5Q-08

FPGA 実装に向けた深層学習での量子化の検証

高木健司[†1]　渥美柾彦[†2]　孟林[†1]
立命館大学理工学部[†1]　立命館大学大学院理工学研究科[†2]

1. はじめに

近年、深層学習は音声認識や画像認識などの分野で高い性能を示し、注目が集まっている。しかし、深層学習は非常に多くのパラメータを用いて計算を行うので、GPU を用いても実行時間が大きい。FPGA は特定の処理においては GPU よりも高い性能を出す。しかし、深層学習の膨大なパラメータは FPGA のリソースを超え、FPGA の応用範囲が制限される。本研究は、極小量子化によりデータを圧縮し、量子化されたデータに適した回路を FPGA に実装して、FPGA 上での極小量子化された深層学習の有用性を検討する。

2. BNN による FPGA での深層学習

2.1 畳み込みニューラルネットワーク

畳み込みニューラルネットワーク（CNN）は、多層構造を持つニューラルネットワークである深層学習の代表的な手法であり、画像認識のための特徴量抽出器として有名である。CNN の学習、推論には畳み込み演算のため膨大な積和演算が必要であり、ハードウェア資源が豊富でないデバイス上では実行時間が大きくなり、時間的な制約をもたらす。

2.2 Binarized Neural Network と FPGA

Binarized Neural Network(BNN)[1]とは畳み込み演算の重みが(-1,1)の二値で表される CNN である。学習時に対象のパラメータxにハードシグモイド関数(σ (x))を適用することで、二値化し、学習を行う。二値化は数式(1)で行う。

$$x = \begin{cases} +1 & with\ probability\ \ p = \sigma(x), \\ -1 & with\ probability\ \ 1-p, \end{cases} \quad \cdots(1)$$

重みの二値化により、計算速度の高速化、メモリの効率化が期待されるが、極小量子化によりパラメータの表現力が小さくなることで、モデルの精度低下や最適化が困難になるといった欠点もある。

Field-Programmable Gate Array(FPGA)とはハードウェアデバイスであり、FPGA 上に実装されている書き換え可能なロジックにより、開発者が所望するハードウェアの構築が可能である。BNN において、重みが 1 と-1 で表されることから、演算の結果は実数 p、n(-p＝n)を入力とすると表 1 のように示される。表 1 から、BNN における演算は論理式で表現でき、BNN における重みを weight、特徴量を input、lsb を最下位ビットとすると、演算結果 ans は式(2)で示される。

表 1. BNN における畳み込み演算の演算結果

	1	-1
p (p>0)	p	n
n (n<0)	n	p

$$ans = \begin{cases} input & if(weight > 0), \\ \sim(input) + lsb & else, \end{cases} \quad \cdots(2)$$

式(2)に基づき、FPGA 実装に向けた深層学習のための回路を構成することで、FPGA 上での量子化された深層学習実行の有用性を検証する。

3. 実験

3.1 BNN の学習、推論

BNN と 32bit 浮動小数点のオリジナルモデルを学習曲線、推論時の精度、実行速度の観点から比較する。モデルは 4 層の畳み込み、2 層のプーリング、2 層の全結合層で構成される計算モデルを用いた。パラメータとしてバイアスは用いず、畳み込み層と全結合層の出力には、正規化を行う BN 層を適用している。データセットには一般的な画像から構成される Cifar10 と手書き文字のデータセットである MNIST を使用した。学習時の実験環境として、エポック数を Cifar10 に対して 200、MNIST に対して 20 とし、損失関数には Adam、学習終了までに学習率が 1e-3 から 1e-4 に変化するようスケジューラを設定する。

3.2 FPGA 実装に向けた畳み込み演算回路構成

FPGA 実装に向けた畳み込み演算回路構成のため、高級言語から FPGA 実装に向けた回路を作成できる高位合成を用いた。データフォーマットを 32 ビットとし、式(2)で示した論理式と比較のため積和演算を用いた回路を構成する。FPGA 上で深層学習を効率的に処理するため、図 1 で示すシストリックアレイ[2]を実装する。実験では出力チャンネル方向に 32、出力 x 方向に 4 の並列性を持つ回路をターゲットデバイス Ultra96-V2 上に構成する。

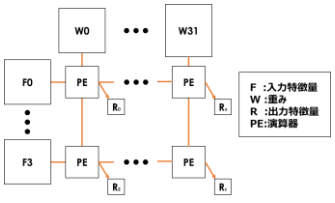

図 1. FPGA 実装に向けた深層学習回路の概要図

Quantization in deep learning for FPGA implementation
†1 Kenji TAKAGI, Lin Meng, College of Science and Engineering, Ritsumeikan University
†2 Masahiko ATSUMI, Ritsumeikan University Graduate School

Copyright ©2022 Information Processing Society of Japan.
All Rights Reserved.

