博士論文

エッジAIハードウェアに向けたニューラルネットワークの新学習アルゴリズムとその低電力アーキテクチャに関する研究

Novel Neural Network Training Algorithms and their Energy Efficient Architectures for Edge-AI Hardware

2023年1月

博士（工学）

北海道大学大学院情報科学院

金子 竜也

エッジAIハードウェアに向けたニューラルネットワークの新学習アルゴリズムとその低電力アーキテクチャに関する研究

論文要旨

本研究は，人工知能（AI）の学習に用いられる誤差逆伝播法と最適化手法の，特にエッジ端末で学習を行うための軽量化アルゴリズムとそのアーキテクチャに関するものである．

現在，AIは画像認識，翻訳，画像生成等，種々のタスクにおいて，従来の手法に対して優位性を示している．これらの情報処理は，エッジ端末などから収集された大量のデータを基に，処理能力の高いAIエンジンを持つデータセンタ等において集中的に行われている．AIの性能（認識能力等）は，そのモデルの複雑度およびデータ量と密接な関係を持ち，AIの高性能化とともに膨大となる演算処理を効率的に行うためのアルゴリズムやアーキテクチャ研究の必要性が高まっている．一方で，我々の身の回り（エッジ）でAIを活用することを考えた場合，データセンタでの処理に偏重したAIシステムでは，セキュリティ（機密・個人情報等を含むデータがクラウドを流れてしまう），リアルタイム性（エッジでの電力制約によりクラウドへの常時接続は困難であるため，間欠動作による遅延が発生する），通信帯域（AIの学習には大量のデータが必要であり通信帯域を圧迫する），といった問題が予測される. これらの問題に対応するために，クラウド上のAIに大きく依存せずに，ユーザの身近なところで（可能な限りオフライン環境下で）AI処理を行う「エッジAI」の実現が期待されている．

AIの演算処理は「推論処理」と「学習処理」に大別でき，AIの恩恵を得るためにはどちらの処理も必須である．そのうち，推論処理（主に積和演算，畳み込み演算と非線形関数による活性化処理）については多くのエッジAIソリューションが存在する．一方，エッジでの学習処理（誤差逆伝播法およびパラメータの最適化手法）については，未だ汎用プロセッサを中心としたソフトウェア的な手法に留まっており，学習処理のコスト（学習にかかる時間，消費電力，ハードウェア資源）は極めて高い．そこで本研究では，電力やハードウェア資源が制限されるエッジにおいて，学習処理を可能とする新規アルゴリズムとそれらを実装するアーキテクチャの構築を目的とした．

最初に，ニューラルネットワークの最も基本的な学習方法である誤差逆伝播法と確率的勾配降下法のハードウェア指向アルゴリズムを構築した．推論処理のみを行う従

来のエッジ AI 研究では，推論処理の演算における数値表現方式を浮動小数点方式から固定小数点方式へと変更，または固定小数点方式のビット数を削減することで，演算の軽量化を実現してきた．本研究では，学習処理における演算のビット精度を制限する（固定小数点方式とする）ことで演算を軽量化した．学習処理を行う演算アーキテクチャを構築して評価した結果，性能を維持するために必要となる最低ビット数，および提案アーキテクチャの並列度を可変にすることで広範な用途に対応できることを明らかにした．

次に，誤差逆伝播法を軽量化する手法を提案し，そのアーキテクチャ構築を行った．上述の固定小数点方式の導入により演算を軽量化した結果，推論処理と学習処理では要求される最低ビット数が異なり，学習時に AI 性能に影響を与えるまで変化するパラメータ数は極小数であることを明らかにした．このことは，パラメータを保存するメモリ容量や，パラメータの更新（メモリへの書き込み）に必要な電力等の大部分は無為に消費されていることを意味する．この発見に基づき，AI 性能を維持したまま，誤差逆伝播法のビット数（メモリ容量），およびメモリへの書き込み回数を削減可能な新アルゴリズムを構築した．さらに，演算に必要なハードウェア資源量が少ないエッジ AI 向け学習アーキテクチャを構築し FPGA に実装して評価した．既存手法と比較して，使用メモリ量を 49.8%削減可能であること，およびメモリアクセスに係る消費電力を 0.0017 倍に削減可能であることを示した．

上述の研究・成果はディジタル回路を対象としたものであるが，次の挑戦として，アナログ回路を導入した「コンピューティングインメモリ（Computing in Memory: CIM）デバイス」のための誤差逆伝播法，および CIM デバイスを用いて学習処理を行うアーキテクチャの構築を行った．AI は多量の積和演算を必要とするため，従来のノイマン型の計算機で AI 演算を行うと，プロセッサとメモリ間のデータ転送に係るボトルネックにより演算に膨大な時間がかかってしまう．この問題の解消に向けた取り組みとして多量のデータ（パラメータ）を保存しているメモリ上で積和演算を行う CIM アーキテクチャが注目されている．本研究では，非ノイマン型アーキテクチャの一種である「ReRAM を用いた CIM AI デバイス」の学習機能の実現に向けた新規アルゴリズムの開発を行った．通常の誤差逆伝播法は，活性化前のアナログ値の読み出しを必要とするため，多くの CIM AI デバイスには適さない．そこで，ディジタル誤差逆伝播法（Digital BP）に着目し，その弱点（性能低下）を補うニューラルネットワークの構造を新たに考案した．この新構造によって，従来の Digital BP では不可能だった線形回帰や多クラスの識別が可能になることを示した．さらに，Digital BP の演算を行うアーキテクチャ構

築とその FPGA 実装を行い，演算コアの消費電力を 10 mW 以下にできることを示した．

最後に，軽量性と高性能を両立する最適化手法の開発を行った．従来のエッジ AI 向け学習アルゴリズム・アーキテクチャ研究では，最適化手法として主に確率的勾配降下法が用いられている．確率的勾配降下法は，極めて単純な最適化手法であるが故に学習の収束性や安定性が悪い．より高度な最適化手法は，大容量のメモリと高度な演算処理を必要とするため，エッジ AI 領域ではこれまで確率的勾配降下法を採用せざるを得ない状況にあった．そこで本研究では，メモリ容量と演算量を削減する新規最適化手法のアルゴリズムを構築した．ディジタル処理では一般的に精度を落とす要因となる「量子化」を積極的に利用する（固定小数点による量子化と，乗除算をビットシフトで代替できる対数量子化を組合せる）ことで，最上ランクの最適化手法（RMSProp 等）と同程度の性能を持ちつつ，省メモリ・省リソースの最適化ハードウェアが構築可能であることを示した．高性能な最適化ハードウェアをエッジ AI に組込むことができれば，高速な学習の収束性，すなわち学習回数を削減できる．一回の学習に係るリソースを減らして低電力化するという従前のアプローチとは異なり，学習回数（ReRAM の書き込み回数）を減らすことで低電力化する．提案手法は，従来手法と比べて約 70% の省メモリ化と 4 倍の高速化を達成可能であることを示した．

目 次

第1章　序論

1.1　研究の主旨

本研究は，人工知能技術の中でもニューラルネットワークの学習に用いられる最適化手法の，特にエッジ端末で実行するためのアーキテクチャとアルゴリズムの両面に関するものである．

