参考資料

Application Note

*メモリと **DDR** の帯域幅低減による **TI** のディープラーニング性能の最適化*

Adam Hua, Reese Grimsley

概要

TIDL は、TDA4x と AM6xA シリーズのプロセッサ上で動作する TI の AI 推論フレームワークであり、内蔵の C7xMMA AI アクセラレータを活用して効率的な AI モデル推論を実現します。C7xMMA は、専用の AI 推論アクセラレータであり、複雑なアーキテクチャを採用しています。TIDL 推論フレームワークは、リソース割り当てを広範に最適化して使用効率を最大化していますが、モデルの推論中もメモリ帯域幅の消費を引き続き大きく発生させることができます。推論リソースをさらに活用し、メモリ使用量を削減するには、TIDL で動作するモデルに追加の最適化が必要です。このドキュメントは、DDR の帯域幅消費を低減するためのモデル最適化方法について詳しく説明しています。

目次

商標

1 はじめに

AI モデルの推論に対応できる現在の SoC は、通常、汎用 GPU を統合するアーキテクチャと、一般的に NPU と呼ばれる専用の AI 推論アクセラレータを内蔵するアーキテクチャのいずれかを採用しています。TDA4x と AM6xA の各製品ラインアップに属する TI の AI アクセラレーション対応 SoC は、後者のアプローチを採用しており、一般的に C7xMMA と呼ばれる独自の NPU を備えています。NPU の 2 つのコンポーネントである C7000 シリーズ浮動小数点デジタル信号プロセッサと行列乗算アクセラレータ (MMA) に由来します。C7x シリーズの DSP コアは RTOS を動作させ、データのスケジューリングと、モデル内での非線形処理を処理します。MMA は C7x と深く結合しており、大部分のニューラルネットワークの計算要件の 99% 超を占める行列の乗算や 2D 畳み込みのような線形代数演算を担当しています。

TI では、TI ディープラーニング (TIDL) 推論フレームワークを提供し、高レベルのオペレーティング システム (Linux、QNX *など*) を使用して容易に起動できるように統合されたインターフェイスを提供しています。呼び出し方法はこの文書の範囲を超えています。私たちは読者が関連するインターフェイスに精通していることを前提としており、モデル最適化手法に焦点を当てます。ユーザーは TIDL ツールを活用して、特定のプロセッサ向けのモデルをコンパイルします。次に、TIDL は、量子化されたコンパイル済みモデルを NPU に導入し、ユーザーは TIDL ランタイム (TIDL-RT)、tivxTIDLNode または ONNX Runtime や Tensorflow-Lite のようなオープンソース ランタイム (OSRT) を使用して推論を起動できます。

TIDL メモリの読み取り/書き込み帯域幅とは、DDR インターフェイス上の負荷を指します。たとえば、単一の推論フレームで DDR から 100MB を読み取る (モデルの重み、入力、中間層の機能マップを含む) と DDR への 50MB の書き込み (モデル出力と中間機能マップを含む) が必要な場合、30fps のフレームレートを達成するには、DDR の読み取り/書き込み帯域幅の合計が 4.5GB/s である必要がありますシングルチャネル DDR4 の実際の帯域幅は約 8GB/s である場合、TIDL モデルの推論はかなりの帯域幅を消費する可能性があります。

AM67A や TDA4VH などの一部のデバイスには、C7x の複数のインスタンスが内蔵されています。並列アクセラレータ間で並列推論タスクを実行すると、DDR の使用率と競合にさらに影響が生じます。ただし、TDA4VH などのプロセッサには複数の DDR インターフェイスが含まれます。システム レベルの性能表示する際には、システム全体の DDR 競合を考慮する必要がありますが、このドキュメントの最適化は、モデル性能を向上させるために最初に DDR の使用率を削減するのに有益です。

DDR の帯域幅を最適化するには、C7xMMA のキャッシュ構造についてある程度の理解が必要で、TIDL ツールを効果的に活用し、モデルを変更する可能性もあります。

