AI に対するニーズの高まりに対応する

この記事は、インテル® デベロッパー・ゾーンに公開されている「Scaling to Meet the Growing Needs of AI」の日本語参考訳です。

人工知能 (AI) とは、コンピューターが人間と同じように考え、行動できるようにすることです。AI はコンピューターを必要としますが、主に人間によって作り上げられたフォーカス、洞察、ヒューリスティックに頼っています。これまで人間を中心とした取り組みが、専門家を中心に行われてきました。専門家 (特に、最初に AI 研究が行われた医療などの分野の専門家) には限りがあるため、成長を続ける AI にとって彼らに頼らなければいけないことが課題でした。

近年、AI は時間とともに向上するアルゴリズムの一種である、マシンラーニングを利用しています。マシンラーニングは、データ処理とハードウェアおよび手法の改善により達成されます。これらの作業は、これまでも AI の原則とされてきましたが、処理と改善に必要な膨大な量のデータが利用できなかったため、実践されてきませんでした。現在では、データ量は毎年倍増しており、計算処理能力よりも速いペースで成長しています。これこそが、現在 AI が注目を集めている真の理由であり、マシンラーニングが AI の能力を発揮するために有効なツールである理由です。

人工知能、マシンラーニング、ニューラル・ネットワーク

ニューラル・ネットワークとは、AI 分野のマシンラーニングに含まれるアルゴリズムの一種で、そのサブクラスにはディープ・ニューラル・ネットワークと呼ばれるグループがあります。ディープ・ニューラル・ネットワークには、2 つ以上の入出力中間層があります。図 1 は、重みによって関連付けられた 単一層の入出力を表したものです。実際のディープ・ニューラル・ネットワークには 100 を超える入出力層があります。

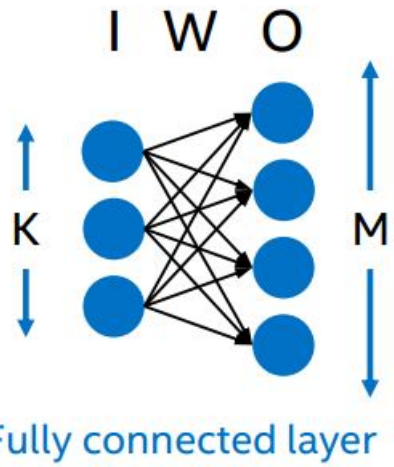

図 1. 単一層のニューラル・ネットワーク

マシンラーニング・アルゴリズムの中でも、ニューラル・ネットワークと呼ばれるアルゴリズムが最も注目されています。この記事では、マシンラーニングとそのアルゴリズムの一種であるニューラル・ネットワークについて取り上げます。図 2 に AI、マシンラーニング、ニューラル・ネットワークの関係を示します。

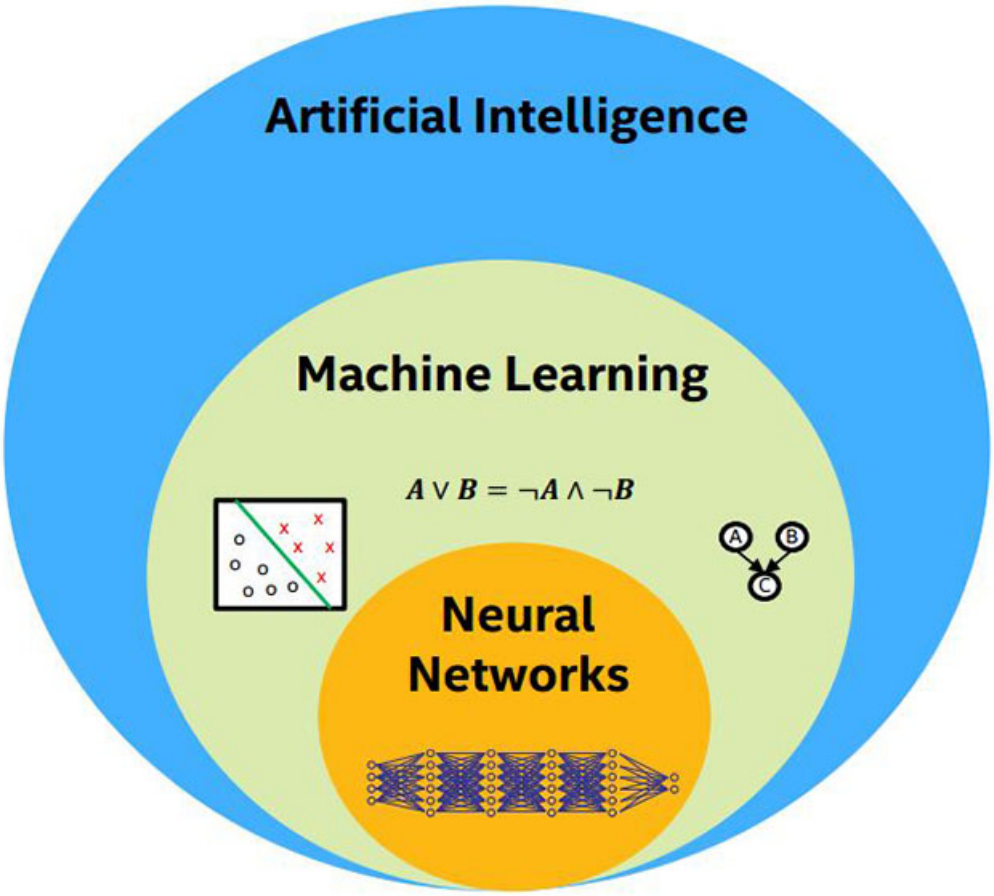

図 2. AI、マシンラーニング、ニューラル・ネットワークの関係

ディープ・ニューラル・ネットワークはイメージを読み取り、対象人物を見つけ、その人物を線で囲んでラベル付けします。ネットワークが適切にトレーニングされている場合、このように動作します。入力から出力を生成するまでの一連の流れは「フォワード・プロパゲーション」と呼ばれ、このタスクは「推論」と呼ばれます。

この処理を適切に行うため、ニューラル・ネットワークをトレーニングする必要があります。このトレーニングが課題です。

トレーニング

どのようにネットワークをトレーニングしたら良いのでしょうか? まず最初に、フォワード・プロパゲーションを実行して、結果を確認します。そして、実際の結果と正解を比較して、その違いを返します (図 3)。これは、バックワード・プロパゲーションと呼ばれます。バックワード・プロパゲーション・アルゴリズムが最も難しい部分です。複数のトレーニング・シナリオ (この例ではイメージ) について、層ごとに細心の注意を払って各エッジの重みを調整する必要があります。

