マシンラーニングで優位に立つ

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「Getting an Edge on Machine Learning」の日本語参考訳です。

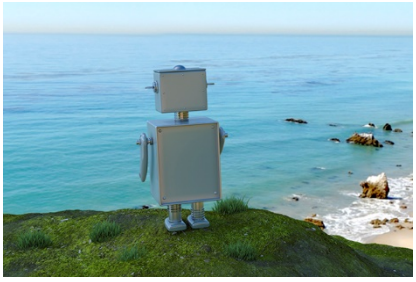
マシンラーニングの能力を発揮する

マシンラーニング・ネットワークは、近い将来組込みシステムの最新ソリューションとなる可能性を秘めています。ノイズの多いデータから特定のパターンを見つけて分類する優れた能力により、設計者は複雑なアルゴリズムの開発を回避するとともに、新たなレベルの人的操作からの独立性を備えた設計が可能になります。また、マーケティングでは、「マシンラーニング」というフレーズを好んで販促資料で使用します。

これらと次の 2 つの点を組み合わせてみましょう。第 1 にマシンラーニングは、大規模データセンターとクラウド・コンピューティング・サービス分野で誕生し、現在でもこれらの分野と関連性がありますが、組込みシステムの制約には馴染みがありません。第 2 に組込みシステムはネットワークに接続されることが増えており、組込みコンピューティングはエッジ・コンピューティングになりつつあります。もはや孤立し、自足を強いられることはありません。組込みシステムは、クラウドへの高レイテンシーで限定的な可用性の帯域幅に依存するようになりつつあります。

これらを考慮すると、マーケティング的には大きなチャンスであり、システム設計者にとっては非常に困難なパーティショニングの課題に直面します。これを受けて、マシンラーニングをエッジ環境に適合させるため、3 つの分野で大規模な取り組みが行われています (図 1)。新しいマシンラーニング・アルゴリズムは、ディープラーニング・モデルをよりコンパクトにしながら、精度と堅牢性を高めようとしています。圧縮技術は、組込みシステムのメモリーとパフォーマンスの制約に適合させるため、ディープラーニング・モデルのサイズの縮小に取り組んでいます。同時に、ハードウェア・アクセラレーター・チップは、これらの制約の緩和しようとしています。

図 1. ネットワークのエッジをのぞき込むマシンラーニング



ディープラーニング・ネットワークの内部

これらの取り組みについて説明するには、最初にディープラーニング・ネットワークの内部 (メモリー要件、データフロー、計算処理) を検証する必要があります。これは当然、圧縮とアクセラレーション手法の議論につながります。

まず構造を確認します。最も単純なディープラーニング・ネットワークは層に分割され、各層は一連のノード (人工ニューロン) で構成されます。各ノードへの入力は、前の層のすべてのノードの出力を収集し、各値に対応する重み係数を掛けたものです (図 2)。ネットワークのアーキテクチャーと特定の層に応じて、ノードは総和、非線形操作、論理操作を行います。ディープラーニング・ネットワークへの入力として、例えば 1280 x 1024 ピクセルの画像について考えてみます。最初の層では、各ノードに 1280 x 1024 の重み配列と画像の各ピクセルからの入力があります。2 番目の層のノードは、最初の層の各ノードから入力を受け取り、後続の層も同様になります。そのため、関連するデータ量と計算量は膨大になります。