2 C7xMMA キャッシュ構造

DDR の帯域幅の最適化は、以下の図に示すように簡略化された TI C7xMMA のメモリ階層を理解することから始まります。

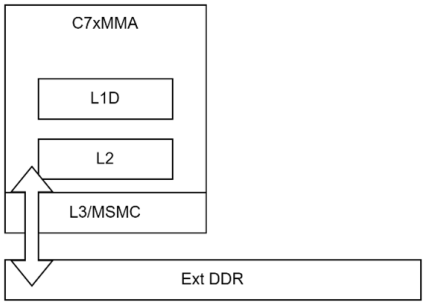

図 2-1. C7xMMA レベルキャッシュ構造

C7xMMA は、3 レベルのキャッシュ構造を採用しています。外部 DDR 以外に、内部 L1D、L2、L3/MSMC キャッシュを内蔵しています。L1D は最小で、演算コアに最も近い (標準サイズ 16KB)。L2 はやや遠い (典型的なサイズは 224KB、448KB) が、MMA のデータ移動メカニズムと緊密に結合している。TDA4x の L3 はマルチコア共有メモリ コントローラ (MSMC) であるのに対し、他の SoC では、SRAM は各 C7xMMA によって個別に管理されます。注:ここに示す L1D、L2、L3 の用語は、TIDL フレームワークの説明に対応しています。チップ データシートは L1P、L1D、L2、および一部の SoC や L3 (つまり TDA4VM の MSMC) を参照することがあります。L2 および L3 領域のサイズは、tidl ツールに含まれる device_config.cfg ファイルにあります

次の図は、4 つのオペレーションを含む一般的なレイヤの推論におけるキャッシュの使用状況を示しています。オペレーション 1 は、DMA から DDR から L2 にデータを直接転送します。オペレーション 2 では、L3 から L2 にデータを移動します。オペレーション 3 は L2 から L3 にデータを転送します。オペレーション 4 は、L3 から DDR にデータを移動します。オペレーション 2 とオペレーション 3 は 1 と 4 の 10 倍以上の効率があります。前のレイヤの機能マップを使用すると、次の 3 つのシナリオが発生します。オペレーション 1 のみ (入力レイヤと前のレイヤ出力が完全に DDR にある場合)、オペレーション 2 のみ (前の機能マップが L3/MSMC に完全に適合する場合)、オペレーション 1 と 2 の両方 (前の出力が L3 に大きすぎる場合、DDR に部分的に格納されます)。現在のレイヤーの機能マップを計算した後、データを L3 に移動するためにオペレーション 3 が優先されます。L3 容量を超えると、Operation 4 は余剰分を DDR に保存します。重み値は常に DDR に格納され、必要に応じて L2 に直接フェッチされます。

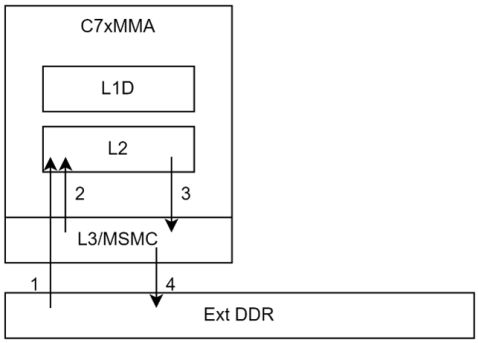

図 2-2. C7xMMA キャッシュ オペレーション

この 3 レベル キャッシュ アーキテクチャは、計算サイクル中の低速な DDR 読み取り/書き込みを回避し、DDR の帯域幅を節約することで、推論の効率を大幅に向上させます。効率性を向上させ、帯域幅を節約するための鍵は、L3 の使用率を最大化し、DDR の機能マップストレージを防止することにあります。次のセクションでは、モデル メモリ使用量の分析について説明します。

