diffusers/pipelines/controlnet/pipeline_controlnet.py (英語)

違いは、モデルの推論をインテル® CPU および GPU プラットフォームで高速化できるように、OpenVINO™ ランタイム API で 4 つのモデルの推論のパイプラインを簡素化していることです。

デフォルトの反復は 20、イメージの形状は 512*512、シードは 42、入力イメージとプロンプトは「Girl with Pearl Earring」です。テスト用にパイプラインの属性を調整またはカスタマイズできます。

```
$ python run_pipe.py
```

batch_size=2 の場合、生成される画像は次のようになります。

オリジナルの Stable Diffusion v1.5 + canny ControlNet で生成されたイメージ

Stable Diffusion で LoRA の重みを有効にする

通常の LoRA の重みには 2 つの種類があり、1 つは ` pytorch_lora_weights.bin`、もう 1 つは safetensors を使用します。ここでは、これらの 2 つの LoRA の重みを有効にする方法を説明します。

LoRA の重みを有効にするための主な考え方は、Stable Diffusion のオリジナルの Unet モデルに重みを追加し、LoRA の重みを維持したまま Unet の IR モデルをエクスポートすることです。

https://civitai.com/tag/lora（英語）にはさまざまな LoRA モデルがあります。ここでは例として、HuggingFace で公開されているモデルをいくつか選択しました。独自のモデルと置き換えることができます。

ステップ 4–1: pytorch_lora_weights.bin で LoRA を有効にする

このステップでは、`pipe.unet.load_attn_procs(…)` 関数を使用して Stable Diffusion の Unet モデルに LoRA の重みを追加する方法を説明します。この方法を使用すると、LoRA の重みは Stable Diffusion の Unet モデルの attention 層にロードされます。

```
$ git clone https://huggingface.co/TheUpperCaseGuy/finetune-lora-stable-diffusion
$ rm unet_controlnet.* unet_controlnet/unet_controlnet.onnx
$ python get_model.py -b 2 -sd stable-diffusion-v1-5/ -lt bin -lw finetune-lora-stable-diffusion/
```

* LoRA の重みを含む新しい IR を生成する前に、既存の Unet モデルを削除することを忘れないでください。

次に、パイプライン推論プログラムを実行して結果を確認します。

```
$ python run_pipe.py
```

ControlNet パイプラインを使用して LoRA の重みを追加した Stable Diffusion モデルでは、次のような画像が生成されます。

Stable Diffusion v1.5 + LoRA bin の重み + canny ControlNet

ステップ 4–2: safetensors 型の重みで LoRA を有効にする

このステップでは、`diffusers/scripts/convert_lora_safetensor_to_diffusers.py` を使用して Stable Diffusion の Unet モデルに LoRA の重みを追加する方法を説明します。ディフューザーは、safetensors 型の LoRA モデルを有効にして新しい Stable Diffusion モデルを生成するスクリプトを提供します。この方法では、新しく生成された Stable Diffusion モデルの重みのパスを LoRA に置き換える必要があります。`alpha` オプションの値を調整して、attention 層の `W = W0 + alpha * deltaW` の結合率を変更できます。

```
$ git clone https://huggingface.co/ntc-ai/fluffy-stable-diffusion-1.5-lora-trained-without-data
$ git clone https://github.com/huggingface/diffusers.git && cd diffusers

$ python scripts/convert_lora_safetensor_to_diffusers.py --base_model_path ../stable-diffusion-v1-5/ --checkpoint_path ../fluffy-stable-diffusion-1.5-lora-trained-without-data/fluffySingleWordConcept_v10.safetensors --dump_path ../stable-diffusion-v1-5-fluffy-lora --alpha=1.5
$ cd .. && rm unet_controlnet.* unet_controlnet/unet_controlnet.onnx
$ python get_model.py -b 2 -sd stable-diffusion-v1-5-fluffy-lora/ -lt safetensors
```

