Vol 57, No 2—April 2023

ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™

AnalogDialogue

機械学習とは何か？【Part 3】畳み込みニューラル・ネットワークをハードウェアで実現する

著者：Ole Dreessen、フィールド・アプリケーション担当スタッフ・エンジニア

概要

本連載では、Part 1とPart 2の2回にわたり、パターンの認識や対象物の分類に使用される畳み込みニューラル・ネットワーク（CNN：Convolutional Neural Network）の特性とアプリケーションについて説明してきました。今回（Part 3）は、このCNNをハードウェアで実現する方法について説明します。具体的には、AI（人工知能）対応のマイクロコントローラ「MAX78000」を使用する方法を紹介します。この製品は、ハードウェアで実現されたCNN用のアクセラレータを搭載しています。これを利用すれば、IoT（Internet of Things）アプリケーションのエッジにおいてAIを利用することが可能になります。なお、必要に応じて本連載の前2回の記事「機械学習とは何か？【Part 1】畳み込みニューラル・ネットワークの基本」と「機械学習とは何か？【Part 2】畳み込みニューラル・ネットワークのトレーニング」も参照してください。

はじめに

一般に、AIを利用するアプリケーションには、サーバ・ファームや高価なFPGAなどが必要です。これは、大量の電力を供給する必要があるということを意味します。つまり、その種のアプリケーションでは、消費電力とコストを低く抑えつつ演算能力を高めることが大きな課題になります。ただ、現在、AIベースのアプリケーションには劇的な変化が訪れつつあります。なぜなら、強力かつインテリジェントなエッジ・コンピューティングを利用できるようになったからです。従来はファームウェアをベースとして演算が行われてきました。それに対し、ハードウェアをベースとするCNNアクセラレータを利用すれば、圧倒的なスピードとパワーが得られます。つまり、演算性能の面で新時代が訪れつつあるということです。インテリジェントなエッジ技術を利用すれば、センサー・ノードにおいて判断を下せるようになります。そうすれば、5GやWi-Fiのネットワークを利用して転送しなければならないデータの量を大幅に削減できます。その結果、従来は実現が不可能だった新たな技術や独自のアプリケーションの開発が強力に推進されるようになりました。例えば、煙検知器／火災検知器を遠隔から管理したり、センサー・ノードによって現場で環境に関するデータを分析したりすることが可能になります。しかも、電源としてバッテリを使用し、そうしたデバイスを何年も運用できるようになるのです。では、そのような機能を具現化するには、具体的にはどのようにすればよいのでしょうか。その1つの答えとして、本稿ではAIマイクロコントローラというハードウェアを使用することで、CNNを実現する方法を紹介します。

超低消費電力のCNNアクセラレータを搭載したAIマイクロコントローラ

MAX78000は、超低消費電力のCNNアクセラレータを搭載したAIマイクロコントローラです。この先進的なSoC（System on Chip）は、リソースに制約があるエッジ・デバイスやIoTアプリケーションに対し、消費電力を極めて少なく抑えつつCNNの機能を提供します。アプリケーションの例としては、物体の検知／分類、音声の処理／分類、ノイズのキャンセル、顔認識などが挙げられます。また、心拍数のような健康状態を表す信号を分析するための時系列データの処理、マルチセンサーによる解析、予知保全などにも活用されます。