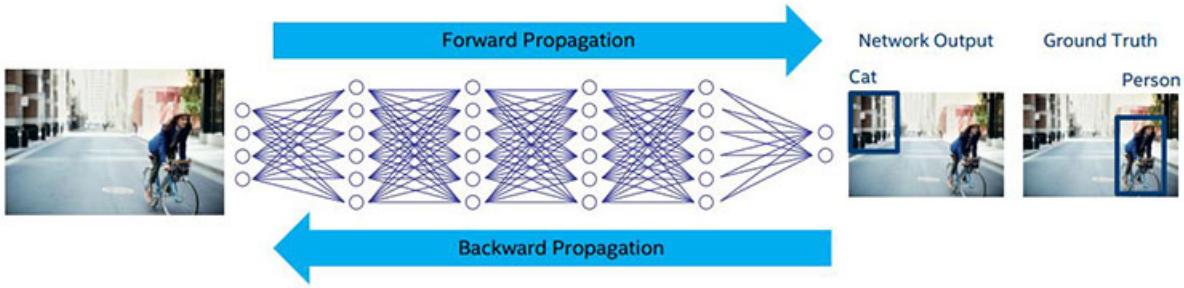

図 3. ニューラル・ネットワークの入力、ノード、出力、正解

最先端のトレーニング・スキームには主に 2 つの課題があります: a) ラベルデータへの依存 (教師ありトレーニング)、b) アルゴリズムの並列性が大きく制限される。トレーニング用のデータのラベル付けは、手間のかかる作業ですが、専門家が行う必要はありません。通常は、ターゲット・オブジェクトを囲むだけであり、誰にでもできます。アルゴリズムの並列性は、ほとんどの場合、共通の特徴を学習するため一緒に処理されるイメージのバッチサイズによって制限されます。大きなバッチサイズはトレーニング・データには適さないため、一般に 1000 程度に抑えられます。

膨大な量の入力データ (イメージ) とネットワーク層を、トレーニングでは通常 1 層ずつシリアルに処理します。ネットワークは非常に大きく、数百万もの重みを決定しなければならないため、トレーニングには時間がかかります。多くの場合、トレーニングにかかる時間 (数週間、場合によっては数カ月にも及ぶ) は許容できるものではありません。ある程度トレーニングされたモデルなしには推論を開始できないため、トレーニング時間の短縮が重要になります。

トレーニング時間を短縮するには、どうしたら良いのでしょうか? マシンラーニングは、完全なモデルが完成するまで開始できません。

優れたトレーニングは、簡潔で優れたモデルをもたらします。これは、優れた風刺画家が 5 筆で人物を表現することができるのに似ています。筆使いの可能性は無限大です。5 筆をどこに、どのように適用するかによって結果が得られます。経験豊富な画家であっても、多くの場合、その理屈を説明できないため、簡単に再現することは難しいでしょう。十分なデータでトレーニングされたマシンラーニング・アルゴリズムは、人の手助けがなくても正しく簡潔な表現を学習することが可能で、オリジナルイメージの認識や再現に使用できます。

自動運転車は、マシンラーニング・プロセスの良い例です。車両には、センサー処理、センサー・データ・キャプチャー、経路計画、運行管理機能が搭載されています。データセンターでは、全車両の車両エンドポイントの管理、車両のシミュレーションと検証、キャプチャーしたセンサーデータの解析を行います。マシンラーニングは、車両とデータセンターの両方で利用されます。車両固有の処理は車両側で行い、広範な車両間の処理はデータセンター側で行います。ほとんどの推論と処理はエッジデバイスとクラウドで行いますが、エンドツーエンドの処理も必要なため、クロスデバイスの学習も行います。

ニューラル・ネットワーク層

全結合層とは、すべての入力がすべての出力 (結果) に関連付けられている層です。層には、入力、重み、出力があります。全結合層は、行列演算を使用する単純な行列処理です。全結合ではない畳み込み層も、ニューラル・ネットワークではよく行列演算を使用して処理されます。計算の 90% 以上が行列乗算で、そのほとんどは密線形代数です。つまり、ニューラル・ネットワークのコアカーネルは、計算負荷がそれほど高くなく、密線形代数に対応したアーキテクチャーであれば問題なく処理できるはずです。

図 4 に必要な乗算処理を示します。フォワード・プロパゲーションは、入力行列と重み行列から出力行列を計算します。バックワード・プロパゲーションは、出力行列と重み行列を確認し、入力行列を計算して返します。そして、重みを更新するため、入力と出力の差分を比較し、新しい重みを計算します (図 4)。