今日，人工知能 (AI: Artificial Intelligence) は特に機械学習のニューラルネットワークの分野において深層学習を代表に画像認識 [1, 2, 3, 4]，自然言語処理 [5, 6]，画像生成 [7, 8] や強化学習 [9, 10] など種々のタスクにおいて，従来手法に対して優位性を示している．ここで AI の性能 (認識率や翻訳精度，データ生成能力) は AI モデルの複雑度と密接な関係を持ち，高性能化と共に要求される処理能力は膨大なものとなっている．そのため近年では，膨大となる演算処理を効率的に行うための研究が活発的に行われており，アルゴリズム研究といったソフトウェアの領域 [11] から，アーキテクチャ研究といったハードウェアの領域 [12] まで幅広く取り組まれている．これらの情報処理はエッジ端末で収集されたデータを基に，処理能力の高い AI エンジンを持つデータセンタを用いて集中的に行う仕組みとなっている．このようなクラウドに集約した情報処理形態では，セキュリティ（機密・個人情報等を含むデータがクラウドを流れてしまう），リアルタイム性（エッジでの電力制約によりクラウドへの常時接続は困難であるため，間欠動作による遅延が発生する），通信帯域（AI の学習には大量のデータが必要であり通信帯域を圧迫する）といった問題が予想される．また，環境との相互作用を通じて継続的な学習を行う場合のみにおいて，知的エージェントは真の自律性を獲得する [13] という点からも，クラウドに依存しないオフライン環境下で AI 処理を行うエッジ AI の実現が期待されている．

現在の AI の主流であるニューラルネットワーク (NN: Neural Network) の演算処理は「推論処理」と「学習処理」に大別でき，AI の恩恵を得るためにはどちらの処理も必須である．そのうち，推論処理（主に積和演算，畳み込み演算と非線形関数による活性化処理）については多くのエッジ AI ソリューションが存在する．一方で，エッジで

の学習処理には誤差逆伝播法 (BP: BackPropagation)[14] と最適化手法 [15, 16, 17] という 2 つの演算の実装が不可欠である．ただし，これらの演算については未だ汎用プロセッサを中心としたソフトウェア的手法に留まっており，学習にかかる時間，消費電力，ハードウェア資源等の演算負荷が極めて高い．そこで本研究では，電力やハードウェア資源が制限されるエッジ AI において，演算負荷の高い学習処理を実現するための新規学習アルゴリズムとそれらを実行するアーキテクチャの構築を目的とした．以下に，その取り組みの概要を示す．

第一に，NN の最も基本的な学習方法である BP と確立的勾配降下法 (SGD: Stochastic Gradient Descent)[15] のハードウェア指向アルゴリズムを開発した．本研究では，学習処理における演算のビット精度を固定小数点方式へと変更・削減し演算を軽量化した．．また学習処理を行う仮想アーキテクチャを考案し評価した結果，性能を維持するために必要となる最低限のビット精度と，並列性を可変にすることで広範な用途に対応できることを明らかにした．

第二に，BP を軽量化する手法を提案し，そのアーキテクチャ構築を行い書き換え可能なハードウェアである FPGA (Field Programmable Gate Array) へと実装した．上述の固定小数点方式の導入により演算を軽量化した結果，推論処理と学習処理では要求される最低ビット数が異なり，学習時に AI 性能に影響を与えるまで変化するパラメータ数は極小数であることを明らかにした．このことは，パラメータを保存するメモリ容量や，パラメータの更新（メモリへの書き込み）に必要な電力等の大部分は無為に消費されていることを意味する．この発見に基づき，本研究では AI 性能を維持したまま，誤差逆伝播法のビット精度（メモリ容量），およびメモリへの書き込み回数を削減可能な新アルゴリズムを構築した．さらに，演算に必要なハードウェア資源量が少ないエッジ AI 向け学習アーキテクチャを構築し FPGA に実装して評価した．既存手法と比較して，使用メモリ量を 49.8%削減可能であることを示したほか，見積もり上であるがメモリアクセスに係る消費電力を 0.0017 倍に削減可能であることを示した

上述の研究・成果はディジタル回路を対象としたものであるが，次の挑戦として，アナログ回路を導入した「コンピューティングインメモリ（Computing in Memory: CIM）デバイス」のための誤差逆伝播法，および CIM デバイスを用いて学習処理を行うアーキテクチャの構築を行った．AI は多量の積和演算を必要とするため，従来のノイマン型の計算機で AI 演算を行うと，プロセッサとメモリ間のデータ転送に係るボトルネックにより演算に膨大な時間がかかってしまう．この問題の解消に向けた取り組みとして多量のデータ（パラメータ）を保存しているメモリ上で積和演算を行う CIM アーキ

テクチャが注目されている．本研究では，非ノイマン型アーキテクチャの一種である「ReRAM を用いた CIM AI デバイス」の学習機能の実現に向けた新規アルゴリズムの開発を行った．通常の誤差逆伝播法は，活性化前のアナログ値の読み出しを必要とするため，多くの CIM AI デバイスには適さない．そこで，Digital BP 法 [18] に着目し，その弱点（性能低下）を補うニューラルネットワークの構造を新たに考案した．この新構造によって，従来の Digital BP では不可能だった線形回帰や多クラスの識別が可能になることを示した．さらに，Digital BP の演算を行うアーキテクチャ構築とその FPGA 実装を行い，演算コアの消費電力を 10 mW 以下にできることを示した．

最後に，軽量性と高性能を両立する最適化手法の開発を行った．従来のエッジ AI 向け学習アルゴリズム・アーキテクチャ研究では，最適化手法として主に確率的勾配降下法が用いられている．確率的勾配降下法は，極めて単純な最適化手法であるが故に学習の収束性や安定性が悪い．より高度な最適化手法は，大容量のメモリと高度な演算処理を必要とするため，エッジ AI 領域ではこれまで確率的勾配降下法を採用せざるを得ない状況にあった．そこで本研究では，メモリ容量と演算量を削減する新規最適化手法のアルゴリズムを構築した．ディジタル処理では一般的に精度を落とす要因となる「量子化」を積極的に利用する（固定小数点による量子化と，乗除算をビットシフトで代替できる対数量子化を組合せる）ことで，より高度な適化手法（RMSProp 等）と同程度の性能を持ちつつ，省メモリ・省リソースの最適化ハードウェアが構築可能であることを示した．高性能な最適化ハードウェアをエッジ AI に組込むことができれば，高速な学習の収束性，すなわち学習回数を削減できる．一回の学習に係るリソースを減らして低電力化するという従前のアプローチとは異なり，学習回数（ReRAM の書き込み回数）を減らすことで低電力化する．提案手法は，従来手法と比べて約 70% の省メモリ化と 4 倍の高速化を達成可能であることを示した．

単一デバイス上で学習も含めた処理を完結させるこれらの研究は，あらゆる環境への適応を可能にするなどエッジ AI の可用性を拡げるものであり，高度な情報処理社会実現への寄与が期待される．

1.2　研究の背景

AI 技術は過去に 2 度の冬の時代を迎えつつも，現在最大の隆盛期にある．2015 年に画像認識の分野において，深層学習 (DL: Deep Learning) を用いた手法が人間による認識精度を越えたこと [19, 20] を発端にして，様々な応用事例が登場している．現在に

至っては文章を入力として人間と変わらない程自然なイラストレーションの作成 [21] や自然な文章の生成 [22, 23] を行う AI モデルが登場するなど．AI 技術はその可能性の高さ存分に発揮している．この潮流は以下の三つの点に端を発していると考えられる．