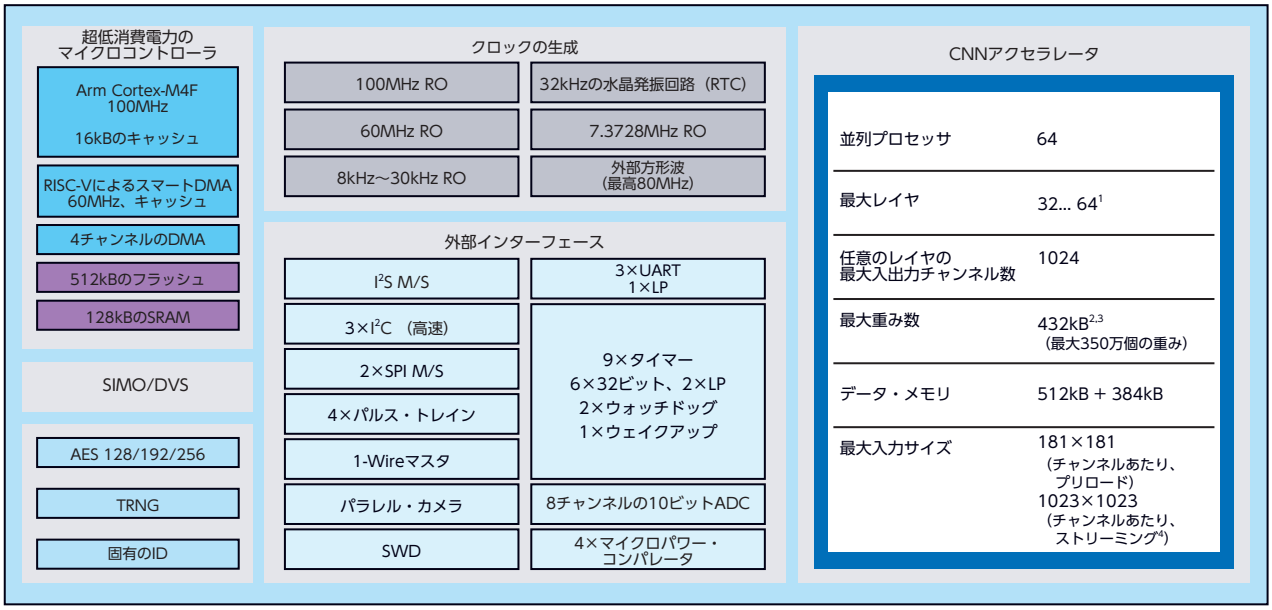

図1. MAX78000のブロック図

図1に、MAX78000のブロック図を示しました。同ICの高い性能は、Arm® Cortex®-M4Fコアによって支えられています。このコアは浮動小数点ユニットを内蔵しており、最高100MHzで動作します。また、同ICはアプリケーションに十分なメモリ・リソースを提供するために、512kBのフラッシュ・メモリと128kBのSRAMを搭載しています。加えて、I²C、SPI（Serial Peripheral Interface）、UART（Universal Asynchronous Receiver Transmitter）や、音声アプリケーションにとって重要なI²Sなど、複数のインターフェースを備えています。更に、60MHzで動作するRISC-Vコアも内蔵しています。このコアは、個々のペリフェラル・ブロックやメモリ（フラッシュ・メモリとSRAM）との間でデータのコピーを行うなど、スマートなDMA（Direct Memory Access）エンジンとして機能します。このコアによって、AI用のアクセラレータ向けに、センサーで取得したデータの前処理が行われます。その間、Armコアはディープ・スリープ・モードに移行することができます。必要があれば、推論結果に応じ、Armコアに対して割り込み（トリガ）をかけることが可能です。その後、Armコアはアプリケーションのメインの作業に相当する処理を実行したり、センサーのデータをワイヤレスで送信したり、ユーザに対する通知を行ったりします。

「MAX7800xシリーズ」のマイクロコントローラには大きな特徴があります。それは、CNNの推論を実行するためのハードウェア・アクセラレータ・ユニットを備えていることです。このユニットは、一般的なマイクロコントローラのアーキテクチャやペリフェラルとは一線を画しています。このユニットにより、必要なすべてのパラメータ（重みとバイアス）に加え、完全なCNNのモデル・アーキテクチャをサポートすることができます。64個の並列プロセッサを搭載しており、パラメータを格納するための442kBのメモリと入力データ用の896kBのメモリを備えています。モデルとパラメータはSRAMに格納されるので、ファームウェアによる調整を行うことで、ネットワークをリアルタイムに適応させることが可能になります。同SRAMは、モデルで1、2、4、8ビットの重みのうちどれを使用するかによって、最大350万個のパラメータに対応することができます。アクセラレータがメモリに関連する機能を内蔵していることから、連続した数学的な演算のたびにマイクロコントローラのバス構造を介してパラメータをフェッチする必要はありません。そのようなフェッチの処理は遅延が大きく、多くの電力を消費するので、大きな問題になります。MAX7800xシリーズのアクセラレータは、プーリングの機能に応じ、32または64のレイヤをサポートします。プログラムが可能な画像の入出力サイズは、各レイヤで最大1024×1024ピクセルです。

ハードウェア・ベースのCNNが消費電力と推論速度にもたらす効果

CNNによる推論は、行列形式の大規模な線形方程式で構成される複雑な演算タスクです。Arm Cortex-M4Fの能力を活かせば、組み込みシステムのファームウェア上でCNNの推論を行うこともできます。但し、考慮すべきいくつかの欠点があります。ファームウェアをベースとする推論をマイクロコントローラ上で実行すると、大量の電力と時間が必要になります。演算に必要なコマンドと関連するパラメータのデータを、メモリから取得した後、中間結果を書き戻す必要があるからです。