3 コンパイル済み TIDL モデルの DDR 読み出し/書き込み分析モデリング

TI モデル コンパイル ツールは、高速な DDR 読み取り/書き込み分析を容易にする対応インターフェイスを提供します。モデルをコンパイルすると、出力フォルダに bufinfolog_0.csv ファイルが生成されます。正確には、artifacts/tempDir/$SUBGRAPH_NAME_tidl_net のような名前のフォルダに置かれます。SUBGRAPH_NAME はモデルに依存します。フォルダ構造の正確な名前は、SDK リリースによって若干変更される場合があります。

RTOS (tidl_model_import.out など) ツールを使用する場合は、perfSimConfig が用意されていること、およびツールがエラーなく実行されることを確認する必要があることに注意してください。モデルのコンパイルに失敗した場合は、帯域幅消費を分析する前にエラーを解決する必要があります。コンパイル プロセス中にエラー ログが発生すると、分析結果が不正確になる可能性があります。

以下の表は、bufinfolog CSV ファイルの主要なフィールドについて説明しています。

表 3-1. *bufinfolog* CSV ファイルのフィールドの説明

フィールド	説明
Ni	入力機能マップ チャネル ディメンジョン。
No	出力機能マップ チャネル ディメンジョン。
InW	機能マップの幅ディメンジョンを入力します。
InH	機能マップの高さディメンジョンを入力します。
OutW	機能マップの幅ディメンジョンを出力します。
OutH	機能マップの高さディメンジョンを出力します。
In-Write-size	実際の入力機能マップ サイズ (バイト単位で計算)。パディングまたは DMA バスのサイズ設定のための小さなオーバーヘッドが含まれる場合があります
In-Write-memSpace	L2 または DDR。すべての層はデータ層 (入力/出力層) を除く L2 を使用
Out-Read-memSpace	現在のレイヤーの計算結果の一時的な保存場所。値は通常、Empty、L2、MSMC、または DDR です。Empty は、計算結果が Out-Write-memSpace に直接格納されることを示します。値が存在する場合は、現在のレイヤーの出力機能マップの一部を最初に out-Read-memSpace に格納してから、このスペースから読み取り、out-Write-memSpace に書き込む必要があることを示します。
Out-Write-memSpace	現在のレイヤーの機能マップの最終保存場所。値は DDR または MSMC です。DDR は、現在のレイヤーの出力機能マップが MSMC に完全に適合しないため、その一部またはすべてが DDR に格納されていることを示します。MSMC は、出力がすべて MSMC に格納され、後続のレイヤーで使用されることを示します。
Out-Write-size	現在のレイヤーの out-Write-memSpace に保存されているデータ量。out-Write-memSpace が DDR の場合、この値は DDR 書き込み帯域幅消費に対する現在のレイヤーの寄与を表します。パディングまたは DMA バスのサイズ設定のための小さなオーバーヘッドが含まれる場合があります
Wt-Write-memSpace	現在のレイヤーの重みがロードされる場所。常に L2 です。重み値はすべて DDR に保存され、使用時は L2 にロードされます。
Wt-Write-size	現在のレイヤーの重みデータのサイズ。この値は、DDR 読み取り帯域幅の消費に寄与します。パディングまたは DMA バスのサイズ設定のためのオーバーヘッドが含まれる場合があります

上記のフィールドの中で、DDR 書き込み帯域幅は主に **out-Write-memSpace** の値によって影響されます。レイヤーの出力機能マップサイズを小さくすることで、DDR 書き込み帯域幅の消費を大幅に減らすことができます。

DDR リード帯域幅は主に **wt-write-size** と **in-write-size** の影響を受けます。特に、**In-Write-Size は**、前のレイヤーの出力が部分的または完全に DDR に格納されている場合にのみ影響を与えます。

したがって、DDR の読み取り/書き込み帯域幅を削減するために重要なのは、重みによる読み取り帯域幅と、大きな機能マップによる読み取り/書き込み帯域幅を最小化するためにモデルを最適化することです。