1. ビッグデータを背景とした大規模な学習用データセットの登場により，従来の “AI を人間がモデル化する行為” そのものを自動化できたこと．

2. DL アルゴリズムの発達により，100 層を越える非常に深い層を持つ AI モデルの学習が可能になったこと．

3. ムーアの法則 [24] に従って CMOS プロセス微細化が進みコンピューティングデバイスの性能が向上したことにより，負荷の高い大規模な AI モデルであっても現実的な実行時間での処理が可能になったこと．

つまり，インターネットの発展や情報端末の普及等の情報処理基盤，NN アルゴリズム改良等のソフトウェアの発達と，GPU (Graphic Processing Unit) や TPU (Tensor Processing Unit)[25] 等のハードウェアの発達，これらの要素が互いに互いを刺激し合うことにより AI は革命的ともいえるほどの発展を遂げてきた．

このような背景から現在の AI 技術を利用する情報処理システムは，ネットワーク上のエッジデバイスでは演算負荷の高い AI 処理は実行せず，データ収集とクラウドへの転送が主な役目となる．そして収集されたデータを基にして大規模かつ高性能な基盤を持つクラウド上で AI 処理を行う．つまり，AI モデルの実体はクラウド上にのみ存在しており，クラウドに集約したシステムである．こうしたサーバ群で構成されるクラウドから，パソコンやスマートフォン更にはウェアラブルデバイスや IoT デバイスに至るまで，演算機器や記憶装置の急増・発展により我々は情報中心の時代に生きているといえる．この時代ではユビキタスなコンピューティングやサービスがクラウドからエッジまで溢れている．Cisco の白書 [26] によると 2020 年までに約 500 億もの IoT デバイスがインターネットに接続されており，そこから生成されるデータは 850 ゼタバイト (ZB) とされている．一方で世界的なクラウドのトラフィックは 20.6ZB しかなく [27]，この点からクラウド集約型からエッジ分散型コンピューティングへの転換が必要とされている．このような IoT 時代の到来による通信帯域の問題の他にも，いわゆる企業秘密や個人的なデータを秘匿したいというセキュリティに関する懸念や，自動運転等に代表されるようなリアルタイム性に関する懸念からエッジで AI 処理を行う機運が高まっている．特にセキュリティ面や環境に適応するという観点からは，その場で学習を行うエッジ AI が求められている．電力やハードウェア資源が制限されるエッ

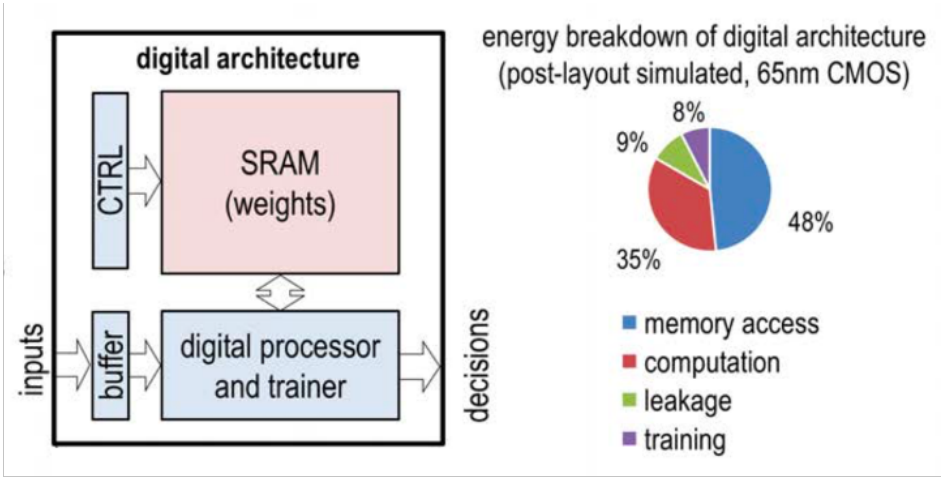

図 1.1 ノイマン型アーキテクチャにおける消費電力内訳 [28]

ジデバイス上で演算負荷の高い AI 処理を行うためには，エッジデバイスというハードウェアを見据えたアルゴリズム・アーキテクチャの研究が不可欠となる．

現代の主流 AI モデルとなる NN は生物の神経細胞を模倣した数理モデルを始まりとする．NN の基本構造は神経細胞を複数個並べたものを層とし，これを多層にしたものである．入力データと NN のパラメータである重みとの積和を取った後，活性化関数を用いて非線形変換を施す．そのために，DL は膨大な量の積和演算によって構成されている．大量のデータやパラメータを用いた演算を行っていることや，活性化関数の性質により NN は高い数値精度を要求しないという特徴を持つ．例えば，多くの機械学習ライブラリでは NN の演算に係る数値を浮動小数点方式の 32 bit で表現しているが，これを固定小数点方式へと変更し更にビット精度を削減した場合でも同等の性能が維持できると知られている [29, 30, 31, 32, 33, 34, 35, 36, 37, 38]．この手法は量子化と呼ばれ，その対象となるのは NN パラメータ，活性化関数の出力である活性値，そして学習時に用いられる勾配の三つである．ここで，パラメータの量子化はメモリ容量に直結することから特にエッジ AI では重要となる．究極的には，[29] で見られるように性能を維持したままパラメータを 0 と 1 の二値化にまでビット精度を削減できることが知られている．ただし，学習時には活性値，勾配共に高いビット精度を持つ必要があることからエッジ上での学習に向けてこの活性値と勾配の量子化が課題となっている．

現代のコンピューティングデバイスの主流はプロセッサとメモリが分かれているノイマン型のアーキテクチャである．このようなアーキテクチャではプロセッサとメモリ間の通信が，速度や電力の面からボトルネックとなることが知られている．図 1.1 に示すようにノイマン型アーキテクチャの AI デバイスにおいて，消費電力の内メモリ

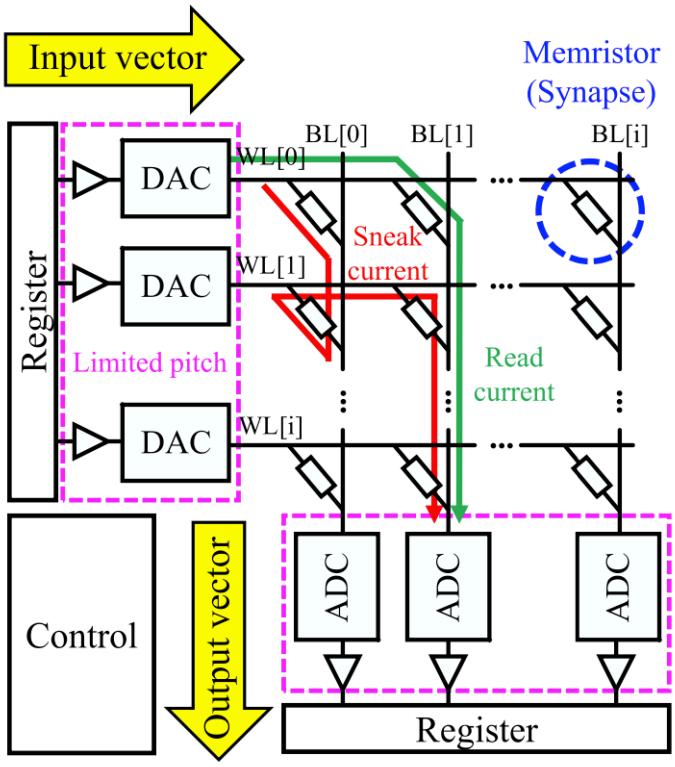

図 1.2 CIM デバイスの典型的なセルアレイ構造 [39]