ここで表1をご覧ください。これは、3つの異なる手法（シナリオ）によるCNNの推論時間と消費電力を比較したものです。ここで例として使用しているモデルは、MNIST（Mixed National Institute of Standards and Technology）データベースのトレーニング・セットを使用して開発したものです。そのトレーニング・セットは、手書きの数字を認識するためのトレーニングに使用されています。開発したモデルは、視覚的な入力データから数字と文字を分類して正確な出力結果を得るためのものです。この例では、各シナリオを使用した場合に費やされた推論時間を測定し、消費電力と速度を比較しました。

表1. CNNの推論時間と1回の推論あたりの消費電力。MNISTのデータセットを使用し、3種のシナリオについて評価を行いました。

シナリオ	推論時間〔ミリ秒〕	1回の推論あたりの消費電力〔µWs〕
(1) MAX32630、MNISTネットワークをファームウェアで推論	574	22887
(2) MAX78000、MNISTネットワークをハードウェアで推論	1.42	20.7
(3) MAX78000、MNISTネットワークをハードウェアで推論、消費電力の削減に向けて最適化	0.36	1.1

表1に示した（1）のシナリオでは、「MAX32630」が備えるArm Cortex-M4F（96MHzで動作）を使用して推論の演算を行いました。一方、（2）のシナリオでは、MAX78000が備えるハードウェア・ベースのCNNアクセラレータを使用して演算を実行しました。ネットワークへの入力として視覚的なデータが提示されてから結果が出力されるまでの時間（推論速度）を見ると、（2）では（1）と比べて1/400に短縮されています。また、1回の推論に必要な消費電力は1/1100に削減されています。（3）のシナリオでは、MNISTネットワークに対する1回の推論に伴う消費電力を抑えられるように最適化を行いました。この場合、結果の正確さは99.6%から95.6%に低下します。その一方で、ネットワークは大幅に高速化され、1回の推論あたりわずか0.36ミリ秒の時間しかかかりません。消費電力も、1回の推論あたりわずか1.1µWsまで削減されています。単3アルカリ電池を2本（トータルの電力量は6Wh）使用するアプリケーションであれば、500万回の推論を実行できることになります（他の回路で消費する電力は考慮していません）。

これらのデータは、ハードウェア・アクセラレータを利用した場合の演算性能の高さを表しています。ハードウェア・アクセラレータをベースとする演算は、商用電源などから継続的な給電を受けることができないアプリケーションにとっては非常に重要な意味を持ちます。MAX78000を採用すれば、大量の電力、インターネットへのブロードバンド・アクセス、長い推論時間を必要とすることなくエッジで処理を実現できます。

MAX78000を採用したアプリケーション

MAX78000を使用すれば、多種多様なアプリケーションを実現できる可能性があります。ここでは、1つの具体的な例について検討することにしましょう。そのアプリケーションでは、バッテリ駆動のイメージ・センサー・システムを実現すると仮定します。そのシステムでは、猫がイメージ・センサーの視野に入ったら、それを検出します。その上で、デジタル出力を利用して猫用のドアを解錠し、猫が家に入れるようにします。

図2は、上記のイメージ・センサー・システムをブロック図として示したものです。このシステムでは、RISC-Vコアによって一定の時間が経過するごとにイメージ・センサーを起動します。同センサーによって画像データを取得したら、それらをMAX78000が備えるCNNにロードします。猫の認識確率があらかじめ設定した閾値を超えると、猫用のドアが開きます。その後、システムはスタンバイ・モードに戻ります。

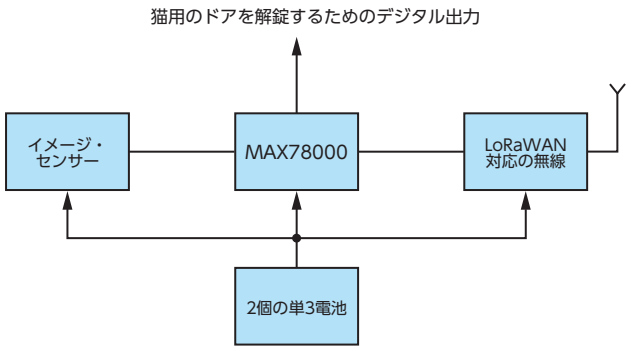

図2. イメージ・センサー・システムのブロック図

開発環境と評価キット

MAX78000をベースとし、エッジでAIを活用するアプリケーションは、どのようにして開発すればよいのでしょうか。その開発プロセスは、以下の2つのフェーズに分けることができます。

【フェーズ1】AIに関する開発。定義、トレーニング、ネットワークの量子化を行う

【フェーズ2】Armのファームウェアの開発。フェーズ1で開発したネットワークとパラメータをC/C++ベースのアプリケーションに組み込む。また、アプリケーション用のファームウェアを開発し、テストする