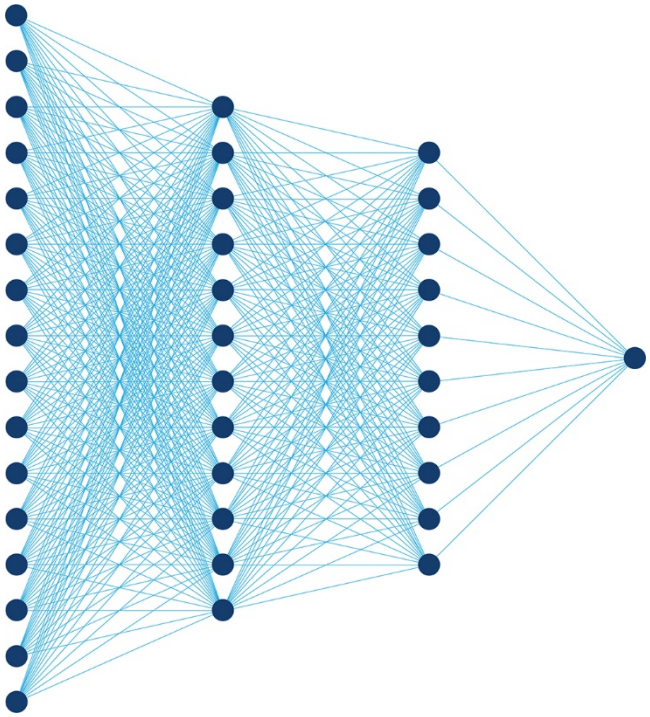


図 2. 1 つの層のノードの出力が次の層の各ノードの入力となる従来のニューラル・ネットワーク

通常、これらのネットワークでは、非線形層が総和層の間に点在しています。ネットワークが深くなるにつれて、層ごとのノード数は小さくなり、ノードの出力は人間にとって意味のあるものになります。例えば、画像分類ネットワークでは、最初の層のノードへの入力は画像のピクセル値であり、出力は画像の小さな断片が存在するかどうかを示す人間には不可解な数値です。ネットワークが深くなるにつれて、出力は意味のあるものになり、最後の層に到達します。各出力は、犬、プードル、トラクター、飛行船など、ネットワークの訓練に使用されるタグに対応します。最終的な出力は、対応するタグが入力時の画像に関連している確率を示します。入力から出力へネットワークが進むにつれて、層は小さくなる傾向にあります。入力画像全体を並列に受け入れるのに十分な幅から 1 つの各タグの出力に十分な幅へ、ネットワークは漏斗 (ろうと) のように先細りになります。

この一般的な図にはさまざまなバリエーションがあり、そのほとんどは特定のアプリケーションに対応するため開発されています。例えば、リカレント・ニューラル・ネットワーク (RNN) と長短期記憶ネットワーク (LSTM) には、内部層から前の層へノード出力をフィードバックしたり、後で使用するため出力をメモリーに保存する構造があります。このようなネットワークは、パターンを正しく分類するため文脈データを必要とする、手書き文字認識や音声認識などのアプリケーションに有用です。RNN と LSTM は、ほかの点では一般的な図によく似ており、以下に述べることのほとんどが当てはまります。

ネットワークのアーキテクチャー (層、接続、および機能) は、設計プロセスの最初に人間によって選択され、訓練済みのネットワークがうまく機能しない場合を除いて変更されません。訓練プロセスは、各ノードの重み配列の値のみを決定し、ネットワークの構造は決定しません。

ディープラーニング・ネットワークには訓練と推論の 2 つのタスクが含まれています。研究室を除くほぼすべての例では、最初にデータセンターで訓練が行われます。これにより訓練済みモデル (つまり、すべての重みが決定されたネットワーク) を作成し、推論を計算するためデータセンターやエッジで使用されます。

訓練には、人的リソースとコンピューティング・リソースの両方が必要です。最も一般的な訓練形式である教師あり学習では、最初に数千から数十万の入力レコードを選択します。例えば、交通やセキュリティー・アプリケーション向けの街の風景写真などです。次に、アプリケーションでそれぞれの風景を正確に、そして完全に分類するのに必要なラベルをアーキテクトが決定します。風景写真のタグには、モーションや脅威レベルなどの特性を

含む、風景で発生する可能性があるオブジェクトの ID が含まれる場合があります。そして、人間が手動で訓練データセットの各レコードに適切なタグを付けます。

訓練を開始します。レコードがネットワークの入力に適用され、層ごとに出力が計算されます。最後の層の結果は、その入力に対して人間が決定したタグと比較されます。次に、最後の層の各ノードの重みを調整し、出力とタグ値の間の誤差を軽減するため、アルゴリズム (通常は勾配降下法) が使用されます。次にアルゴリズムは、出力誤差をさらに軽減するため、前の層へ移動し、各ノードの入力の重みを調整します。ネットワークの最初の層までこれが繰り返されます。そして、次の入力データレコードが適用され、ネットワーク出力とタグが比較され、前述のプロセスが繰り返されます。