アクセスが約 50%を占めており，電力に制限を持つエッジ AI では特に重要な課題となる．この解決に向けて，メモリ上で積和演算を行うコンピューティングインメモリ (CIM: Computing-in-Memory) が注目されている．特に，低電力でデータの保存が可能な ReRAM (Resistive RAM) や PCM (Phase Change Memory) 等の不揮発性メモリ (NVM: Non-Volatile Memory) と組み合わせた研究が盛んに行われている [39, 40, 41, 42]．図 1.2 に示すように一つの NVM を NN のパラメータである結合重みとし，クロスバーアレイ構造を用いることでキルヒホッフの法則によりアナログ的に積和演算を実行する．今後のエッジ AI 研究においてボトルネックを解消する CIM の研究が重要性を増している．

NN の学習は出力データと教師データから算出した誤差を基にして，最適なパラメータとなるように調整することで進んでいく．ここでその内容は，得られた誤差をモデル全体へと伝播する BP とパラメータの更新値を決定する最適化手法の二つに分けることができる．近年では学習も対象にしたエッジ AI 研究が行われている [43, 44, 45, 46, 47] が，その多くは BP に関するものであり最適化手法については最も基本的な手法である SGD やそれを基にした手法が用いられている．一方でアルゴリズム分野では [15,

16, 17] に見られるように，AI 分野そのものでは最適化手法の研究は盛んである．今日の AI の発展に最適化手法研究は大きく寄与しており，このような高度な最適化手法は最も単純な SGD と比べて，汎化性能や収束までにかかる学習回数が優れている．特に収束性の良さは学習回数の削減が可能であることから演算回数そのものを減らすことによる消費電力の削減という抜本的なアプローチになり得る．しかし，高度な最適化手法は高度な演算処理と大容量のメモリを必要とすることが課題となる．

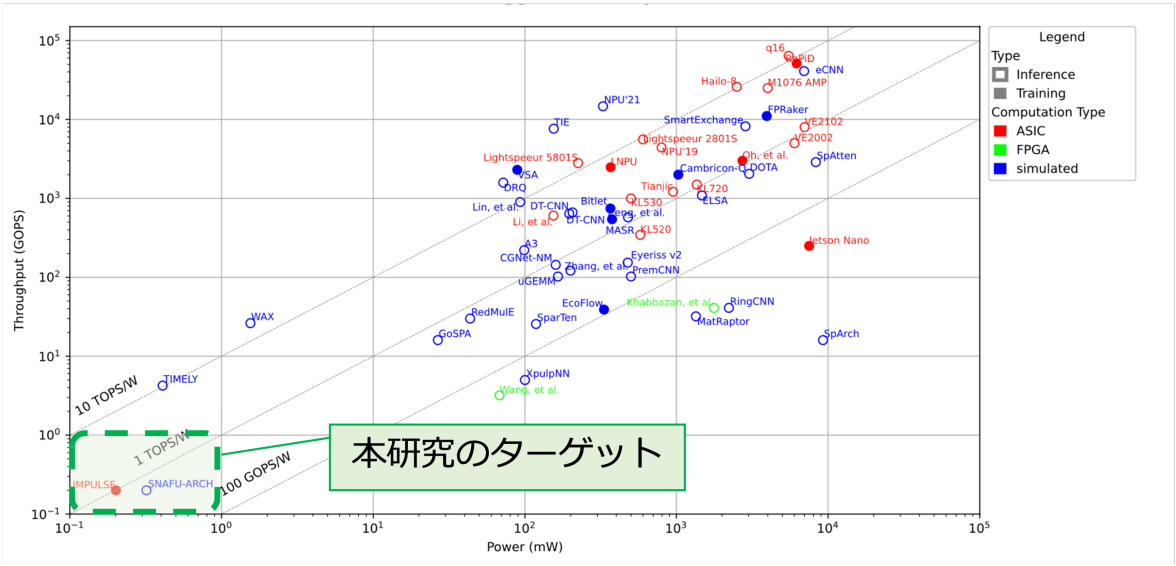

図 1.3 エッジ AI ハードウェアの性能と消費電力と本研究のターゲット ([48] を基に作成．)

1.3 研究の目的

上述の背景より，本研究では，電力やハードウェア資源が制限されるエッジ AI において演算負荷の高い学習処理を実現するための新規学習アルゴリズムとそのアーキテクチャの構築を目的とした．図 1.3 にエッジ AI やクラウド AI ハードウェアの電力・性能 (スループット) に関する位置付けを示す．背景で述べたように，AI ハードウェアはクラウド向け用途とエッジ向け用途とで二つに大別可能である．当初 AI ハードウェアは膨大な AI 処理の高速化・高効率化を目的としてクラウド用途での発展が進んでいた．次第に我々の実世界での応用を見据えて，低電力で推論処理を行うエッジ用途のハードウェアが出現した．これらのハードウェアはデータを高速かつ高効率に処理することに重きが置かれており，並列化をはじめとしたアーキテクチャ上の工夫が為されている．一方で，本研究の位置付けはより低電力なサブミリワットオーダとなる図左下を

目指すものとなる．本研究ではエッジ AI デバイスのみで学習から推論までの AI 処理の完結を目標としている．つまり，完全なオフライン環境下での運用を前提とするものであり，そのため学習データは外界とのセンシング等によって得ることが期待される．このような場合において，スループットは外界とのインタラクトに律速されることが予想されるために処理の高速化ではなく消費電力の削減を目指した．従って，本研究で提案するアーキテクチャは行列演算を時間方向に展開することで一つの演算器を使い回す等の最小構成で AI 処理を行うことを基本的な設計思想に据えている．あるいは，高速な処理が求められるか否かはアプリケーションに依存するところが大きく，高速性が求められる場合においても演算器の並列化について余地が残っており，消費電力とのトレードオフの選択が可能となっている．以上を踏まえて，学習処理を含めたエッジ AI デバイス開発に向けて新規学習アルゴリズムの開発に取り組んだ

1.4 本論文の構成

以上で述べた研究背景と研究目的に基づき，本論文は以下の章で構成される．

第 2 章

本章では，ニューラルネットワークの最も基本的な学習方法である誤差逆伝播法と確立的勾配降下法のハードウェア指向アルゴリズムを開発した．従来の学習を含まないエッジ AI 研究では数値表現方式を浮動小数点方式から固定小数点方式へと変更することで，演算の軽量化を実現してきた．同様に本研究では，学習時の数値表現に係るビット精度を固定小数点方式へと変更しビット精度を制限することで演算を軽量化し，また仮想のアーキテクチャを考案した．性能を維持するために必要となる最低限のビット精度を探査し，提案アーキテクチャの並列性を可変にすることで広範な用途に対応できることを示した．

第 3 章

本章では，誤差逆伝播法を軽量化する手法を提案し，アーキテクチャの構築を行い FPGA へと実装した．上述の固定小数点方式の導入により演算を軽量化した結果，推論処理と学習処理では要求される最低ビット数が異なる．1 回の学習で全てのパラメータが更新されるものの，モデルの出力に影響を与えるまでに変化するのはパラメータ数は極小数であることを明らかにした．つまり，パラメータに関わるメモリの容量や書