4 モデル最適化

上記に基づいて、DDR 帯域幅を減らすには、L3 で中間機能マップを維持する必要があり、意図的なモデル設計が必要になります。

4.1 単純構造モデル

単純なモデルは線形で分岐しない構造を持ちます。以下に示す EfficientNet の初期セクションは完全にシーケンシャルです。ここでは、各レイヤーの出力を L3 に収めるだけで十分です。TIDL は、DDR との相互作用を回避し、L3 を使用して実行するようにレイヤーを自動的に構成できます。DDR インタラクションは、中間機能マップが L3 サイズよりも大きい場合にも必要になります。

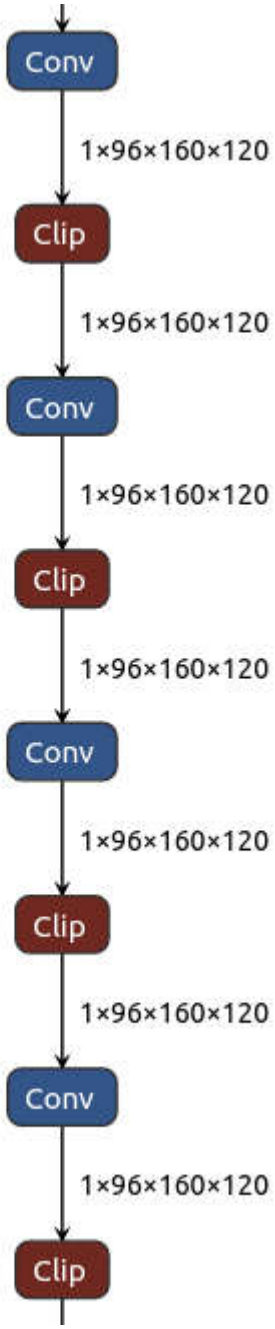

図 4-1. シーケンシャル レイヤー オーダーを使用したシンプルなモデル

4.2 複雑な構造

実際には多くのモデルは連続層のみよりも複雑な構造やグラフ パターンを持っています。次のセクションでは、より複雑な構造の例と、DDR とキャッシュの使用について説明します。

4.2.1 残存構造

多くのバックボーン ネットワークは、図に示すような残存構造を使用しており、トレーニング中に有益な「残余」と呼ばれるローカライズされた並列パスを作成します。残差は消失勾配問題を回避します。

コンパイル時に、TIDL はさまざまな計算順序 (左ブランチファースト、右ファースト、インターリーブ) の DDR 帯域幅をシミュレートし、最も効率的なものを選択します。また、最初の Conv レイヤーの出力が add オペレーションまたは DDR に即座に書き込まれるまで L3/MSMC に留まるかどうかを決定します。DDR に格納すると、直接帯域幅コストが発生し、L3 に保持すると左ブランチの計算中にメモリが占有され、左ブランチの一部が DDR を使用するように強制される可能性があります。

TIDL では L3 占有率を最大化する戦略を選択しますが、大きな中間機能マップでは DDR を使用する必要がある場合があります。この場合は、スキップ接続に沿った単一の機能マップとは対照的に、DDR に送られる複数の機能マップを避けるために、長いパス (図の左側) のサイズを最適化する優先順位を付けることをお勧めします。