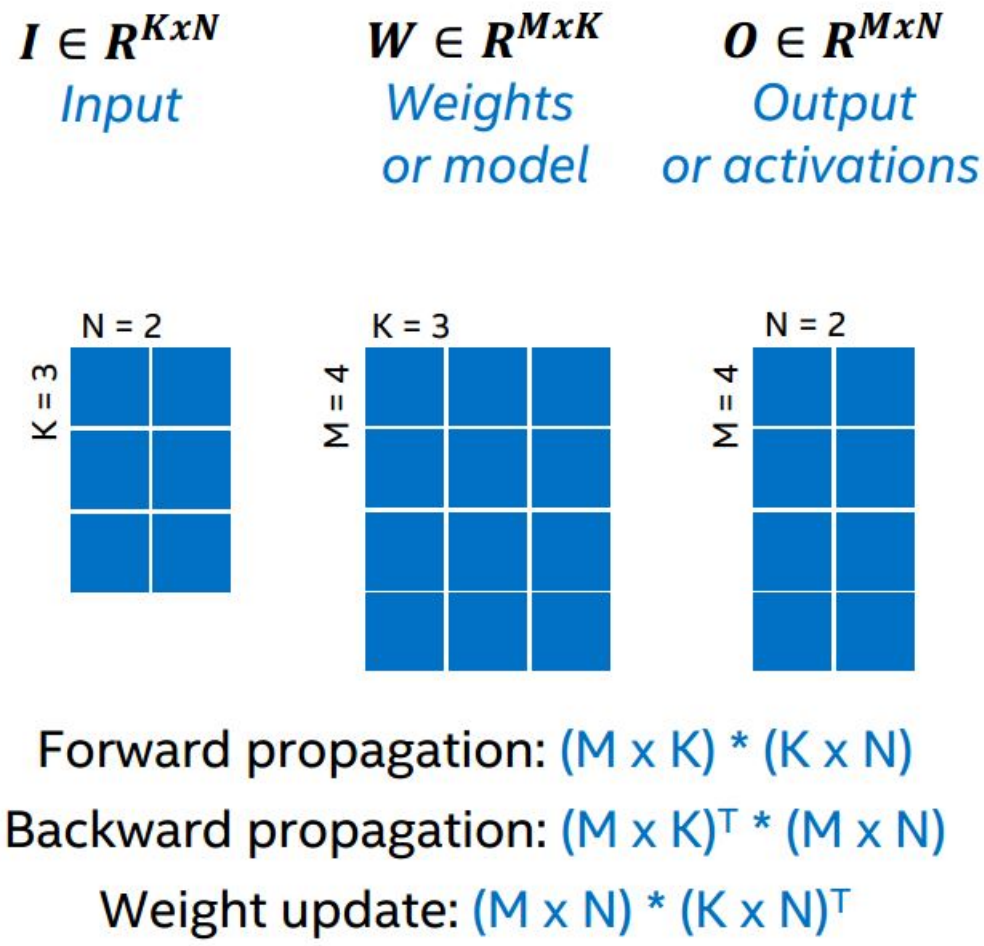

図 4. 単一層の行列と行列計算

並列性

並列性には、データ並列性、モデル並列性、ハイブリッド並列性があります。

データ並列性では、データを 2 分割し、同じ重みを使用して 2 つの異なるノードで並列に実行します。重みは同じですが、並列に実行するため、入力データと出力データが分割されます。

モデル並列性は、データ並列性に似ていますが、モデル (重み) に適用されます。重みが 2 分割され、データが 2 分割した重みで並列に実行されます。通常、データ量 (n) に対しモデルサイズが大きい (n2) 全結合層で使用されます。

ハイブリッド並列性 (図 5) は、データ並列性とモデル並列性の組み合わせです。

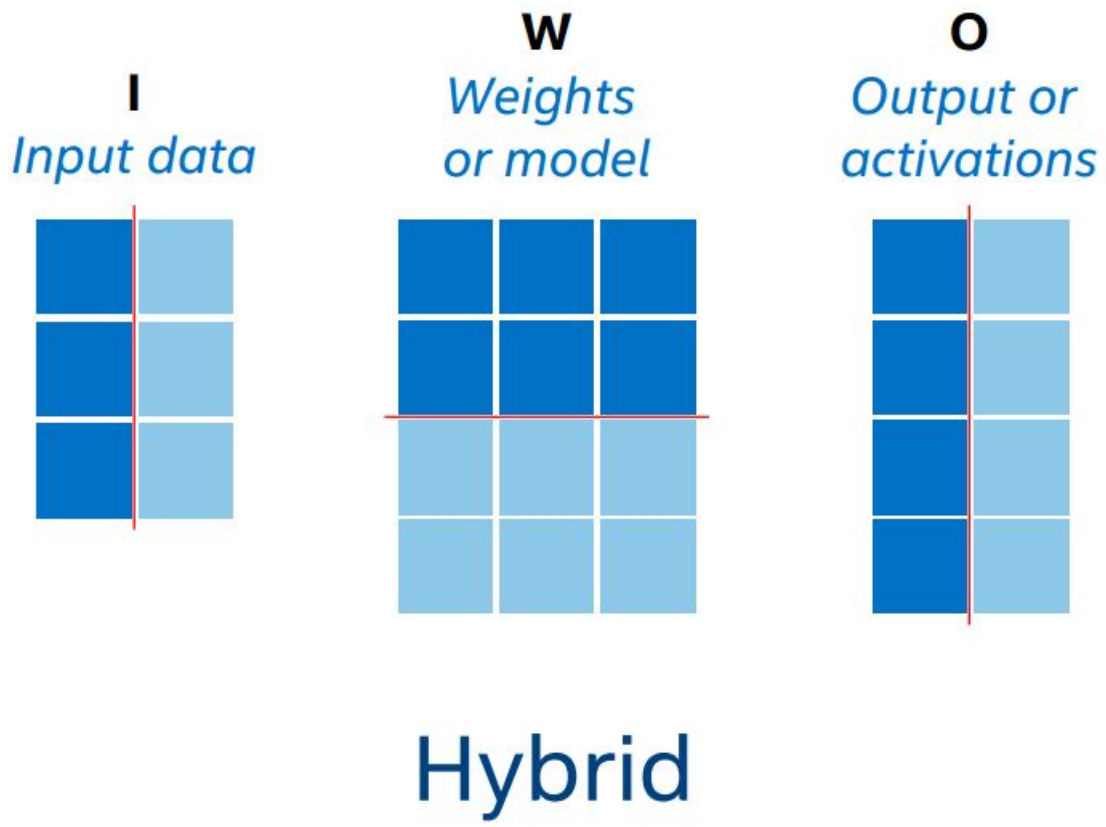

図 5. 並列に実行するためデータと重みが分割されたハイブリッド並列性

最適な並列性を判断するには、どうしたら良いのでしょうか? データまたはモデルを分割すると、行列が変更され、処理が困難になります。例えば、大きな正則 (次元がほぼ同じ) 行列を数回分割すると、縦長の (行数が多く、列数が少ない) 歪行列になります。正則行列の場合、データブロックをプロセッサーのキャッシュやメモリーにフェッチしたり、ワイドな SIMD 並列処理を利用できますが、歪行列ではこれらを利用できません。場合によっては、同じノードの計算が分割され、それらをまとめるため追加の作業が必要になります。ノード間通信と歪行列は、高度な並列アーキテクチャー向けの最適化を妨げます。

一般に、次のことが言えます。

- 活性度 (出力) > 重み (モデル) の場合、データ並列性を使用します。
- 重み > 活性度の場合、モデル並列性を使用します。

各並列性における分割は、次のことを意味します。

- データ並列性では、活性度が重みよりも非常に小さくなります。
- モデル並列性では、重みが活性度よりも非常に小さくなります。
- 歪行列は計算効率を低下させます (例えば、10x4k 行列はキャッシュサイズに関係なくプロセッサー時間を占有します)。