き込み電力等の大部分は無為に消費されていることを意味する．この発見に基づき本研究では，性能を維持したまま誤差逆伝播法のビット精度・書き込み回数を削減する新規 BP アルゴリズムの提案を行った．また，演算リソース量が最低となるエッジ AI 向けアーキテクチャを考案し FPGA へと実装した．既存手法と比較して，使用メモリ量を 49.8%削減可能であること，見積もり上であるがメモリアクセスに係る消費電力を 0.0017 倍にまで削減可能であることを示した．

第 4 章

本章では，アナログコンピューティングインメモリデバイスに向けた誤差逆伝播法の開発と，アーキテクチャの構築を行い FPGA へと実装した．ニューラルネットワークは多量の積和演算から成り立っているために，従来のノイマン型のディジタルアーキテクチャではプロセッサとメモリ間のデータ転送がボトルネックとなる．この問題点の解消に向けた取り組みとしてデータ（パラメータ）を保存しているメモリ上で積和演算を行うコンピューティングインメモリデバイスが注目されている．本研究では，CIM デバイスの一つとなる，不揮発性メモリである ReRAM をクロスバー状に配置することでアナログ回路的に推論処理を行う AI チップに向けた新規学習アルゴリズムの開発を行った．通常の誤差逆伝播法は，活性化前のアナログ値の読み出しを必要とするため，多くの CIM AI デバイスには適さない．そこで，ディジタル誤差逆伝播法（Digital BP）に着目し，その弱点（性能低下）を補うニューラルネットワークの構造を新たに考案した．この新構造によって，従来の Digital BP では不可能だった線形回帰や多クラスの識別が可能になることを示した．さらに，Digital BP の演算を行うアーキテクチャ構築とその FPGA 実装を行い，演算コアの消費電力を 10 mW 以下にできることを示した．

第 5 章

本章では，軽量性と高性能を両立する最適化手法の開発を行った．従来のエッジ AI 向け学習アルゴリズム・アーキテクチャ研究では，最適化手法として主に確立的勾配降下法を基にしたアルゴリズムを用いている．確率的勾配降下法は極めて初歩的な最適化手法であるが故に学習の収束性や安定性等で劣るものがある．一方で，より高度な最適化手法は大容量のメモリと高度な演算処理を必要とするため，エッジ AI 領域では確立的勾配降下法を使用せざるを得ない状況にあった．本研究では，量子化を多重に施すことでメモリ容量と演算処理の削減をする新規最適化手法アルゴリズムのを構築した．ディジタル処理では一般的に精度を落とす要因となる「量子化」を積極的に利

用する（固定小数点による量子化と，乗除算をビットシフトで代替できる対数量子化を組合せる）ことで，より高度な最適化手法（RMSProp 等）と同程度の性能を持ちつつ，省メモリ・省リソースの最適化ハードウェアが構築可能であることを示した．高性能な最適化ハードウェアをエッジ AI に組込むことができれば，高速な学習の収束性，すなわち学習回数を削減できる．一回の学習に係るリソースを減らして低電力化するという従前のアプローチとは異なり，学習回数（ReRAM の書き込み回数）を減らすことで低電力化する．提案手法は，従来手法と比べて約 70%の省メモリ化と 4 倍の高速化を達成可能であることを示した．

第 6 章

以上の研究をまとめ本論文の総括を行う．

参考文献

[1] M. Tan and Q. Le. “EfficientNetV2: Smaller Models and Faster Training”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. **139**. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 10096–10106.

[2] Z. Liu, Y. Lin, Y. Cao, et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 10012–10022.

[3] H. Liu, Z. Dai, D. So, et al. “Pay Attention to MLPs”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, et al. **34**. Curran Associates, Inc., 2021, pp. 9204–9215.

[4] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, et al. “Mlp-Mixer: An All-Mlp Architecture for Vision”. In: *Advances in Neural Information Processing Systems* **34** (2021), pp. 24261–24272.

[5] A. Vaswani, N. Shazeer, N. Parmar, et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. **30**. Curran Associates, Inc., 2017.

[6] J. Kaplan, S. McCandlish, T. Henighan, et al. “Scaling Laws for Neural Language Models”. In: *arXiv preprint arXiv:2001.08361* (2020). arXiv: 2001.08361.

[7] S. Kong and D. Ramanan. “OpenGAN: Open-set Recognition via Open Data Generation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 813–822.

[8] P. Dhariwal and A. Nichol. “Diffusion Models Beat Gans on Image Synthesis”. In: *Advances in Neural Information Processing Systems* **34** (2021), pp. 8780–8794.

[9] D. Silver, J. Schrittwieser, K. Simonyan, et al. “Mastering the Game of Go without Human Knowledge”. In: *nature* **550**.7676 (2017), pp. 354–359.

[10] D. Hafner, T. P. Lillicrap, M. Norouzi, et al. “Mastering Atari with Discrete World Models”. In: *International Conference on Learning Representations*. 2021.

[11] T. Liang, J. Glossner, L. Wang, et al. “Pruning and Quantization for Deep Neural Network Acceleration: A Survey”. In: *Neurocomputing* **461** (2021), pp. 370–403. DOI: 10.1016/j.neucom.2021.07.045.

[12] Z. Zou, Y. Jin, P. Nevalainen, et al. “Edge and Fog Computing Enabled AI for IoT-An Overview”. In: *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 2019, pp. 51–56. DOI: 10.1109/AICAS.2019.8771621.

[13] A. Amravati, S. B. Nasir, S. Thangadurai, et al. “A 55nm Time-Domain Mixed-Signal Neuromorphic Accelerator with Stochastic Synapses and Embedded Reinforcement Learning for Autonomous Micro-Robots”. In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, pp. 124–126. DOI: 10.1109/ISSCC.2018.8310215.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning Representations by Back-Propagating Errors”. In: *nature* **323**.6088 (1986), pp. 533–536.

[15] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[16] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). arXiv: 1412.6980.

[17] P. Foret, A. Kleiner, H. Mobahi, et al. “Sharpness-Aware Minimization for Efficiently Improving Generalization”. In: *International Conference on Learning Representations*. 2021.

[18] T. Oohori, H. Naganuma, and K. Watanabe. “A New Backpropagation Learning Algorithm for Layered Neural Networks with Nondifferentiable Units”. In: *Neural Computation* **19**.5 (2007), pp. 1422–1435. DOI: 10.1162/neco.2007.19.5.1422.

[19] O. Russakovsky, J. Deng, H. Su, et al. “Imagenet Large Scale Visual Recognition Challenge”. In: *International journal of computer vision* **115**.3 (2015), pp. 211–252.

[20] K. He, X. Zhang, S. Ren, et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[21] R. Rombach, A. Blattmann, D. Lorenz, et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10684–10695.

[22] T. Brown, B. Mann, N. Ryder, et al. “Language Models Are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, et al. **33**. Curran Associates, Inc., 2020, pp. 1877–1901.

[23] *ChatGPT: Optimizing Language Models for Dialogue*. "https://openai.com/blog/chatgpt/".

[24] R. Schaller. “Moore’s Law: Past, Present and Future”. In: *IEEE Spectrum* **34**.6 (1997), pp. 52–59. DOI: 10.1109/6.591665.

[25] N. P. Jouppi, C. Young, N. Patil, et al. “In-Datacenter Performance Analysis of a Tensor Processing Unit”. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. 2017, pp. 1–12.

[26] *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. "https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf".

[27] *Cisco Global Cloud Index: Forecast and Methodology, 2016-2021*. "https://virtualization.network/Resources/Whitepapers/0b75cf2e-0c53-4891-918e-b542a5d364c5_white-paper-c11-738085.pdf".

