C-012

FPGA タブレットを用いた Deep Learning アプリケーションの高速化の検討

A Study of Accelerating a Deep Learning Application using an FPGA Tablet

佐藤　知哉†　　成見　哲†
Tomoya Sato　　Tetsu Narumi

1. はじめに

近年、スマートフォンなどの携帯型端末の普及や、利用方法が多様化したことにより、アプリケーション自体の機能や性能への需要も上がってきている。この問題を解決するにあたり、ソフトウェア側からの解決策とハードウェア側からの解決策が考えられる。ソフトウェア側からの解決策では、処理の並列化やアルゴリズムの改善などが考えられる。ハードウェア側からの解決策では、専用の周辺回路やモジュール、チップを搭載するなどの端末自体のスペックや機能を向上する方法が考えられる。

一方で、近年、Deep Learning という多層で構成されたニューラルネットワークによる機械学習方法に注目が集まっている。Deep Learning では、画像や音声などの大規模なデータを扱うことから、CPU だけでの処理では扱いづらいこともあり、GPU を用いた高速化を伴うことが多い。しかし、GPU は携帯端末に搭載されておらず、広く普及しているスマートフォンやタブレット端末が持つスペックでは、実現性は低い。

そこで、本研究では FPGA を用いた Android タブレット端末を用いることで、Deep Learning 手法を実装したアプリケーションの高速化に関して検討を行う。

2. FPGA を用いたタブレット端末

塩谷の研究[1]では、Xilinx 社が発売している Programmable SoC を搭載した ZedBoard [2]をベースにした Android タブレットの開発を行っている。チップ上の ARM コアで Android OS を動作させることにより、タブレット端末として利用できる。また、パーシャルリコンフィギュレーション機能を用いることで、OS 起動中に FPGA の回路の一部を書き換えることが可能である。これにより、動かすアプリケーションごとに専用回路を用意し、それらを必要に応じて書き換えていくことにより、アプリケーション全体の高速化を目指している。このタブレット端末システムはハードウェア部、ソフトウェア部、ソフトハード部の３つの要素で構成されている。本研究では、このタブレット端末を用いて Deep Learning アプリケーションの高速化の検討を行う。

2.1 ハードウェア部

ハードウェア部は電源部分、制御部分、表示部分、入力部分、無線部分、外装部分から成り立っており、内部バッテリやタッチディスプレイなどのタブレット端末に必要な機構を備えている。また、通常のタブレット端末と異なり、外部 IO として ZedBoard の PMOD を二つ利用できる。

†電気通信大学 大学院 情報・通信工学専攻

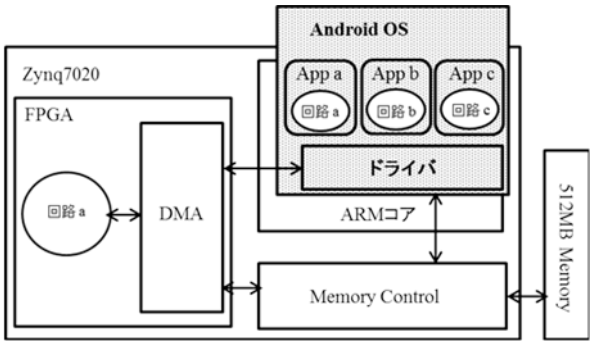

図 1　FPGA タブレットシステム図

2.2 ソフトウェア部

ソフトウェア部は、OS 部分、ドライバ部分、API 部分、アプリ部分に分けられる。ドライバや API は、FPGA への回路の書き込みやメモリ操作を行うために用意されており、アプリケーション側で API を呼び出すことで操作を行う。API は C と Java の両方で用意されている。

2.3 ソフトハード部

ソフトハードは、主に部分再構成が可能な回路のことを呼んでいる。OS からの映像信号や外部 IO を使用する為の回路、処理回路でデータを扱うための DMA 回路、グローバルクロックを使用するための回路がある。また、アプリの持つ専用回路を書きこむための領域も備えている。アプリと専用回路とのやり取りは DMA コントローラとメモリコントローラで行われる(図 1)。

3. Deep Learning アプリケーション

Deep Learning のひとつとして、畳み込みニューラルネットワークと呼ばれる深層学習方法がある。本研究では、畳み込みニューラルネットワークを Android アプリケーションとして実装する。

3.1 ニューラルネットワーク

ニューラルネットワーク（以下 NN）とは、入力層からのデータが、層状に並び隣接する層間のユニットが結合している構造を通って、出力層に伝播するネットワーク構造のことである。ネットワークを構成するユニットはそれぞれ複数の入力に対して重みを掛けて、バイアスを足した値を出力する。この重みを適当な値に設定することで任意の出力を得ることができる。重みは誤差逆伝播法を用いて各

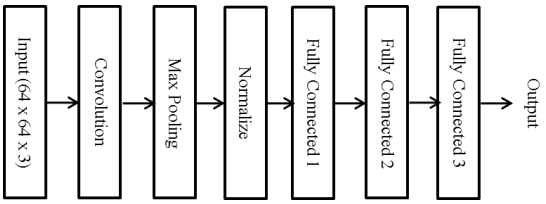

図 2　実装する畳み込みニューラルネットワーク

層の出力誤差を計算し、確率的勾配降下法により値の修正を行う。このとき、誤差の逆伝播中に勾配が消失してしまう問題や浅い局所最適解に収束してしまうなどの問題がある。本研究では、ドロップアウト[3]を用いて過適合を避けるようにしている。