次に、パイプライン推論プログラムを実行して結果を確認します。

```
$ python run_pipe.py
```

ControlNet パイプラインを使用して LoRA の重みを追加した Stable Diffusion モデルでは、次のような画像が生成されます。

Stable Diffusion v1.5 + LoRA safetensors の重み + canny ControlNet

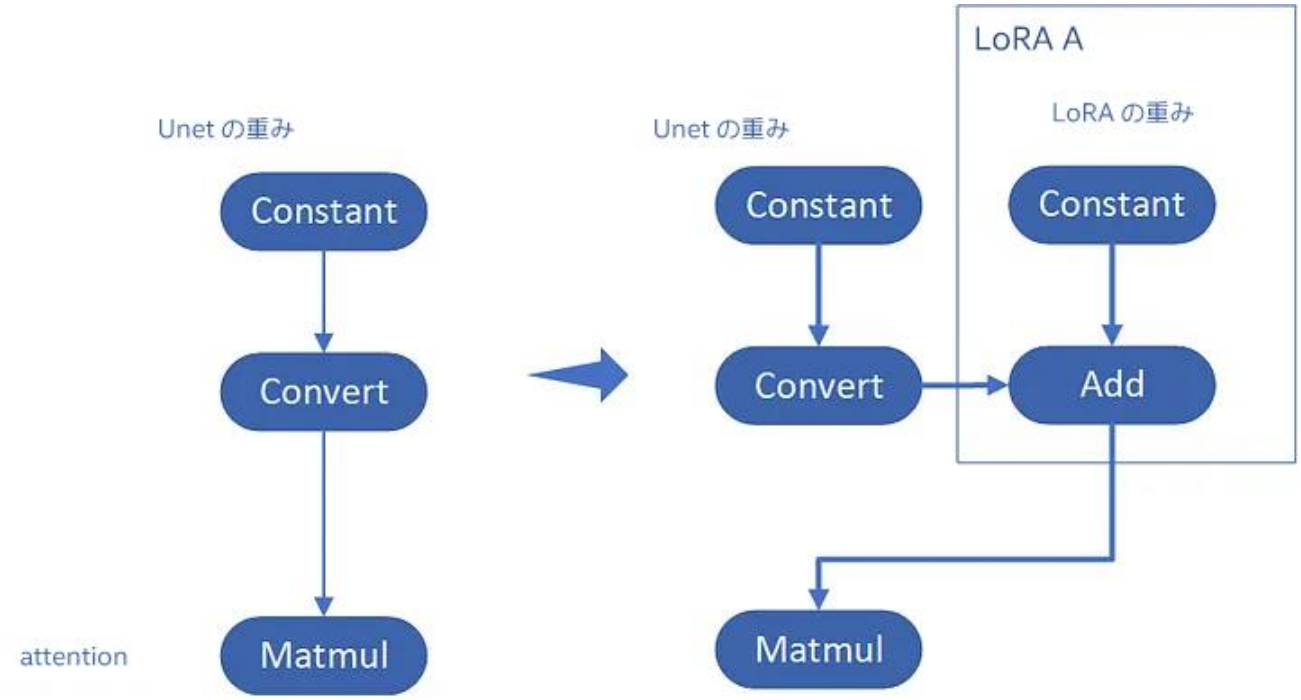
ステップ 4–3: MatcherPass でランタイム LoRA マージを有効にする

このステップでは、Unet または text_encoder モデルをコンパイルする前に、ランタイムに LoRA の重みを追加する方法を紹介します。同じ Unet/text_encoder 構造を再利用して画像スタイルを変更する、複数の異なる LoRA の重みを備えたクライアント・アプリケーションで役立ちます。

この方法は、safetensors ファイル内の LoRA の重みを抽出し、Unet モデル内の対応する重みを見つけて、LoRA の重みバイアスを挿入します。LoRA の重みを追加する一般的な方法は次のとおりです。

W = W0 + W_bias(alpha * torch.mm(lora_up, lora_down))

OpenVINO™ `opset10.add(W0, W_bias)` を使用して、Unet の attention の重みに Add 操作を挿入します。Unet モデルのオリジナルの attention の重みは `Const` op によりロードされ、共通の処理パスは `Const->Convert->Matmul->…` です。LoRA の重みを追加する場合は、計算した LoRA の重みバイアスを `Const->Convert->Add->Matmul->…` で挿入する必要があります。call_back() 関数で繰り返し `opset10.add()` を挿入する `openvino.runtime.passes.MatcherPass` を採用しています。



LoRA の重みを追加するモデル変換の方法

変換操作は、最初に opset.Add() を挿入します (次はデバイスでのモデルのコンパイル中)。グラフは定数の畳み込みを行い、Add 操作と次の Matmul 操作を組み合わせて、モデルのランタイムの推論を最適化します。これは、LoRA の重みをオリジナルのモデルにマージする効率の良い方法です。

実装のソースコードを確認すると、`InsertLoRA(MatcherPass)` と呼ばれる MatcherPass 関数の定義が見つかります。

```
class InsertLoRA(MatcherPass):
    def __init__(self,lora_dict_list):
        MatcherPass.__init__(self)
        self.model_changed = False

        param = WrapType("opset10.Convert")

        def callback(matcher: Matcher) -> bool:
            root = matcher.get_match_root()
            root_output = matcher.get_match_value()
            for y in lora_dict_list:
                if
root.get_friendly_name().replace('.','_').replace('_weight','') == y["name"]:
                    consumers = root_output.get_target_inputs()
                    lora_weights =
ops.constant(y["value"],Type.f32,name=y["name"])
                    add_lora =
ops.add(root,lora_weights,auto_broadcast='numpy')
```

```
                for consumer in consumers:
                    consumer.replace_source_output(add_lora.output(0))

                # Use new operation for additional matching
                self.register_new_node(add_lora)
            # Root node wasn't replaced or changed
            return False
        self.register_matcher(Matcher(param,"InsertLoRA"), callback)
```