スケーリング

大規模な並列化では、通信時間の合計計算時間に占める割合が高くなります。その結果、ノードの計算能力は上がりますが、トレーニング時間全体におけるワークロードのパフォーマンスが低下します。大規模な計算リソース (パブリッククラウドなど) を利用して、大規模なモデルのトレーニング時間を短縮するには、ナイーブな並列スキームに加えて、ハイブリッド並列性の使用とノード間通信の制限/管理の 2 つの最適化が必要です。これらの最適化を行うことで、効率良くスケーリングできます。ハイブリッド並列性は、行列が歪曲されるのを防ぎます。さらに、ノード間通信を軽減するため、ノードグループを作成し、各グループ内で活性度の変換とグループ間で重みの変換を行うと良いでしょう。

ディープラーニングにおける通信パターン

ノード間通信パターンの種類について詳しく見てみましょう。複数のノードにわたって並列化する場合、ノード間通信が必要になります。例えば、ドット積の計算で行列の 1 つの要素があるノードにあり、もう一方の要素が別のノードにある場合、ドット積を求めるには、2 つの異なるノードにある部分活性度の通信と結合が必要になります。データが次の層にない場合、計算を行うことはできません。図 6 にノード間通信の詳細を示します。

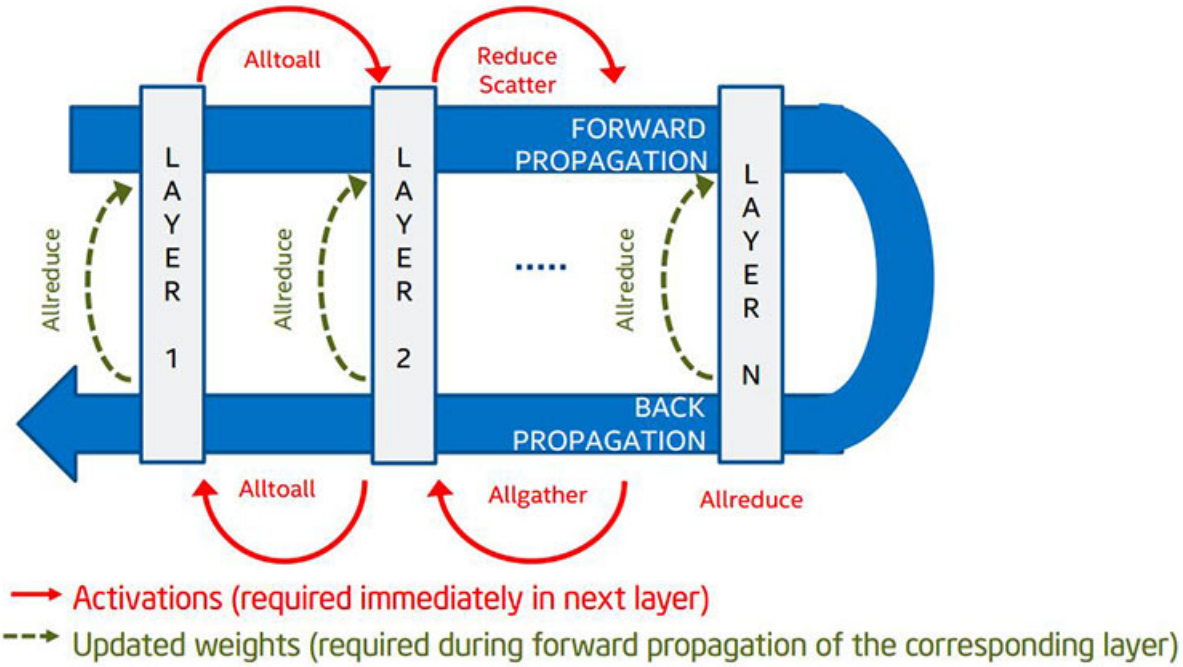

図 6. ディープラーニングにおける MPI 集合操作

緑色の線 (Allreduce) は待機できますが、赤色の線 (Alltoall ほか) は次の層の計算に必要です。緑色の線は緊急性の低い通信を表し、データは次のフォワード・プロパゲーション・フェーズで必要になります。赤色の線は緊急性の高い通信を表し、データは次の層の計算に必要で、遅延により処理パイプラインにバブルが生じます。効率の損失は異なる並列性により対応できますが、赤色の線は対応できません。そのため、赤色の線で示すメッセージは、優先的にスケジュールする必要があります。

要約すると、通信パターンの効率を向上するには、次のステップが必要になります。

1. 処理時間を最小にするため、さまざまなノード間通信プリミティブのパフォーマンスを最適化します。
2. 通信によるパフォーマンスへの影響を最小限に抑えるため、通信と計算をできるだけオーバーラップさせます。
3. 緊急性の高いノード間通信メッセージを優先的にスケジュールします (緑色よりも赤色を優先します)。