潜在的に数百万のノード、数万から数十万の重み、そして数十万のレコード数を含む初期の層は、データ集約型タスクです。その結果は、比較的新しい入力であっても高精度で正確に分類できる、適切に重み付けされた訓練済みネットワークです。この訓練済み推論ネットワークをエッジ・コンピューティング・プラットフォームにエクスポートします。

実用的にする

推論に必要なディープラーニング・ネットワークの計算量が訓練よりもはるかに少ない場合であっても、典型的な組込みシステムでは、特にレイテンシー要件の厳しいアプリケーションにおいて推論が大きな負荷となります。そのため、設計者は推論の計算タスクを単純化するため、アーキテクチャーの縮小、圧縮、ハードウェア・アクセラレーションの 3 つの方法を採用してきました。

このトピックに関しては、2018 年の Hot Chips カンファレンスのワークショップにおいて、MIT の Song Han 准教授が素晴らしい発表を行っています。Han 准教授の発表は 102 スライドからなるものでしたが、ここでは 3 つの方法について簡潔に説明します。

アーキテクチャー

設計者がネットワーク中の層の数と、層と層の間の接続数を減らすことができれば、推論のメモリー要件と計算量の軽減につながります。以前の経験がある場合を除いて、特定のネットワーク設計は特定の問題と訓練セットがどの程度うまく機能するか予測することはできません。特定のディープラーニング・ネットワークの設計で 16 層すべてが必要かどうか確認する唯一の方法は、いくつかの層を削除してネットワークを訓練し、テストしてみることです。この調査のデメリットは、設計者が精通しているネットワーク・アーキテクチャーを使用し続ける傾向があることですが、調査によって大幅な節約が可能かもしれません。

その一例が静止画像の分類 (有名な ImageNet チャレンジ) です。ディープラーニング・ネットワークの典型的なケースでは、各ノードは前の層のすべてのノードから重み付けされた入力を受け取りますが、画像分類の研究者はこれを大幅に単純化する畳み込みニューラル・ネットワーク (CNN) を発見しました (図 3)。初期の層で CNN は全結合ノードを小さな畳み込みエンジンに置き換えます。各入力の重みに代わって、畳み込みエンジンには小さな畳み込みカーネルのみがあります。入力画像でカーネルを畳み込み、特徴マップ (画像の各ポイントにおける画像とカーネルの間の類似性を示す 2D 配列) を生成します。この特徴マップは、非線形化されます。畳み込み層の出力は 3D 配列です (層内の各ノードの 2D 特徴マップ)。この配列に、解像度を下げて 2D 特徴マップのサイズを縮小するプーリング操作を適用します。