[28] S. K. Gonugondla, M. Kang, and N. Shanbhag. “A 42pJ/Decision 3.12TOPS/W Robust in-Memory Machine Learning Classifier with on-Chip Training”. In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, pp. 490–492. DOI: 10.1109/ISSCC.2018.8310398.

[29] M. Courbariaux, Y. Bengio, and J.-P. David. “BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations”. In: ().

[30] M. Courbariaux, I. Hubara, D. Soudry, et al. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. 2016. DOI: 10.48550/ARXIV.1602.02830.

[31] M. Rastegari, V. Ordonez, J. Redmon, et al. “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks”. In: *Computer Vision - ECCV 2016*. Ed. by B. Leibe, J. Matas, N. Sebe, et al. Cham: Springer International Publishing, 2016, pp. 525–542.

[32] F. Li, B. Liu, X. Wang, et al. *Ternary Weight Networks*. 2016. DOI: 10.48550/ARXIV.1605.04711.

[33] C. Zhu, S. Han, H. Mao, et al. *Trained Ternary Quantization*. 2016. DOI: 10.48550/ARXIV.1612.01064.

[34] P. Yin, S. Zhang, Y. Qi, et al. *Quantization and Training of Low Bit-Width Convolutional Neural Networks for Object Detection*. 2016. DOI: 10.48550/ARXIV.1612.06052.

[35] S. Zhou, Y. Wu, Z. Ni, et al. *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*. 2016. DOI: 10.48550/ARXIV.1606.06160.

[36] D. Miyashita, E. H. Lee, and B. Murmann. *Convolutional Neural Networks Using Logarithmic Data Representation*. 2016. DOI: 10.48550/ARXIV.1603.01025.

[37] I. Hubara, M. Courbariaux, D. Soudry, et al. *Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations*. 2016. DOI: 10.48550/ARXIV.1609.07061.

[38] A. Zhou, A. Yao, Y. Guo, et al. *Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights*. 2017. DOI: 10.48550/ARXIV.1702.03044.

[39] J.-M. Hung, X. Li, J. Wu, et al. "Challenges and Trends inDeveloping Nonvolatile Memory-Enabled Computing Chips for Intelligent Edge Devices". In: *IEEE Transactions on Electron Devices* **67**.4 (2020), pp. 1444–1453. DOI: 10.1109/TED.2020.2976115.

[40] G. Burr, R. Shelby, C. di Nolfo, et al. "Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165,000 Synapses), Using Phase-Change Memory as the Synaptic Weight Element". In: *2014 IEEE International Electron Devices Meeting*. 2014, pp. 29.5.1–29.5.4. DOI: 10.1109/IEDM.2014.7047135.

[41] C. Frenkel and G. Indiveri. "ReckOn: A 28nm Sub-Mm2 Task-Agnostic Spiking Recurrent Neural Network Processor Enabling on-Chip Learning over Second-Long Timescales". In: *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. **65**. 2022, pp. 1–3. DOI: 10.1109/ISSCC42614.2022.9731734.

[42] 河野 和幸. "不揮発性メモリを用いた AI チップの実装技術". In: 電子情報通信学会誌, *The journal of the Institute of Electronics, Information and Communication Engineers* **103**.5 (May 2020), pp. 543–548.

[43] S. Yu, Z. Li, P.-Y. Chen, et al. “Binary Neural Network with 16 Mb RRAM Macro Chip for Classification and Online Training”. In: *2016 IEEE International Electron Devices Meeting (IEDM)*. 2016, pp. 16.2.1–16.2.4. DOI: 10.1109/IEDM.2016.7838429.

[44] D. Soudry, D. Di Castro, A. Gal, et al. “Memristor-Based Multilayer Neural Networks with Online Gradient Descent Training”. In: *IEEE Transactions on Neural Networks and Learning Systems* **26**.10 (2015), pp. 2408–2421. DOI: 10.1109/TNNLS.2014.2383395.

[45] M. Giordano, K. Prabhu, K. Koul, et al. “CHIMERA: A 0.92 TOPS, 2.2 TOPS/W Edge AI Accelerator with 2 MByte on-Chip Foundry Resistive RAM for Efficient Training and Inference”. In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492347.

[46] B. Crafton, M. West, P. Basnet, et al. “Local Learning in RRAM Neural Networks with Sparse Direct Feedback Alignment”. In: *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE. 2019, pp. 1–6.

[47] T. Gokmen and W. Haensch. “Algorithm for Training Neural Networks on Resistive Device Arrays”. In: *Frontiers in Neuroscience* **14** (Feb. 2020), p. 103. DOI: 10.3389/fnins.2020.00103.

[48] C. Åleskog, H. Grahn, and A. Borg. “Recent Developments in Low-Power AI Accelerators: A Survey”. In: *Algorithms* **15**.11 (2022). DOI: 10.3390/a15110419.

第2章　ハードウェア指向学習アルゴリズムとそのアーキテクチャ

2.1　導入

本研究は，ニューラルネットワーク (NN) の最も基本的な学習方法である誤差逆伝播法 (BP) と確立的勾配降下法 (SGD)[15] のハードウェア指向アルゴリズム開発を目的とするものである．現在の AI において主流である NN においてその処理形態は二つに大別される．一つは与えられた入力を基に予測結果を出力する推論処理である．もう一つは，正しい予測を可能とするために AI モデルを訓練する学習処理である．AI 専用ハードウェア研究は推論処理の分野においてよく研究されており [12]，その多くは処理を効率良く実行するために，主として数値表現方式を浮動小数点方式から固定小数点方式へと変更することで演算の軽量化を行うものである．そのため学習を対象とした本研究においても，まずは数値表現に係るビット精度を固定小数点方式へと変更しビット精度を制限することで学習に係る演算の軽量化を行い，これを踏まえた仮想のアーキテクチャの考案を行った．本研究では，従来の AI 専用ハードウェアに見られるような推論処理のみではなく，学習処理も含めたアーキテクチャを構築することから，学習方法が特徴的なデータ生成を行う AI モデルである Generative Adversarial Networks(GAN)[49] を用いた評価を行った．以下に，GAN についての説明を行う．

2.1.1　GAN: Generative Adversarial networks

GAN はその名のとおりデータ生成を行う AI モデルの一つであり，その最大の特徴はネットワークの構造にある．GAN は 2 つの NN を対立させて学習を行う．NN の 1 つは生成器（Generator）と呼ばれるデータを作り出す NN, もう 1 つは識別器（Discriminator）と呼ばれるデータの真贋を区別する NN である．式 (2.1) に GAN の損失関数を示す．$G(z)$ はノイズを入力したときの生成器の出力，$D(x)$ は入力が訓練データのときの識別器の