図 7 は、最適化した Allreduce プリミティブのパフォーマンスの向上 (現在のインテル® MPI ライブラリー実装と比較) を示しています。

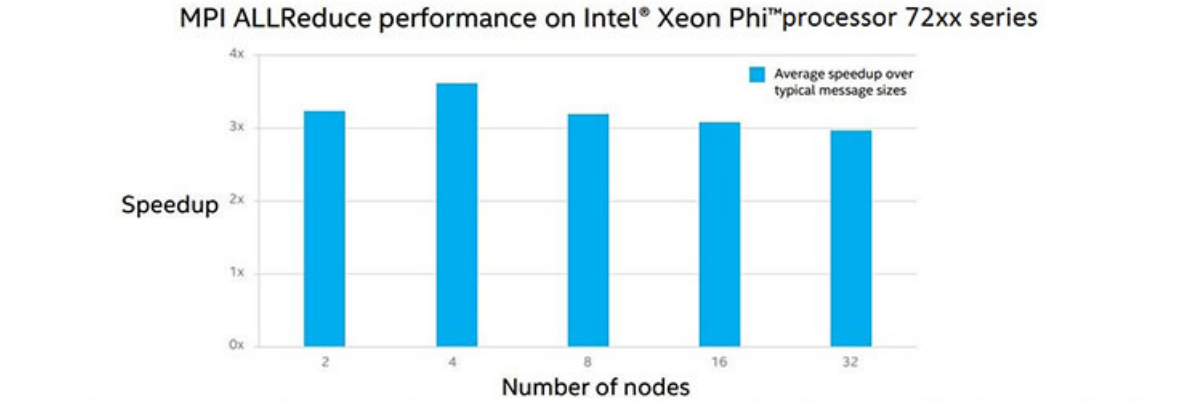

図 7. MPI Allreduce パフォーマンス

通信サイクル数を少なく、計算サイクル数を多くすると効率が向上します。図 8 と図 9 にこれらの最適化の効率を示します。

図 8. 効率良い通信を使用した場合のスケーリング効率

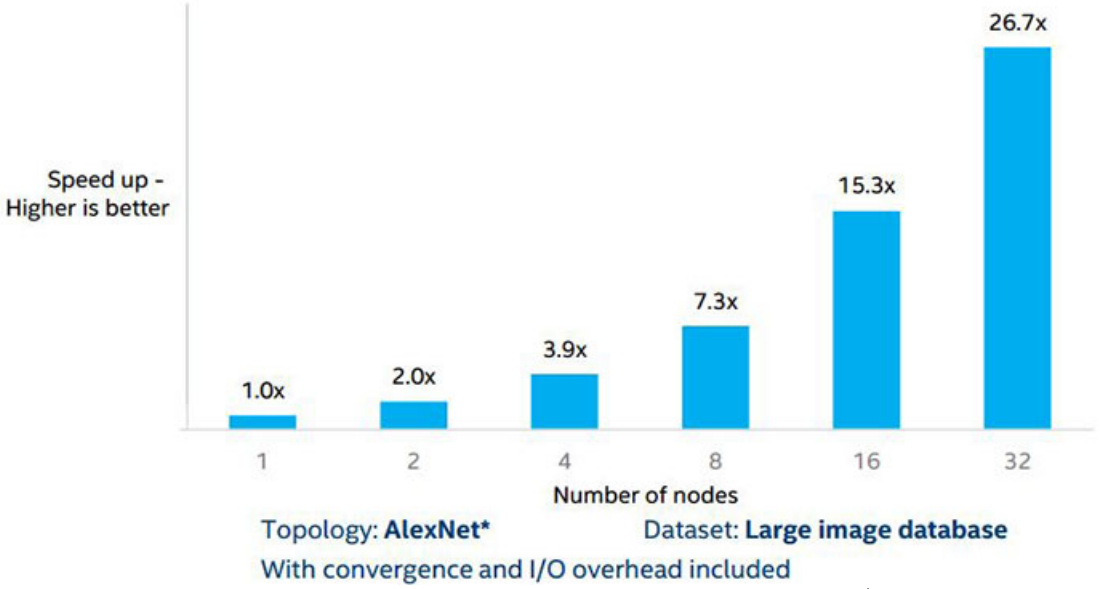

図 9. 効率良い通信を使用した場合の AlexNet のスケーリング効率

これらの最適化を組み合わせることで、全体のスケーラビリティーが向上し、さまざまなニューラル・ネットワーク・トポロジーでパフォーマンスが向上します。図 10 は、一般的な 4 つのトポロジーのスケーリング結果です。ここに表示されているのは 128 ノードまでですが、我々は数千コアでのスケーリングに取り組んでいます。

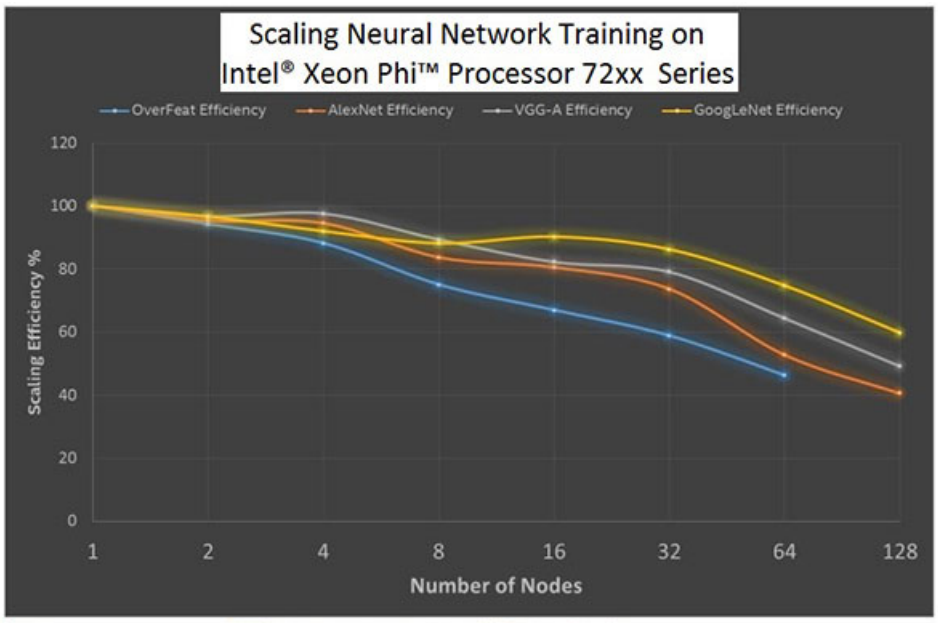

図 10. 一般的な 4 つのトポロジーとインテル® Xeon Phi™ プロセッサー上でのスケーリング効率

トレーニング時間を短縮した後の作業

トレーニング時間の短縮を達成したら、次は何をすべきでしょうか? 簡潔で、高速かつ正確なモデルが準備できたら、エッジデバイスに配置して、実際のデータで実行し、推論を生成します。そして、推論のスコアカードをサーバーに送ります。サーバーは、新しいデータで再度トレーニングを行い、モデルを改善して、さらに多くのデバイスに配置します。

図 11. 計算の好循環

モデルの推論の精度が高いほど、多くのエッジデバイスに配置される可能性が高まり、多くの場所に配置されることで、(予測が正しかった場合と正しくなかった場合について) より多くの推論データがサーバーに寄せられ、サーバーがモデルの推論の精度を調整および向上できるようになります。そして、モデルがさらに多くのデバイスに配置され、このサイクルが繰り返されます。これが、計算の好循環 (図 11) です。

この循環は、実際にデバイスに送信できなければ完全ではありません。ネットワークには常に制約があるため、デバイスへの送信にはさらに追加の作業が必要です。エッジデバイスとデータセンターを繋ぐネットワークの長いレイテンシーと帯域幅の制限により、リアルタイムで高速にモデルの再トレーニングを行うことができるデータセンター・インフラストラクチャーを構築することは困難です。ハードウェア、ネットワーク、ワイヤレス接続が改善されることで、この課題は取り組みやすくなるでしょう。