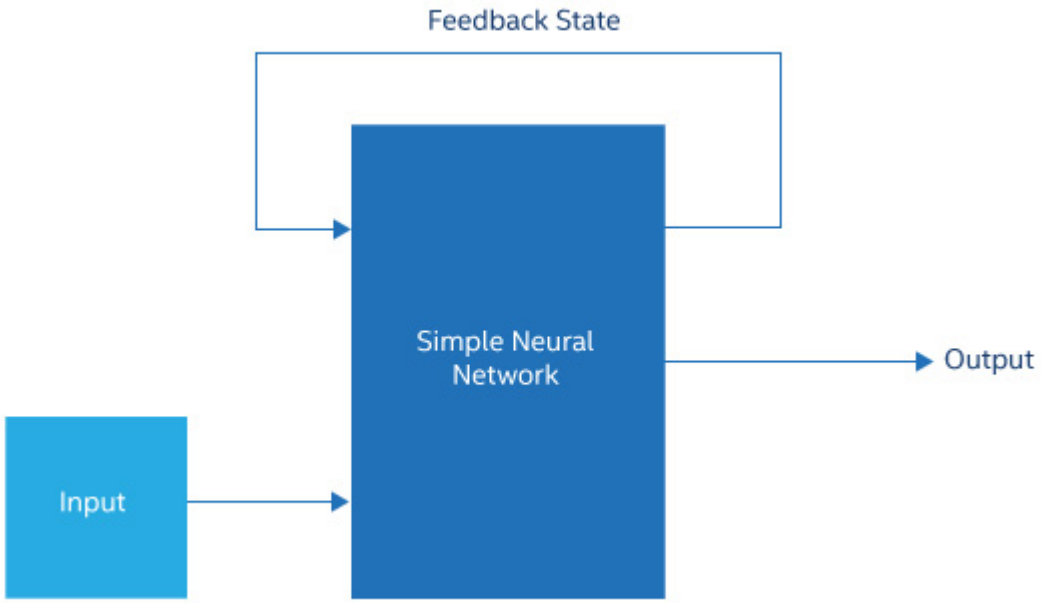


図 3. 通常 RNN は中間状態または結果の一部を入力にフィードバックする単純なネットワーク

最近の CNN には、多くの畳み込み層があり、その後にプーリング層が続きます。ネットワークの出力側に向かって、畳み込み層とプーリング層が終わると、残りは全結合層です。そのため、ネットワークは入力側から出力側に向かって細くなり、各タグの出力を生成するのに十分な幅の全結合層で終了します。同じ深さの全結合ディープラーニング・ネットワークと比較すると、重み、接続、ノード数を大幅に軽減できます。

圧縮

マシンラーニング・コミュニティーでは、「圧縮」という用語は従来のデータ圧縮とは異なる意味で使用されます。ここでは、圧縮は推論の生成に必要な計算の量と困難さを軽減する各種手法で構成されます。

そのような手法の 1 つが枝刈り (pruning) です。ディープラーニング・ネットワークの訓練では、通常、重み行列にはゼロや非常に小さな値が含まれます。実際には、その重みで乗算される入力は計算する必要がないため、推論の計算を表すデータフロー・グラフからその枝全体を刈り取る (pruning) ことができます。経験から、ネットワークを枝刈り (pruning) して再度訓練することで、精度を向上できます。

圧縮の別の手法として、重みのビット数を減らします。多くのデータセンター・サーバーは、すべての値を単精度浮動小数点として保持しますが、研究者ははるかに低い精度の重み (わずか 2 ～ 3 ビット) で 32 ビット浮動小数点と同じ精度を維持できることを発見しました。同様に、非線形化後のノード出力も数ビットで十分な場合があります。推論モデルがサーバー上で実行される場合、これはほとんど効果がありません。しかし、実際に 2 ビットまたは 3 ビットの乗数を効率良く実装し、この圧縮を利用できる MCU や FPGA アクセラレーターでは非常に有用です。

全体として、枝刈り (pruning)、有意性の減少、および関連する手法によって一部のケースでは推論作業が 1/20 ～ 1/50 に軽減されました。訓練済みネットワークは、一部のエッジ・コンピューティング・プラットフォームでも利用できる可能性があります。圧縮が十分ではない場合、設計者はさまざまなハードウェア・アクセラレーションを利用することができます。

ハードウェア

推論に必要な計算は、それほど多様でもなければ、複雑でもありません。主に、入力に重みを掛けて、ノードごとに結果を合計する、多数の積和演算 (乗累算または MAC) です。すべての負の値をゼロに設定する正規化線形関数 (ReLU)、非線形変換を行う双曲線正接関数またはシグモイド関数、プーリング処理を行う max 関数などの非線形関数もあります。全体として、典型的な線形代数のワークロードに似ています。

ハードウェアにおいては、スーパーコンピューターからのアイデアが流用されます。最も簡単な方法は、入力、重み、出力をベクトルとし、汎用 CPU に実装されるベクトル SIMD ユニットを使用します。さらに高速にするため、GPU で多数のシェーディング・エンジンを使用しています。スラッシングや高いミス率を回避するため、入力、重み、出力データを GPU のメモリー階層に配置することは容易ではありませんが、GPU はデータセンターのディープラーニングにおいて最も広く使用される非 CPU ハードウェアとなりました。最近の世代の GPU は、そのような用途に対応するため進化しており、浮動小数点シェーディング・ユニットを補完する小さなデータ型や行列-数学ブロックが追加されています。

これらの適応は、命令のフェッチとデコードを軽減/排除、データ移動の軽減、並列処理を最大限に利用、および圧縮を使用する、というアクセラレーション・ハードウェア設計者の基本手法を示しています。これらすべてを互いに干渉せずに行う必要があります。