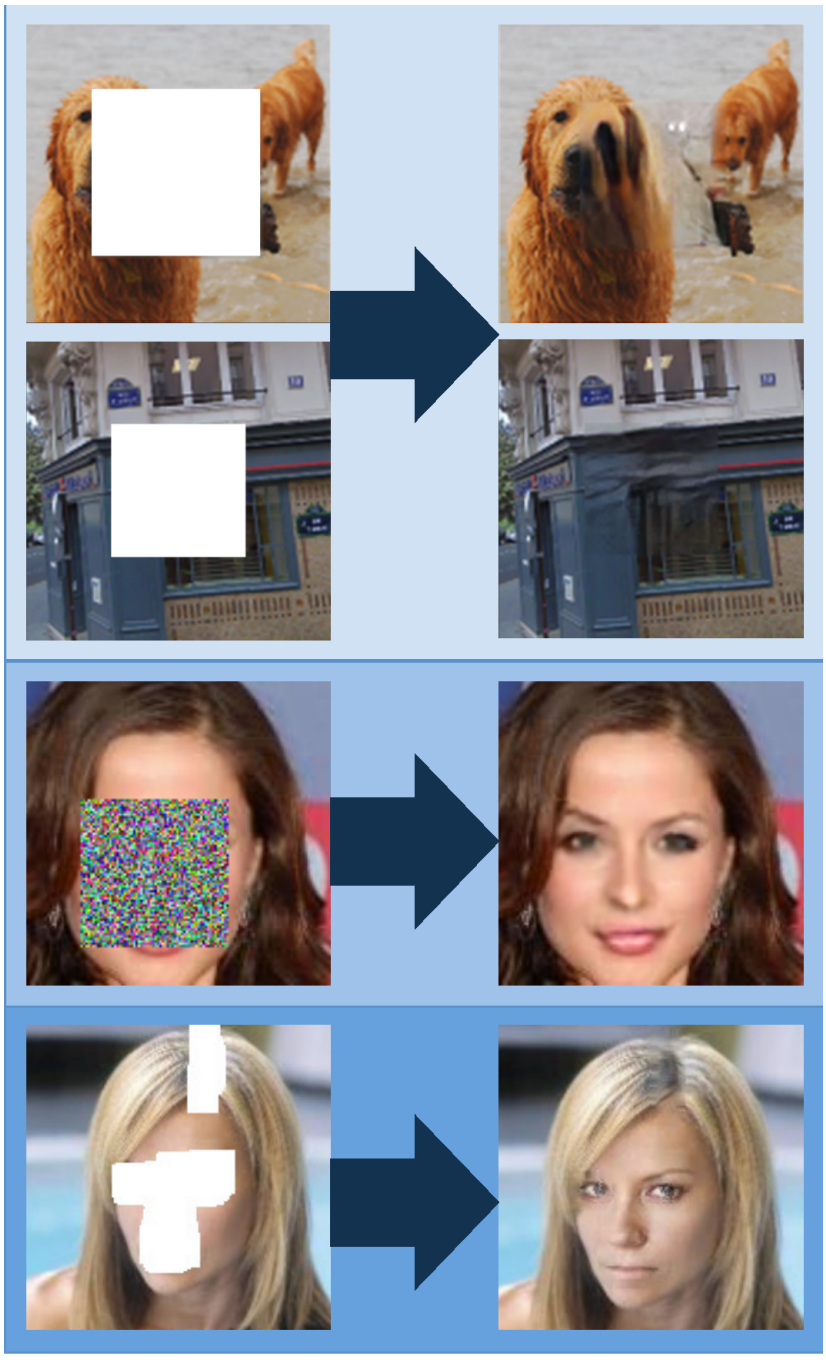

図 2.1　Context Encoder[50]（上）は GAN を利用した初期の画像補完手法である．背景など比較的単純な画像の復元に比べ，顔といった複雑な要素を持つ画像の復元には適さない．Face Complition[51]（中）は GAN を利用した顔画像の補完に特化した手法である．画像全体とマスク部の真贋を 2 つの識別器を用いて区別し，さらに顔を解析するネットワークも用いている．Globally and Locally Consistent Image Completion[52]（下）も同様に画像全体とマスク部分のそれぞれの真贋を 2 つの識別器を用いて区別する．顔画像だけではなく背景画像の補完も可能である．

出力，x は入力する訓練データ，z はノイズを表す．

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

生成器 G はノイズ z を入力としてデータの生成を行い，生成したデータ $G(z)$ は識別器 D の入力となる．生成器 G の目的は，識別器 D が誤識別を起こすようなデータを生成することであり，式 (2.1) を最小化するように学習する．また，識別器 D の目的は，は入力として受け取ったデータが訓練データであるか，生成されたデータであるか正しく識別することであり，式 (2.1) を最大化するように学習をする．つまり，式 (2.1) は識別器 D の性能が向上すると，$log(D(x))$ と $log(1 - D(G(z)))$ が大きくなり，生成器 G の性能が向上すると，$log(1 - D(G(z)))$ が小さくなるという構造となっている．生成器と識別器を交互に学習させることにより，片方の AI モデルの成長によってもう一方の AI モデルの成長も促進される．この繰り返しによって，生成器は本物のデータと見紛うようなデータが出力可能となる．また，2 つの NN は特に決まった構造を持っておらず，多層パーセプトロンや畳み込み NN などで構成することも可能である．

GAN のソフトウェアに関する研究は興隆の真っ只中にあり，単にデータを生成するだけに留まらず図 2.1 に挙げるような画像の補完といった難しいタスクをもこなしている．しかし，ソフトウェア実装が隆盛を極めている一方で GAN のハードウェアに関する研究事例は少なく，これは学習処理が必要なことに起因すると考えられる．また，比較的新しい AI モデルである GAN の学習に使用される最適化手法はその新しさゆえに Adam[16] といった最新鋭のアルゴリズムである．しかし，従来行われてきたの NN のハードウェア実装は学習した重みを用いて計算を加速させることや省電力高効率に計算を行うこと等の推論処理に特化している．学習処理は大規模な GPU サーバ上でソフトウェア上で行われており，我々の調査する限りにおいて Adam 等の学習アルゴリズムを効率よく実行する専用ハードウェアは存在していない．そのため，ハードウェアで学習を行う場合は SGD といった旧来の最適化手法に頼らざるを得なくなるのが現状である．以下に最適化手法についての説明を行う．

2.1.2 最適化手法

TensorFlow や Pytorch などの機械学習ライブラリでは最適化手法が用意されていることが多く，実装する際にはその詳細を知らなくても比較的楽に実装できる．しかし，ハードウェア実装の場合では演算内容を論理回路を用いて構成していくために詳細を知らずに実装することはできない．今回のハードウェア実装にあたって用いる最適化

手法は最も基本的な手法である確率的勾配降下法（SGD）とした．この SGD について説明する．

勾配降下法

まず，基本となる誤差関数 $J(\theta)$ を最小化する勾配降下法を式 (2.2) に示す．θ は重みやバイアスといった NN モデルのパラメータであり，η は学習率である．

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.2}$$

訓練データ全体に対して損失関数の勾配を計算しパラメータを更新するため 1 回の学習速度が遅くなり，また，メモリに収まらないデータセットに対しては処理できない．1 度の学習に訓練データ全体 (バッチ) を用いることからバッチ勾配降下法とも呼ばれる．

確率的勾配降下法

次に，確率的勾配降下法を式 (2.3) に示す．

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.3}$$

勾配降下法とは異なり訓練データ全体に対して損失関数の勾配を計算するのではなく，訓練データの中から適当に抽出した 1 つのデータを用いて損失関数の勾配を計算する．SGD は学習速度の増加を見込めるだけではなく，学習途中での訓練データの追加にも対応できる．

ミニバッチ勾配降下法

そして，現在の NN 学習の主流となっているミニバッチ勾配降下法を式 (2.4) に示す．

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i+n)}; y^{(i+n)}) \tag{2.4}$$

SGD とは異なり，訓練データの中から適当に 1 つ抽出するのではなく，n 個抽出し損失関数の勾配を計算する．データセットであるバッチを n 個のサブセット (ミニバッチ) に分割し，ミニバッチ毎に学習を行う手法である．n の値はハードウェア実装を考えると 2 のべき乗である方が望ましい．通常，ミニバッチ勾配降下方が用いられる場合にも SGD という用語が使用され，一般的に SGD と称されているのは専らこちらの方である．本論文で用いる SGD もこちらを意味する．