さらに、エッジデバイスを理解することが重要です。多くの場合、計算能力やメモリー容量により、モデルサイズが制限されます。ツール要件とソフトウェア・サポートは、エンドツーエンドで対応しなければなりません。インテル® Deep Learning SDK は、モデルの配置に関するそれらの要因を考慮して設計されています。

エッジデバイス上にモデルを配置する場合、圧縮、精度、スループットの 3 つのトレードオフがあります (図 12)。モデルサイズが大きすぎてデバイスに配置できない場合、モデルを圧縮、縮小するなどして小さくする必要があります (図 13)。モデルを圧縮するには、分散させます [1, 2]。

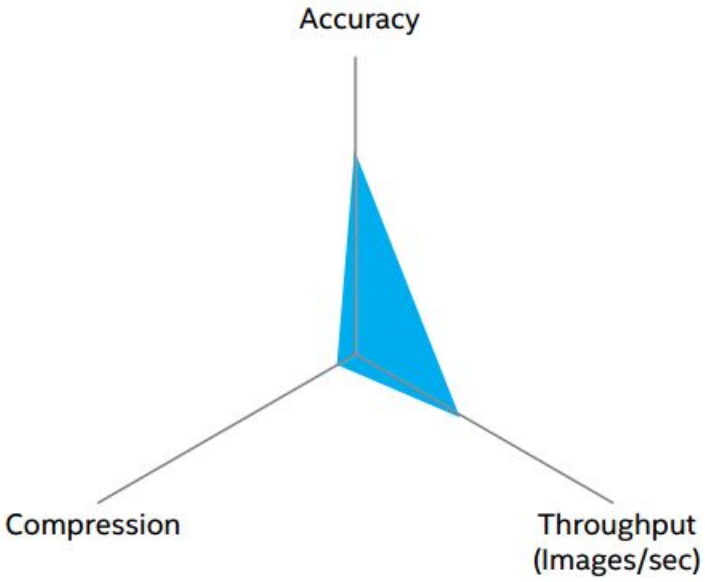

図 12. 3 軸のモデル・パフォーマンス

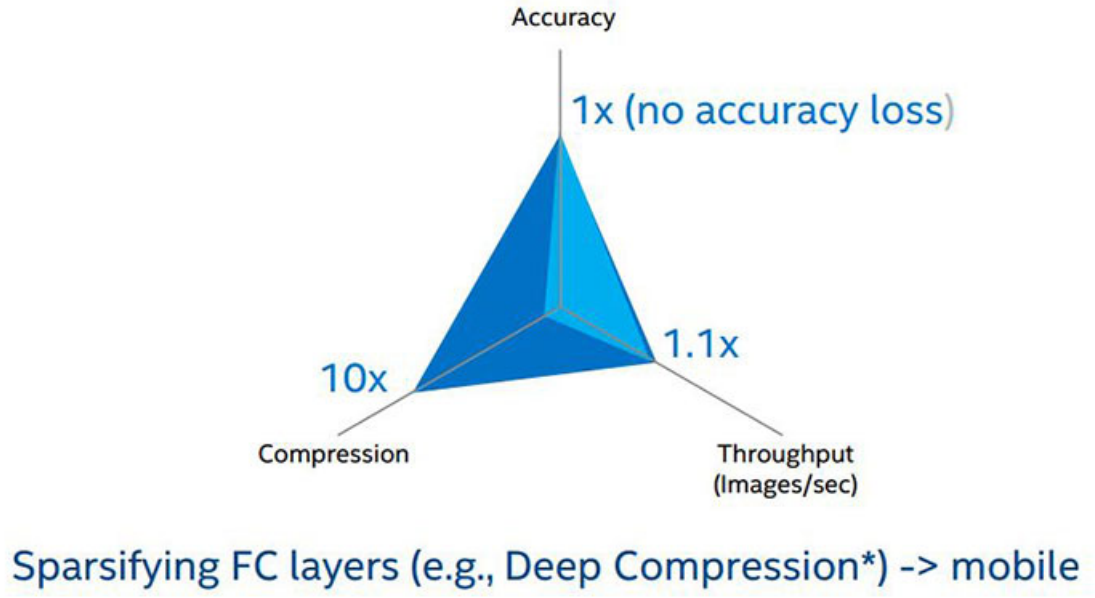

図 13. 通常の配置と比較した圧縮 (濃い青色) のパフォーマンス

特定の状況下では、高い推論スループットが求められます。その場合、推論の精度を下げることで、スループットを上げることができます (図 14)。

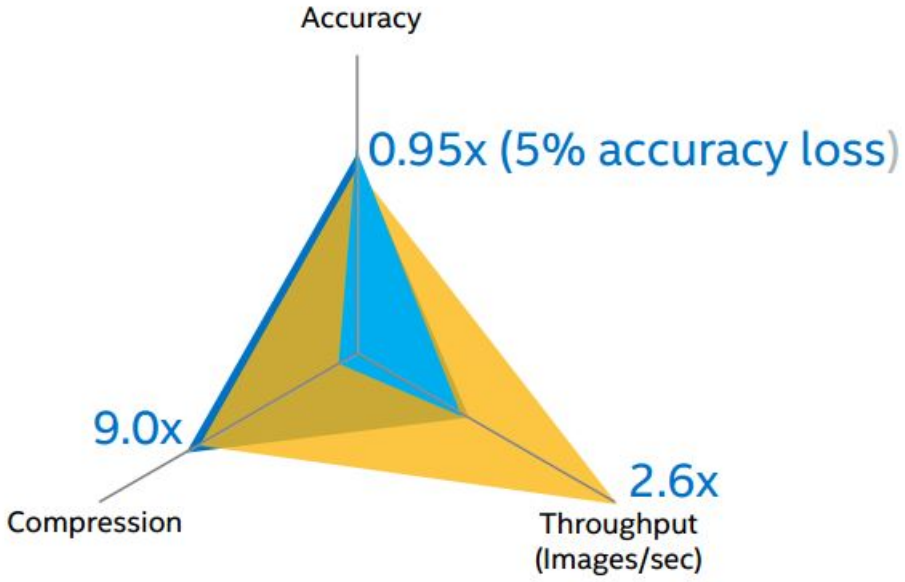

図 14. 精度を下げてスループットを上げる

つまり、デバイス要件と使用するコンテキストに応じて精度のトレードオフを決定します。

Nervana*

2016 年にインテルは、マシンラーニング業界のリーダーであり、ハードウェア・エンジニアリング、システム・ソフトウェア、マシンラーニング、クラウド分野向けの機械知能プラットフォームである Nervana* を買収しました。