`InsertLoRA(MatcherPass)` 関数は `manager.register_pass(InsertLoRA(lora_dict_list))` で登録され、`manager.run_passes(ov_unet)` で呼び出されます。このランタイム MatcherPass 操作の後、デバイスプラグインを使用してグラフをコンパイルします。

パイプライン推論プログラムを実行して結果を確認します。結果はステップ 4–2 と同じになります。

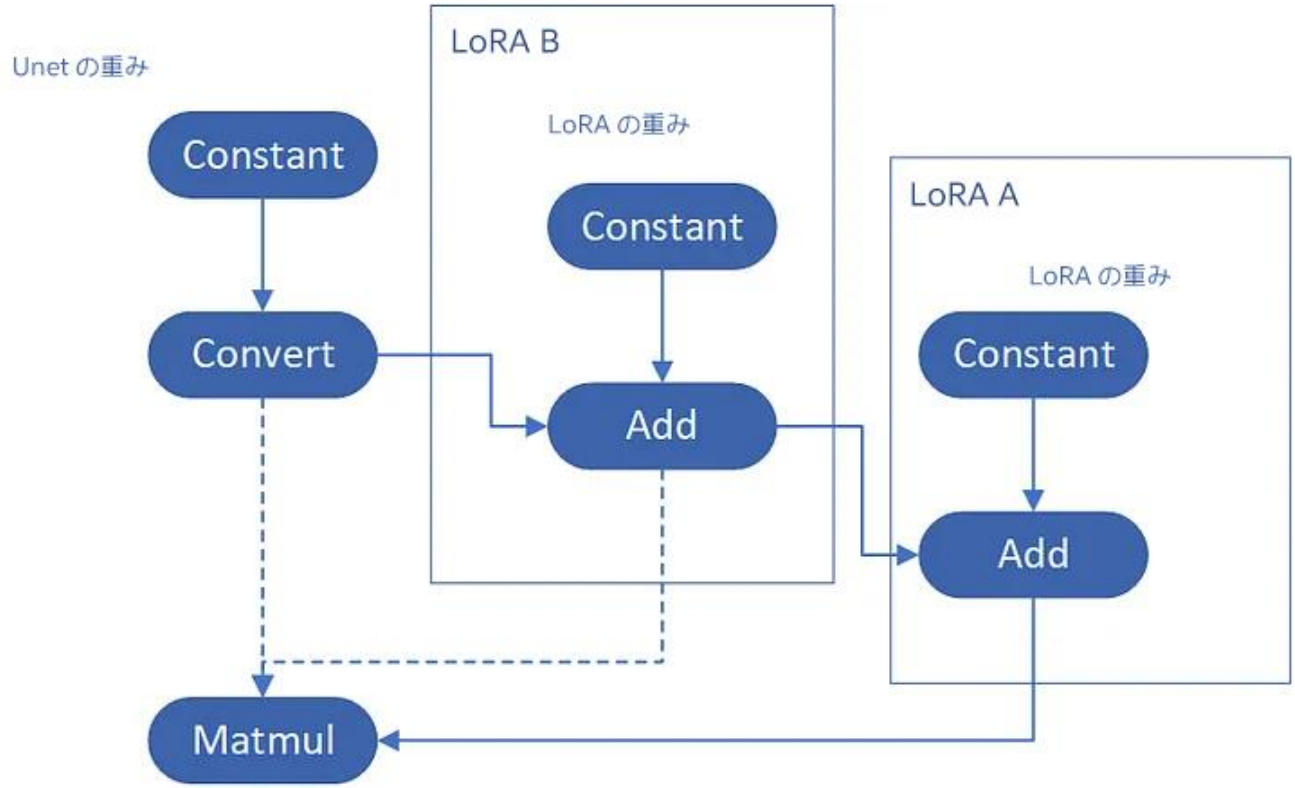
```
python run_pipe.py -lp fluffy-stable-diffusion-1.5-lora-trained-without-data/fluffySingleWordConcept_v10.safetensors -a 1.5
```