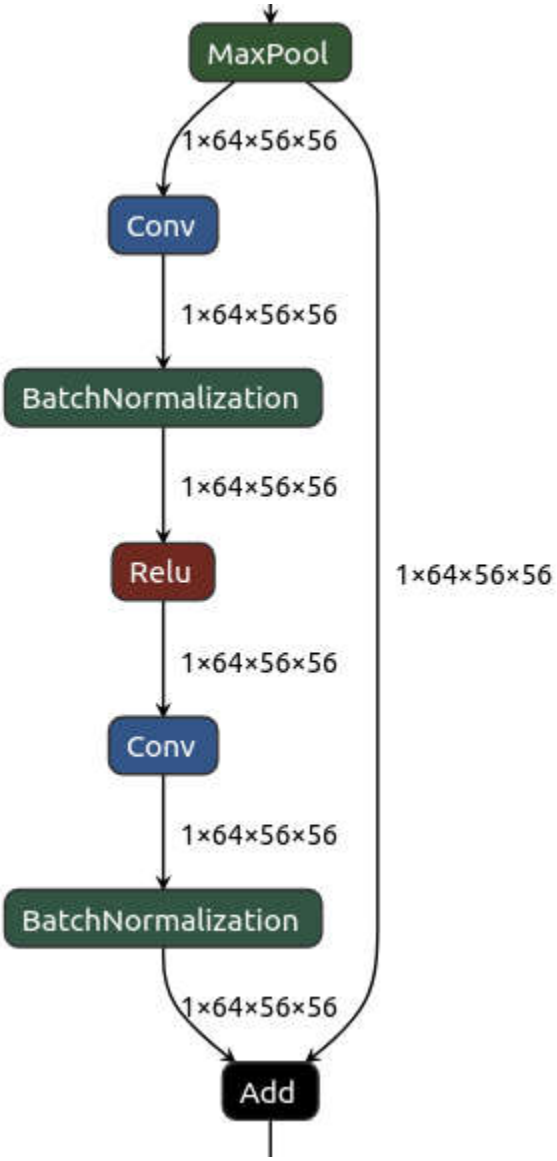

図 4-2. ニューラル ネットワーク内の残存構造。右側のパスの「スキップ」接続には、左側のパスが完了するまで、機能マップを保存する必要があります

4.2.2 並列分岐マージ

多くの場合、アプリケーションでは複数の深い並列分岐が 1 つにマージされたり、1 つの分岐が複数の深い並列パスに分割されたりします。これは、複数入力ニューラルネットワークで特に一般的です。図は、グリッド サンプル オペレータがパスをマージした後の従来の 4 入力 BEV ネットワークの一部を示しています。

パスはマージする前に深くなっているため、機能マップはマージポイントで DDR に配置する必要があります。このため、DDR 帯域幅の消費は避けられません。このようなアーキテクチャは、DDR 帯域幅の超過とその結果生じるボトルネックを回避するために必要な場合にのみ使用してください。ただし、重みによって発生する DDR 読み取り帯域幅を減らすことができます。モデル アーキテクチャは、複数の入力ヘッドを統合し、特定のモデル レイヤーのバッチ寸法を 1 より大きい値に設定するように変更できます。これにより、重み 1 回だけロードする必要があります。