4. 実験結果

4.1 BNN の学習、推論の実験結果

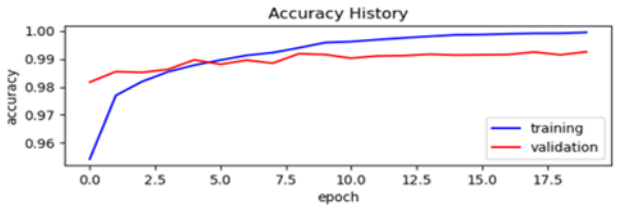

図 2. MNIST オリジナルモデルの学習曲線

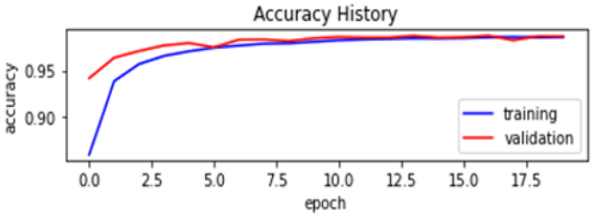

図 3. MNIST の二値化モデルの学習曲線

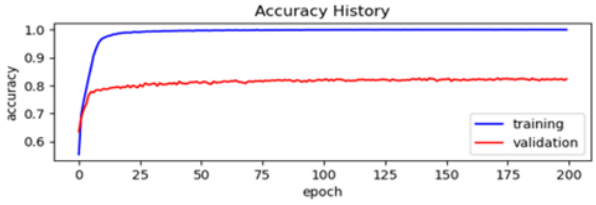

図 4. cifar10 のオリジナルモデルの学習曲線

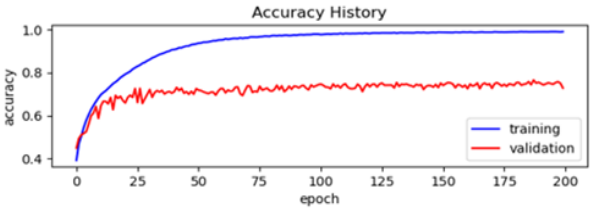

図 5. cifar10 の二値化モデルの学習曲線

表 2 . オリジナルと二値化モデルの認識精度

	32bit	1bit
MNIST(epoch20)	99.26%	98.69%
Cifar10(epoch200)	82.35%	72.84%

表 3. オリジナルと二値化モデルのテストの実行速度

	32bit	1bit
MNIST(10000 枚)	3.801s	3.786s
Cifar10(10000 枚)	3.575s	3.593s

図 2 と図 3 は、学習データ MINST を用いた、オリジナルモデルと提案手法の認識精度を示す. 図 4 と図 5 は、学習データ cifar10 での評価結果である。横軸は学習の epoch 数で、縦軸は、認識精度を示す。上記の結果より、MNIST の場合オリジナルモデルと比較して、重みの二値化における精度の低下はほとんど見られなかった。Cifar10 の場合は約 10%の精度の低下がみられた。重みを 1、-1 にしても内部では 32bit で処理するので、実行速度に変化はなかった。

4.2 FPGA 実装に向けた深層学習のための回路の構成結果

表 4 各回路のハードウェア資源使用率

	DSP	FF	LUT
Logic	0	4%	50%
Multi	71%	20%	39%

表 4 は、Ultra96-V2 実装に向けた深層学習のための回路のハードウェア資源使用率を示す。DSP は積和乗算器を、FF はフリップフロップを、LUT はルックアップテーブルを示す。Logic は論理演算を用いて、Multi は積和演算を用いて回路を構成した場合を示す。表 4 の結果から、Logic の最も大きなハードウェア資源使用率は LUT の 50%であり、Multi の DSP の使用率 71%と比べて小さく、さらなる並列性が期待できる。また、Logic では、乗算を用いないので、DSP の使用率が 0%であった。

回路の処理速度に着目すると、Logic では 14 クロック、Multi では 29 クロックかかる。Multi のアーキテクチャに着目すると、クロック毎に呼び出される各入力特徴量と各重みのデータ数に基づいて、DSP が並列に並び、パイプライン処理が行われる構造である。DSP を用いた乗算には 4 クロックの時間がかかる。しかし、Logic の論理演算では、1 クロックで得られるデータ数を 1 クロックで処理することが可能である。そのためクロックの差異が生じた。

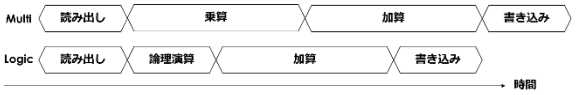

図 6. Logic と Multi のタスクごとのタイミング概要図

以上から、論理演算を用いた場合、実行時間が短縮でき、さらなる並列処理が可能であることから、FPGA 上での極小量子化された CNN の有用性が期待できる。

5. まとめ・展望

本研究では重みの二値化による、認識精度への影響を調査した。重みの二値化では、手書き文字のような簡単なデータセットにおいては比較的大きな認識精度の低下は見られなかった。また、論理演算を用いて、FPGA に向けた BNN の畳み込み演算回路を構成した。合成結果から、積和演算に比べて、さらなる並列性が期待でき、高速な処理が可能であることが見込まれる。展望として、vgg16 などのより多層なモデルを用いて、分類が難しいデータセットも含めた量子化、さらに極小量子化だけでなく 8bit 量子化などの有用性の検証と BNN 全体の FPGA 上での実行を目標とする。

参考文献

[1] M. Courbariaux, et al., : Training Deep Neural Networks with binary weights during propagations, arXiv:1511.00363, 2015.

[2] 天野英晴，他: FPGA の原理と構成，2016 年.

Copyright ©2022 Information Processing Society of Japan.
All Rights Reserved.