Nervana* は、機械知能プラットフォームの構築に取り組んでおり、コンピューターを使用して大規模なデータセットを作成、処理し、それらに基づいて推論を行っています。Nervana* の目標は、ディープラーニングやその他のアルゴリズムを最適化して、処理を高速化することです。

しかし、マシンラーニングの目的は、人間の問題に対するソリューションを提供することです。そこで、我々はディープラーニングをどこで、どのように使用できるか考えてみました。以下は、Nervana* のマシンラーニング・プラットフォームを利用可能な分野です。

- ヘルスケア – 医療用画像は最大分野の 1 つです。MRI や CT スキャンの体積測定映像では、単一のイメージが問題を引き起こす可能性があります。医療用の静止画像は、場合によっては、一辺が最大 200,000 ピクセルあるため、1 つのイメージのみでもベンチマークのデータセットより大きくなります。そのため、計算負荷が非常に高く、効率良くスケールする必要があります。
- 農業 – ゲノムの問題、気候モデリング、(農作物を選別して収穫する) 収穫ロボットで使用できます。低レイテンシーの推論が必要とされるエッジとクラウドで特別なスケーリングが必要になります。
- 金融 – 金融機関に関連した膨大な量の IT 問題など、多くのユースケースがあります。異なる方法で多種多様な金融商品が取引される取引所は、ディープラーニングを利用することで、取引時間と取引方法に集中することができます。また、潜在的な不正行為を予測して、有害事象から取引所を保護することもできます。
- 車両 – 音声認識、ドライバーアシスト、自動運転はすべて、非常に大きなデータセットを使用しており、多くのデータを収集しています。車内のエッジで処理できるだけでなく、データセンターでも処理可能な、非常に大規模で完全なソリューションが必要とされます。

ディープラーニングはコア・テクノロジーと言えます。Nervana* では、すべての関連情報を中央のコアで処理し、各クライアントで「Google® Brain」モデルを適用しています (図 16)。

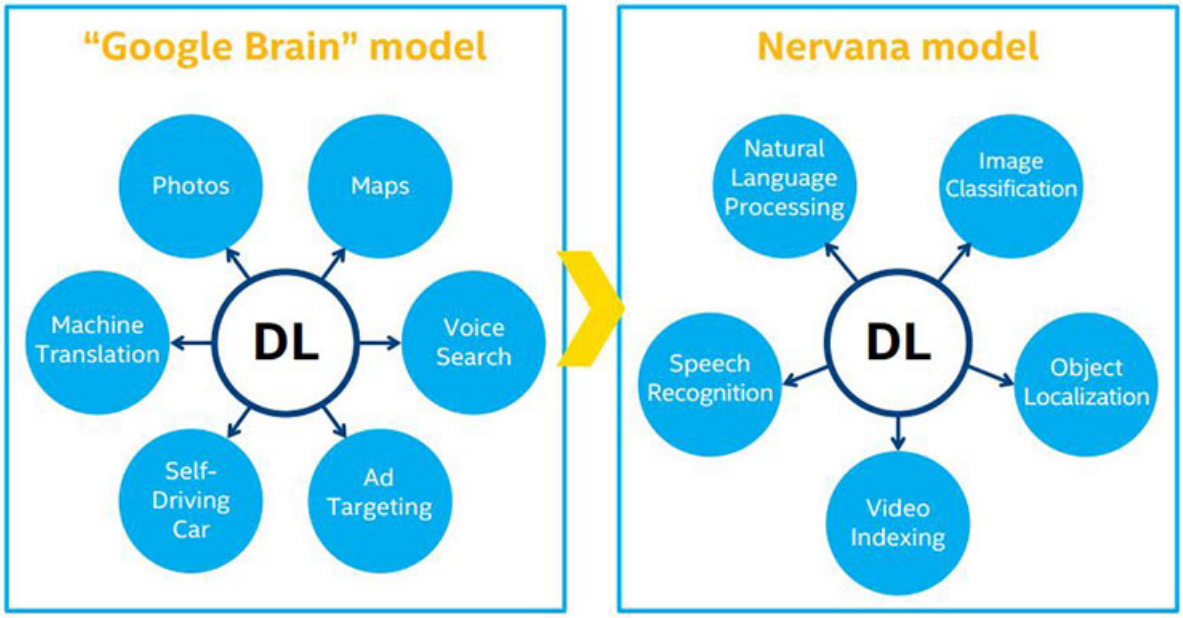

図 16. Nervana* のディープラーニング・モデルと Google® Brain モデル

ソリューション、エンジニア、配置方法、ディープラーニング製品の開発に関するあらゆるものから、ディープラーニングがコア・テクノロジーであることを理解できます。

Nervana* のディープラーニング・アプローチは、クラウドに配置されます。クラウドは、最も迅速にディープラーニングをクライアントへ配置することができます。探索的データサイエンスとトレーニング・データ・モデルは、弾性があり、スケーラブルなクラウドへの高帯域幅、低レイテンシーの接続に適しています。これが最適なディープラーニングの形態であり、クリーンなインターフェイス、低ダウンロード要件、クラウドサービスへの情報送信の容易さを備えているとさらに良いでしょう。

マシンラーニングにおけるインテルの役割

インテルは、ハードウェア/ソフトウェア・ロードマップとフレームワーク・アップデートにおいて、マシンラーニングへの対応と改善に取り組んでいます。

ハードウェア

現時点で最高密度の計算ソリューションであるインテル® Xeon Phi™ プロセッサーは、高度な並列アーキテクチャーにより、ディープラーニングに最適なプラットフォームです (図 17)。インテル® Xeon® プロセッサーと計算モデルが同じであるため、インテル® Xeon® プロセッサー向けに開発されたものはすべてインテル® Xeon Phi™ プロセッサーで実行できます。メモリー・サブシステムとファブリックの統合により、大幅なパフォーマンスの向上が得られます。