これらの手法を使用するアーキテクチャー上のアプローチがあります。最も簡単なのは、ダイ上の乗算器、加算器、小さな SRAM ブロックをインスタンス化して、それらをネットワークオンチップでリンクすることです。これは、推論を実行するリソースを提供しますが、演算器との間でデータを効率良く受け渡すという重要な課題をプログラマーに課します。このような設計は、不可解なプログラミングの課題に根ざす過去の多くの超並列コンピューティング・チップから派生したものです。

Google* の Tensor Processing Unit* (TPU) などのチップは、ディープラーニング・ネットワーク固有の構造に応じて演算器を構成することで、アプリケーションに基づいたアプローチを採用しています。そのようなアーキテクチャーは、ネットワークの入力重み付け乗算を非常に大きな行列乗算と見なし、それらを実行するためハードウェア行列乗算器を作成します。TPU では、オペランドがユニットからユニットへアレイを自然に流れる、乗算器のシストリック・アレイで乗算が実行されます。アレイの周辺には活性化と重みの値を供給するバッファーが配置されており、その後に活性化関数とプーリング・ハードウェアが続きます。

行列操作を自動的に行うようにチップを構成することで、TPU ではプログラマーが詳細なレベルで演算器や SRAM のデータ移動をスケジュールしなくても済みます。プログラミングは、入力と重みを行列にまとめ、ボタンを押すだけの簡単なものになります。

しかし、そこには問題もあります。前述のとおり、枝刈り (pruning) は疎行列になる可能性があり、これを単純に TPU などのデバイスに供給すると、無意味な乗算や加算が多発します。ハードウェアを効率良く活用するには、モデル開発の圧縮フェーズで、これらの疎行列を小さな密行列に並べ替える必要があります。

3 つ目のアプローチでは推論タスクを一連の行列乗算ではなく、データフロー・グラフとしてモデル化します。アクセラレーターは、データが一方から入り、構成可能なリンクを介して処理要素のグラフのようなネットワークを流れて出力として現れる、データフロー・エンジンとして設計されています。このようなアクセラレーターは、枝刈り (pruning) したネットワークに必要な操作のみを実行するように構成できます。

アーキテクチャーを選択したら、次に実装方法を検討します。コストとスケジュールの観点から、多くのアーキテクチャーの開発は FPGA で開始されます。ディープラーニング・モデルが大きく変化することが想定され、単独のアクセラレーター設計ではすべての変更に対応できない場合など、一部は引き続き FPGA が使用されます。しかし、モデルの変更がわずかな場合 (例えば、層の配置や重みの変更など)、ASIC または CPU 統合アクセラレーターが推奨されます。

エッジ・コンピューティングとその制約に話を戻します。マシンラーニング・ネットワークが一連のサーバーで実行される場合、サーバー CPU、GPU、FPGA、または ASIC アクセラレーター・チップでの実行がすべて選択肢となります。しかし、制約のある環境で実行しなければならない場合 (例えば、工場に設置されている機械、ドローン、カメラなど)、小型の FPGA や ASIC が必要になります。その一例として、インテルは最近、PC の USB ポートに差し込むメモリースティック形式の ASIC アクセラレーター・チップを発表しました。

ハンドセット、低電力 ASIC、アプリケーション・プロセッサーに内蔵されたアクセラレーター・ブロックなど、非常に制約の厳しい環境では、SoC が唯一の選択肢です。これまでのところ、これらの制約により設計者は単純な乗算器アレイを使用する傾向にありましたが、ニューロモルフィック設計の優れた電力効率は、次世代の組込みアクセラレーターにとって非常に重要になる可能性があります。

いずれにせよ、マシンラーニングはもはやデータセンターだけのものではありません。推論はエッジに移行しつつあります。研究者が従来のディープラーニング・ネットワークから、継続的な教師なし学習などの概念 (新しい環境に素早く適応できる可能性があるだけでなく、大量のリソースを必要とする可能性もある) に移行するにつれて、エッジにおけるマシンラーニングの問題がアーキテクチャー開発の最先端となることは確実です。