ControlNet パイプラインを使用して LoRA の重みを追加した Stable Diffusion モデルでは、次のような画像が生成されます。

Stable Diffusion v1.5 + ランタイム LoRA safetensors の重み + ControlNet

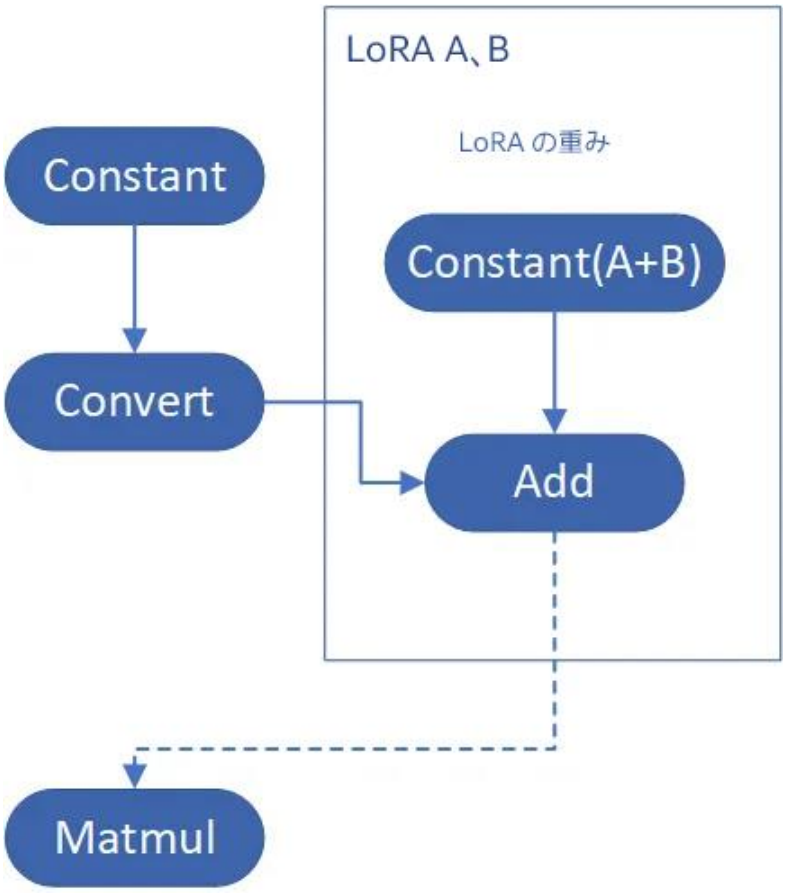
ステップ 4–4: 複数の LoRA の重みを有効にする

複数の LoRA の重みを追加するには、さまざまな方法があります。ここでは 2 つの方法を紹介します。LoRA A と LoRA B という 2 つの LoRA の重みがあると仮定します。ステップ 4–3 に従って MatcherPass 関数をループし、オリジナルの Unet Convert 層と LoRA A の Add 層の間に挿入します。実装は簡単ですが、パフォーマンスは良くありません。



方法 1: InsertLoRA() を 2 回ループする

MatcherPass 関数のロジックを考慮してください。この関数は、Convert 型のすべての層をフィルターで除外し、重み Constant により接続されている各 Convert 層が LoRA の重みファイル内で微調整および更新されているかどうかの条件判断を行うために必要です。LoRA を有効にする主なコストは InsertLoRA() 関数によるものであるため、1 回の InsertLoRA() 関数の呼び出しで複数の LoRA ファイルの重みを追加する方法を考えます。



方法 2: すべての LoRA の重みを一緒に追加する

上記の複数の LoRA の重みを追加する方法を使用すると、2 つ以上の LoRA の重みを追加するコストが、1 つの LoRA の重みを追加するコストとほぼ同じになります。

dreamlike-anime-1.0 (英語) で Stable Diffusion を変更して、アニメのような画像を生成してみましょう。ここでは、https://civitai.com/tag/lora (英語) から Stable Diffusion 1.5 用の 2 つの LoRA の重みを選択しました。

- soulcard: https://civitai.com/models/67927?modelVersionId=72591 (英語)
- epi_noiseoffset: https://civitai.com/models/13941/epinoiseoffset (英語)

次のようなプロンプトを生成するには、プロンプトのエンジニアリング作業を行う必要があるでしょう。

- **プロンプト**: 「1girl、cute、beautiful face、portrait、cloudy mountain、outdoors、trees、rock、river、(soul card:1.2)、highly intricate details、realistic light、trending on cgsociety、neon details、ultra-realistic details、global illumination、shadows、octane render、8k、ultra sharp」
- **ネガティブプロンプト**: 「3D、cartoon、low-res、bad anatomy、bad hands、text、error」
- **シード**: 0
- **num_steps**: 30
- **canny low_threshold**: 100

```
$ python run_pipe.py -lp soulcard.safetensors -a 0.7 -lp2
epi_noiseoffset2.safetensors -a2 0.7
```

次のようにソウルカード特有の縁が付いたアニメのような女の子の画像を取得できます。

Stable Diffusion dreamlike-anime-1.0 + canny ControlNet + soulcard + noiseoffset

関連情報 (英語)

OpenVINO™ のダウンロード

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