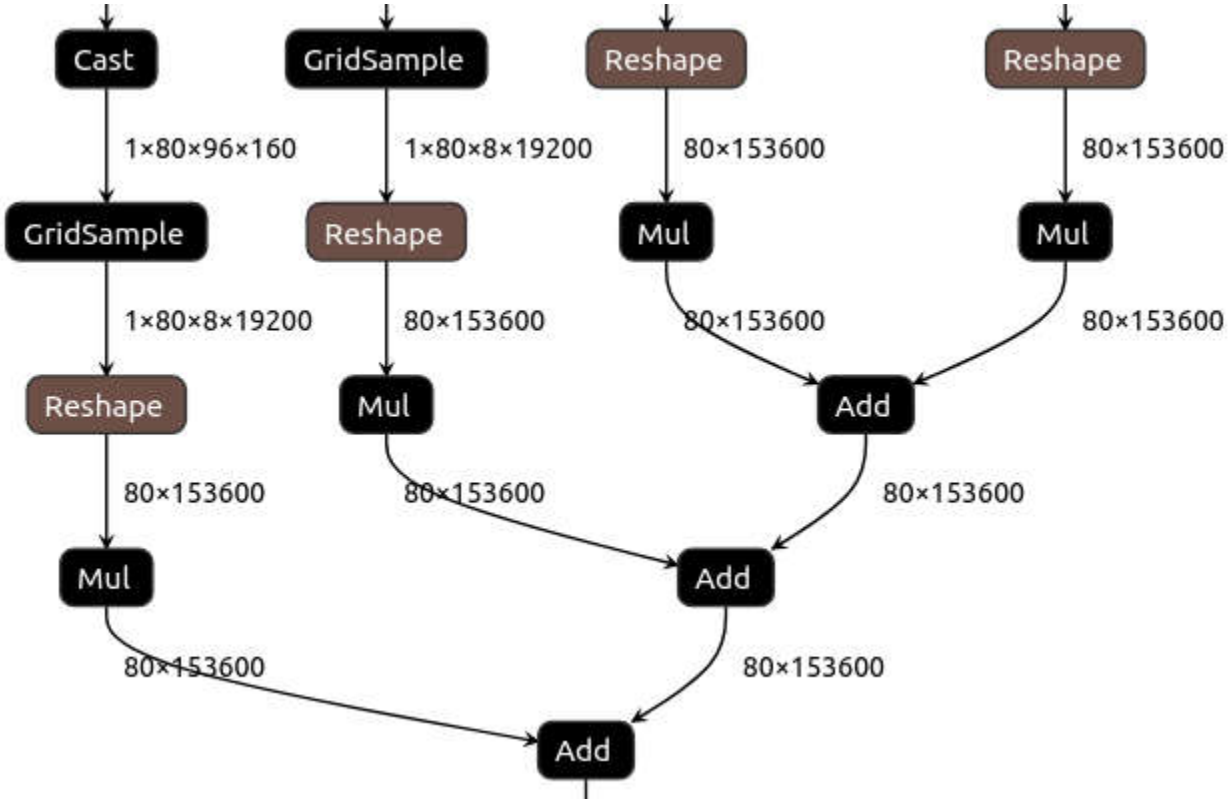

図 4-3. 複雑な構造複数の並列分岐をマージします

たとえば、上の図では、同じバックボーン ネットワークが GridSample レイヤーの前にあり、このバックボーンの各レイヤーの機能マップは比較的小さくなっています。4 つの分岐を 2 つまたは 1 つにマージでき、それに応じてバッチ ディメンジョンが調整されます。これらの後に適切なスライス レイヤーまたはデータシェーピング レイヤーが続き、バッチを再度分離して、表示されている Add レイヤーで再結合できるようにします。このアプローチにより、同じ重みが繰り返しロードされるのを減らすか防止することができ、その結果、DDR 読み取り帯域幅のオーバーヘッドが減少します。この方法では、マージされたバックボーン ネットワーク内の機能マップのサイズに注意する必要があります。

5 まとめ

DDR 帯域幅のモデル最適化には、主にレイヤーごとの機能マップ サイズの縮小と深さの増加を伴います。複雑な構造では、DDR 帯域幅の消費を回避できない場合があります。TI の Model Zoo は、最適化と検証が完了した多数のモデルとバックボーンを提供しています。一般的なアーキテクチャがすでに成熟しているので、迅速な改良を意図して、モデルのバックボーンを TI の最適化済みバージョンに置き換えることを検討してください。

このドキュメントでは、モデル DDR の帯域幅消費を分析し、モデルを最適化してそれを削減する方法を詳細に説明しています。これは、TDA4x、AM6xA シリーズの SoC や、TIDL 推論フレームワークのユーザーに関連があります。通常、これらの方法を適用すると、最適化されたモデルは入力と出力だけの帯域幅を消費し、システム全体のリソースを解放します。

6 参考資料

1. https://www.ti.com/product/AM62A7
2. https://www.ti.com/tool/PROCESSOR-SDK-AM62A
3. https://www.ti.com/product/AM67A
4. https://www.ti.com/product/TDA4VM
5. https://www.ti.com/product/TDA4VE-Q1
6. https://www.ti.com/product/TDA4VM-Q1
7. https://www.ti.com/product/TDA4AL-Q1
8. https://www.ti.com/product/TDA4VL-Q1
9. https://www.ti.com/product/TDA4VP-Q1
10. https://www.ti.com/product/TDA4VH-Q1
11. https://www.ti.com/product/TDA4VEN-Q1

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月