3.2 畳み込みニューラルネットワーク

畳み込み NN は、通常の NN に畳み込み層とプーリング層を加えた NN であり、主に画像認識に応用されている。畳み込み層では畳み込むフィルタ値を学習する。この時、畳み込み層では重みを共有しており、同一チャネルの全ユニットで共有している。プーリング層では局所受容野の考え方を反映しており、上位層のユニットは決まった局所的な下位層のユニットと接続されている。

3.3 実装するネットワーク

本研究で実装する畳み込み NN を図 2 に示す。FPGA タブレット上の Android OS が管理するメモリの都合により、各層のパラメータ数に制限がある。そこで、畳み込み層を一層だけにし、入力サイズや中間層のチャネル数を小さくすることでパラメータ数の増大を防いでいる。

各層の活性化関数には出力層を除いて正規化線形関数を選択している。出力層にはソフトマックス関数を用いており、出力結果は各カテゴリへの分類結果である。また、多クラス分類問題を想定し、誤差関数には交差エントロピーを用いた。プーリング層では最大プーリングを採用した。

4. 計算時間モデル

３章で構築したネットワークでは、畳み込み層の演算のウェイトが大きい。そのため、この部分に関して専用回路化を行い、アプリケーションの高速化を目指したい。一般的に、処理の専用回路化の規模が大きくなるとより高速になるが、開発コストが増大するというトレードオフが存在する。そこで、本研究では専用回路化するにあたって、畳み込み層の順伝播関する処理に焦点を当てることにした。

4.1 モデル式

畳み込み層の順伝播に関する処理時間は式(4.1.1)となる。

$$T = T_{CPU} + T_{FPGA} + T_{COM} \qquad (4.1.1)$$

ここで、$T_{CPU}, T_{FPGA}, T_{COM}$はそれぞれ、CPU での処理時間、FPGA での処理時間、CPU-FPGA 間のデータ転送時間を指す。T_{CPU}は畳み込み関数内で FPGA に計算させる箇所をコメントアウトしてそれ以外の処理時間を測定することで求める。具体的には、活性化関数の適用の処理がある。T_{COM}は要素数 N の float 型配列を FPGA に転送する時間を測定することで求める。

T_{FPGA}は次のように見積もる。畳み込み処理は出力データのパラメータに関するループ、フィルタに関するループの入れ子構造で処理されており、実際の算術演算は乗算と加算である。ここで、FPGA タブレットのアプリケーションから利用できる回路書き換え可能領域には 120 個の DSP が存在するので、積和演算が 1 クロックで完了し、かつ 120 個の乗算器を用いて並列化を行った場合を仮定する。フィルタとの畳み込みとバイアスの加算を含めた全体の計算時間は式(4.1.2)となる。

$$T_{FPGA} = \frac{T_c}{120} * N_f * W_{out} * H_{out}(C * W_{fil} * H_{fil} + 1) \qquad (4.1.2)$$

但し、T_cはクロック周期、N_fは畳み込むフィルタ数、W_{out}, H_{out}は出力されるデータの幅と高さ、Cは入力チャネル数、W_{fil}, H_{fil}はフィルタの幅と高さである。扱う画像及びフィルタが正方形の場合、それぞれの辺の大きさをL_{out}, L_{fil}と置くと式(4.1.3)に簡略化することができる。

$$T_{FPGA} = \frac{T_c}{120} * N_f * {L_{out}}^2 ({C * L_{fil}}^2 + 1) \qquad (4.1.3)$$

4.2 測定結果

T_{CPU}, T_{COM}について、FPGA タブレット上で測定した結果、表 1 のようになった。また、今回実装する畳み込み層のパラメータより、6x6 フィルタ 12 枚、入力画像は 64x64x3 とし、FPGA の動作クロックが 100MHz であることを考慮するとT_{FPGA}も求めることができる。以上より、式(4.1.1)から畳み込み層の順伝播処理の計算時間の推定値が求まり、その値は 1.615ms であった。また、CPU のみで畳み込み層の順伝播の計算時間は 52.367ms であった。このことから FPGA による部分専用回路化を行うことで約 32 倍のスピードアップが期待される。また、活性化関数の正規化線形関数も回路に組み込むと高々3072 要素の符号検査で済むため、約 72 倍の高速化が期待できる。ただし、実際の回路の設計に依存する部分が大きい為、この見積もり値よりも効率が悪くなることが予想される。

表 1　モデル式の各パラメータの測定結果

パラメータ	時間(ms)
T_{CPU}	0.862
T_{COM}	0.725
T_{FPGA}	0.028
T	1.615

5. おわりに

本研究では近年注目を集めている Deep Learning の手法の一つである畳み込み NN をタブレット端末上で実現し、その処理の高速化に関する提案をした。また、FPGA による専用回路化により通常よりも約 32～72 倍の高速化が期待できることを示した。また、順伝播だけでなく、逆伝播の計算、プーリング層や全結合層の専用回路化も同様に見積もることができ、FPGA の資源を有効活用することができればさらなる高速化が望める。

参考文献

[1] 塩谷 丈史, 成見 哲, “FPGA タブレットによるモバイルアクセラレータ”, 情報処理学会研究報告,Vol2015-HPC-148, No18 (2015)
[2] ZedBoard, http://zedboard.org/product/zedboard
[3] N.Srivastava, G.E.Hiton, A. Krizhevsky, I. Sutskever, and R. Salakhut-dinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machin Learning Research*, 15:1929-1958, 2014