開発プロセスでは、最初にAI用のモデリング、トレーニング、評価を実施します。この段階では、PyTorchやTensorFlowといったオープンソースのツールを活用することができます。GitHubのリポジトリでは、ユーザがPyTorchの開発環境を使用し、MAX78000のハードウェアの仕様を考慮しながら、AIアプリケーション用のネットワークの構築とトレーニングの計画立案に役立つ包括的なリソースを提供しています。同リポジトリには、いくつかの簡単なAIネットワークや顔認識アプリケーションなどが用意されています。

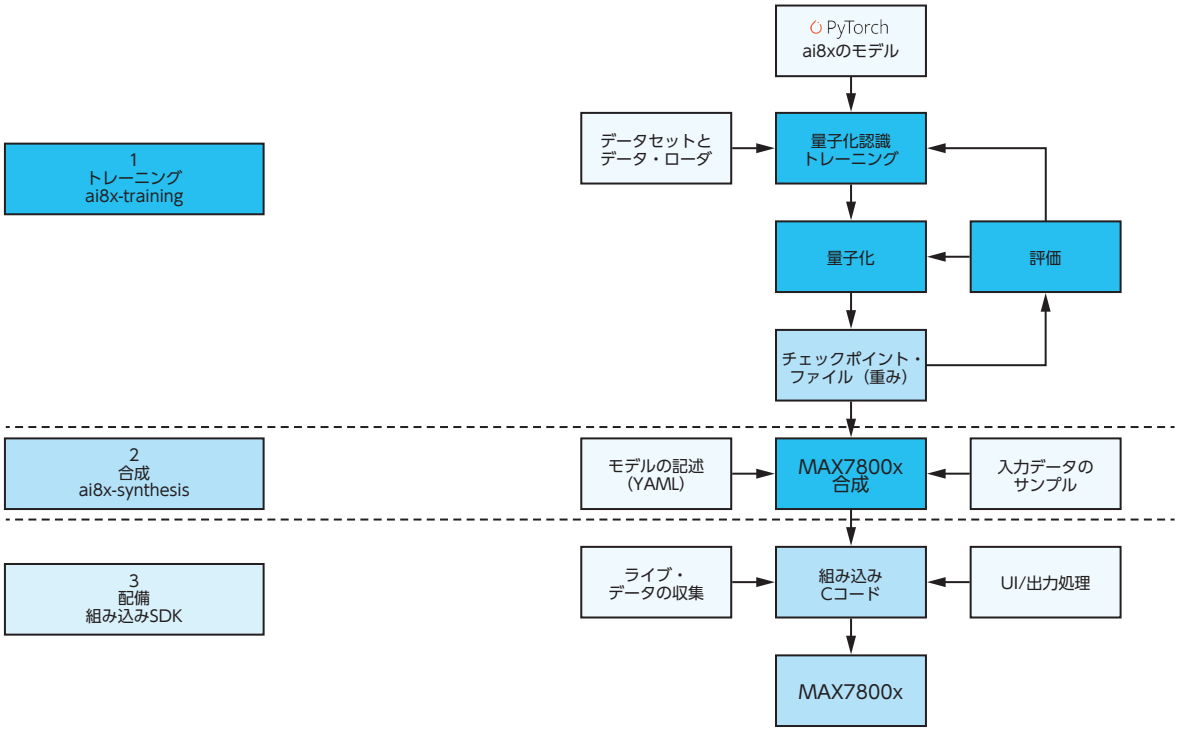

図3. AIアプリケーションの開発プロセス

図3は、PyTorchを使用する場合の標準的な開発プロセスを示したものです。最初に行うのは、AIネットワークをモデル化することです。MAX7800xシリーズの全製品が、PyTorch環境で利用可能なすべてのデータ操作をサポートするハードウェアを搭載しているわけではありません。この点には注意してください。そのような理由から、最初に、アナログ・デバイセズが提供するai8x.pyというファイルをプロジェクトにインクルードする必要があります。このファイルには、MAX78000を使用するために必要なPyTorchのモジュールと演算子が含まれています。この設定に基づき、ネットワークを構築した後、トレーニング・データを使用してトレーニング、評価、量子化を行います。このステップの結果は、最終的な合成処理用の入力データを含むチェックポイント・ファイルとなります。この最後の処理では、ネットワークとそのパラメータがハードウェア・ベースのCNNアクセラレータに適合する形式に変換されます。なお、ネットワークのトレーニングは、どのようなコンピュータ（ノート型、サーバなど）を使用することでも実施できます。しかし、CUDA（Compute Unified Device Architecture）対応のグラフィック・カードがサポートされていない場合、長い時間を要する可能性があります。小規模のネットワークであっても、数日から数週間かかることも十分にありえます。