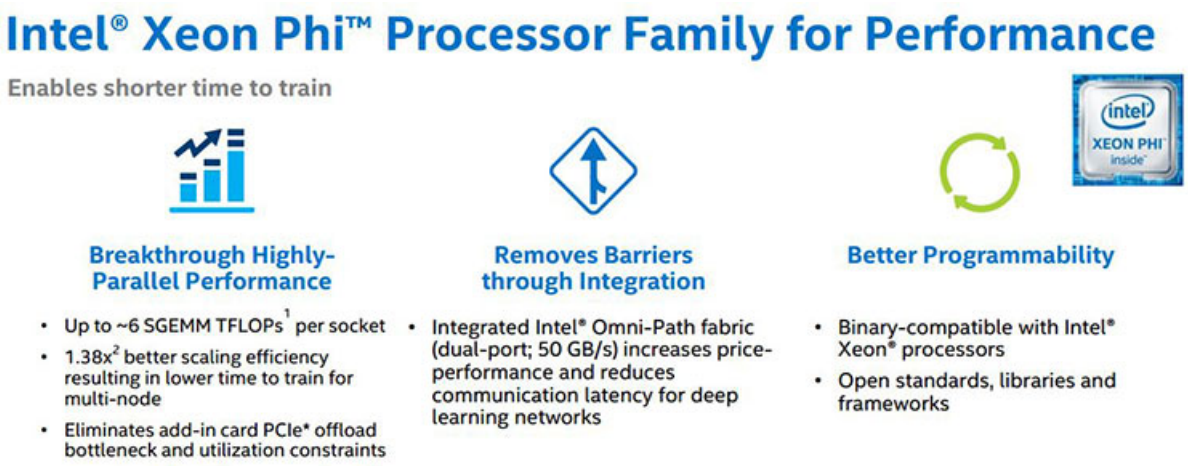

図 17. ディープラーニング・パフォーマンスにおけるインテル® Xeon Phi™ プロセッサーの利点

インテル® Xeon Phi™ プロセッサー 72xx 製品ファミリーは、プログラミングの容易さ、インテル® Xeon プロセッサーと共通のプログラミング、並列アーキテクチャーにより、高いパフォーマンスと生産性の両方を実現します。

ソフトウェア

複数のノードにわたるトレーニングと推論のスケーリングはハードウェアで対応しますが、各ノードのスケーリングはソフトウェアで対応します。インテル® マス・カーネル・ライブラリー (インテル® MKL) は、ノードレベルの最適化にとって最も重要なライブラリーです。インテル® MKL を使用することで、パフォーマンスを最大 24 倍向上できます (図 18)。

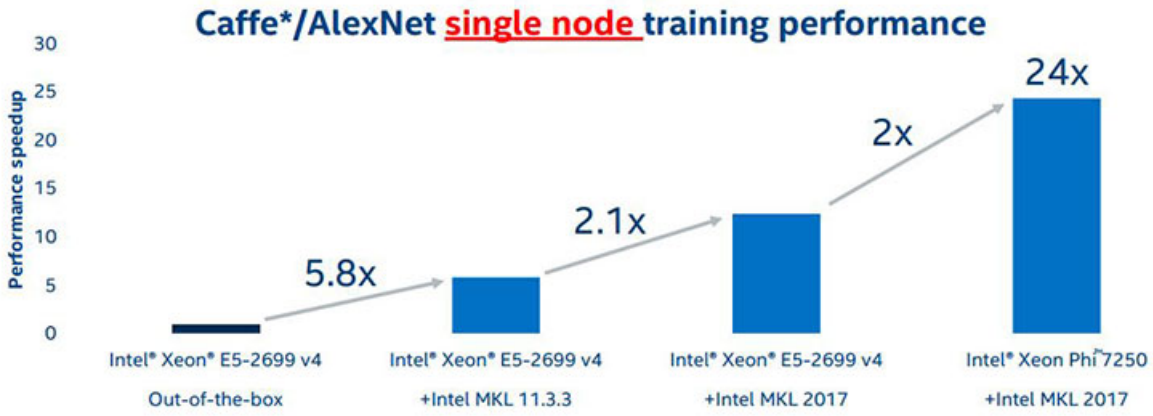

図 18. インテル® MKL による単一ノードのトレーニングのパフォーマンス向上

このパフォーマンスは、インテル® MKL のみによってもたらされたものです。インテル® MKL を利用するためにコードを変更する必要はありません。複数のノードでは、さらにパフォーマンスの向上が見込めます。今後、一般的なディープラーニング・フレームワークはすべてサポートされ、複数のノードでスケーリングできる予定です。現時点では、Caffe* とインテル® MKL によりパフォーマンスを向上できます。

インテルは、ディープラーニング・ソリューションの迅速な設計、トレーニング、配置を支援するツールをまとめた、インテル® Deep Learning SDK も提供しています。

まとめ

AI のビジョンとパワーは、日々の退屈な作業を排除するでしょう。マシンラーニングは、AI の真の力を解き放ちます。デジタルデータの急増と誰もがネットワークでつながる社会によってもたらされる計算の新しい好循環の実現に、マシンラーニングは重要な役割を果たします。自動運転車、農業、ヘルスケア、製造業などの分野では、マシンラーニングによりコンピューティング・アプリケーションの可能性が大幅に広がります。これらの分野では、未だに単純な意思決定タスクが人によって行われています。マシンラーニングと AI の進化に伴い、これらの分野は改善され、より良いものへと変化していくでしょう。

ディープラーニングの計算要件は、早急に取り組むべき課題です。複雑なモデルで数週間から数カ月かかっているトレーニング時間を、数時間から数日に短縮する必要があります。マシンラーニングの計算インフラストラクチャーは、開発者にパフォーマンスと生産性の両方を提供できなければなりません。また、クラウドの効率性を活用する必要があります。

利用可能なデータ、モデルの並列性、ノード間通信などの制限により、分散マシンラーニングのスケーリングは容易ではありません。新しいインテル® Deep Learning SDK (今後 Nervana* のクラウドスタックを統合予定) は、パフォーマンスを損なうことなく、トレーニング時間の短縮とリソースに制約のあるエッジデバイスへのモデル配置のトレードオフに伴う複雑さを隠匿/軽減します。

関連資料

1. Jongsoo Park, Sheng Li, Wei Wen, Hai Li, Yiran Chen, Pradeep Dubey. “Holistic SparseCNN: Forging the Trident of Accuracy, Speed, and Size.” http://arxiv.org/abs/1608.01409 (英語)
2. Song Han, Huizi Mao, William J. Dally. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” http://arxiv.org/abs/1510.00149 (英語)

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。