開発プロセスのフェーズ2では、アプリケーション用のファームウェアを構築します。そのファームウェアには、データをCNNアクセラレータに書き込み、その結果を読み取る仕組みを盛り込みます。フェーズ1で作成したファイルは、#includeディレクティブによってC/C++プロジェクトに統合します。EclipseやGNU Toolchainといったオープンソース・ツールをマイクロコントローラ用の開発環境として使用することも可能です。アナログ・デバイセズは、ソフトウェア開発キット「Maxim Micros SDK (Windows)」を、必要なコンポーネントと構成（コンフィギュレーション）をすべて含むインストーラとして提供しています。このキットには、ペリフェラル用のドライバの他、アプリケーションの開発を容易にするためのサンプルや説明書も含まれています。

エラーが発生しない状態でプロジェクトのコンパイル／リンクが完了したら、ハードウェア上で評価を実施することができます。アナログ・デバイセズは、そのためのものとして2種類のハードウェア・プラットフォームを開発しました。図4に示した「MAX78000EVKIT」と図5に示した「MAX78000FTHR」の2つです。後者は、やや小型／軽量のフォーム・ファクタで実現されています。なお、両ボードにはVGAカメラとマイクが付属しています。

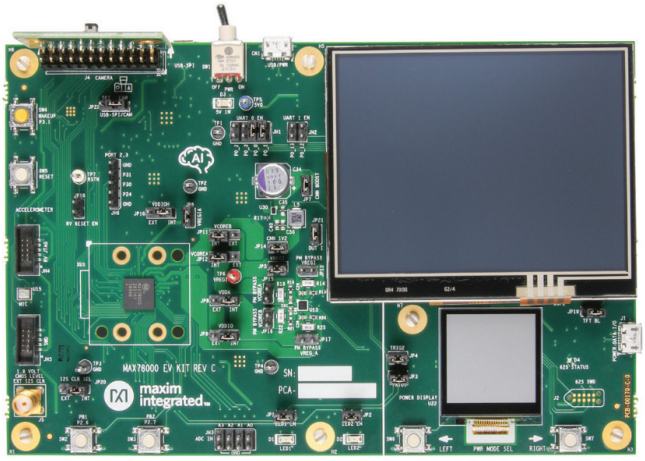

図4. MAX78000EVKITの外観

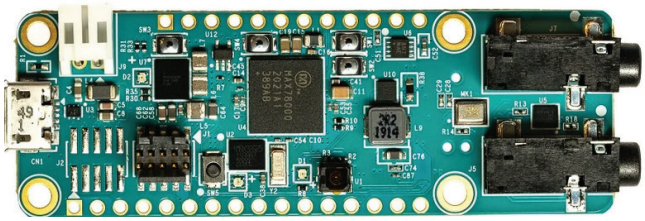

図5. MAX78000FTHRの外観

まとめ

従来、AIアプリケーションを動作させるには、サーバ・ファームや高価なFPGAといった消費電力の多いハードウェアが必要でした。CNNアクセラレータを搭載したMAX78000は、それらとは一線を画すソリューションです。同製品を採用すれば、1個のバッテリによって長期間にわたりAIアプリケーションを運用することができます。消費電力が極めて少なく、エネルギー効率に優れることから、エッジにおいてAIを活用することが可能になります。従来は不可能だったエキサイティングなAIアプリケーションをエッジで運用できる可能性が広がっているのです。詳細については、「超低消費電力の人工知能（AI）マイクロコントローラ」をご覧ください。

参考資料

「Session 2 - AI at the Edge: A Practical Introduction to Maxim Integrated's MAX78000 AI Accelerator（セッション2 - エッジにおけるAI：AI用のアクセラレータ「MAX78000」の実用に向けた手引き）」Analog Devices

Video Series: Understanding Artificial Intelligence（ビデオ・シリーズ：AIについて理解する）、Analog Devices

著者について

Ole Dreessenは、アナログ・デバイセズでフィールド・アプリケーションを担当するスタッフ・エンジニアです。2014年に入社しました。それ以前は、Avnet Memec、Macnicaで通信技術や高性能のマイクロプロセッサのサポート業務に従事。マイクロコントローラとセキュリティに関する広範な専門知識を有しています。カンファレンスやイベントなどにおけるプレゼンテーションの経験も豊富です。余暇にはChaos Computer Clubの熱心なメンバーとして活動。リバース・エンジニアリングや組み込みセキュリティなどに関する取り組みを行っています。

AD5704-0-4/23