ここで，GANの最大の特徴は二つのモデルを対立させて交互に学習するという学習方法にある．そのため，従来の推論処理に特化したアーキテクチャではエッジAI上でGANらしさを活かした応用アプリケーションの開発を見込めない．その観点からも学習処理を含めたアーキテクチャが必要であり，本章ではハードウェア指向のGANを実装し，ゼロからの学習が可能であることを示す．本章では，NNの学習に用いられる最も基本的な手法であるBPとSGDのハードウェア向けアルゴリズムを，その場での学習が重要な意味を持つGANへの実装に向けた説明を以下の通りに行う．まず，Pythonの数値計算ライブラリNumpyを用いてGANの実装を行い，この実装をもとに固定小数点化した際のビット精度を求める．シミュレーション結果を用いてSGDを固定小数点化した手法を用いた場合でもGANの学習が可能であることを示す．次に，提案するGANアーキテクチャの説明を行う．提案アーキテクチャは推論処理とBP演算とを行う二種の既存アーキテクチャから構成されており，そのアーキテクチャの説明とハイパーパラメータによって決まるパイプラインチャート等を示す．また，各種パラメータによって性能やリソースがどのように変化するのかを示す．

2.2　提案手法

近年の AI プログラミングの流行に TensorFlow や Pytorch といった機械学習ライブラリの果たした役割は大きいと考えられる．これらのライブラリ利用することで，推論処理や学習処理の個々の演算の中身を理解せずとも AI モデルの実装が可能となった．このブラックボックス化によって研究や応用において参入のハードルは下がったものの，一方でハードウェアへの実装を考えた場合にはこのブラックボックスを解き明かす必要がある．また，ハードウェアとソフトウェアでは小数点方式の違いもある．浮動小数点方式による潤沢な数値表現を行えるソフトウェア実装に比べ，高効率な演算を目指すハードウェア実装では固定小数点方式であることが望ましい．一方で数値の表現力不足により，固定小数点のビット精度によっては学習が進まないといった問題が生じる恐れがある．しかし，むやみにビット精度を取るとリソースが増大してしまう恐れがあるため，性能とリソースとのトレードオフから固定小数点のビット精度を求めなければならない．

2.2.1　多層パーセプトロンによる予備実験

前述の通り GAN はデータを生成するモデルとデータの真贋を判別するモデルという 2 つの NN から構成される．仮に 2 つの NN が 3 層の多層パーセプトロン (MLP) で構成されているとすると，識別器の学習の際には 3 層の MLP として学習を行う．一方で，生成器の場合には識別器の出力側から逆伝播を行う関係から，3 層の MLP ではなく 5 層の MLP とみなして学習を行う．MLP において層が増えると勾配が消失してしまう可能性が生じてくる．そのため，勾配消失問題の確認と，また，SGD の実装の確認もかねて Python の数値計算ライブラリである Numpy を用いて，5 層の MLP の実装を行った．

図 2.2 に今回実装した MLP の層構造を示す．ミニバッチサイズは実行時間の都合上から 10 とした．通常 MNIST は 28x28 サイズであるが，GAN を模した多層パーセプトロンという関係から生成器の入力層のサイズである 100 に近づけるようにするためサイズを 14x14 に縮小した．また，MNIST は 0〜9 の 10 クラス存在しているが，識別器の真偽の 2 クラス分類に対応させるために偶数か奇数かの 2 クラス分類を行った．中間層と出力層の活性化関数は Leaky ReLU と Sigmoid 関数の 2 種類を用いた．入力層側から活性化関数を Sigmoid，Sigmoid，Leaky　ReLU，Sigmoid とした場合，この組み合わせを S-S-L-S のように示す．

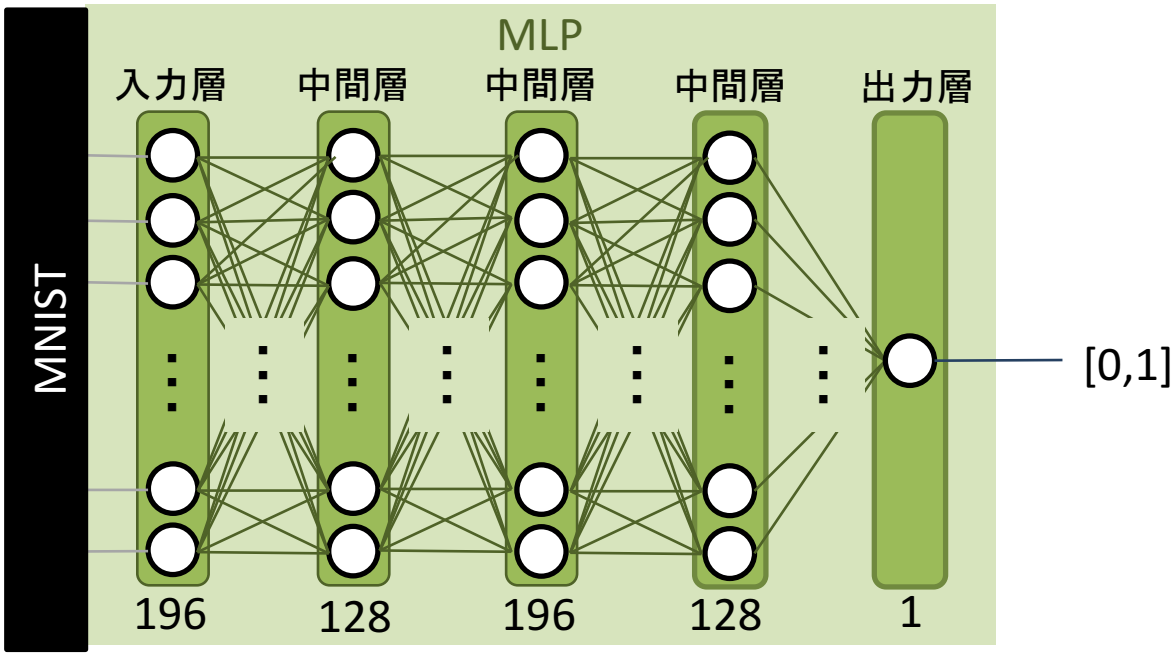

図 2.2　GAN を模した 5 層パーセプトロンによる 2 クラス分類．与えられた画像が偶数か奇数かの判別を行う．中間層の活性化関数の組み合わせにより，勾配消失が起こる組み合わせを探索する．活性化関数は Leaky ReLU と sigmoid 関数を用いた．

ここで，活性化関数として図 2.3 に示すような ReLU 系の関数を使う場合には重みの初期化に注意を払わなければならない．活性化関数が Sigmoid 関数であるときにはあるユニットの出力の最大値は 1 となるが，ReLU 系の関数を使用するとその線形性から，逆伝播の際に勾配の値が爆発的大きくなるという問題点がある．そのため，式 (2.5) に示す He Initialization[53] を用いる等で勾配の爆発問題に対応する必要が生じる．

$$w_l \sim N(0, \sqrt{\frac{2}{N_l}}) \tag{2.5}$$

$$b_l = 0 \tag{2.6}$$

本初期化方法は，平均が 0，分散が $\sqrt{\frac{2}{N_l}}$ の正規分布を用いて（N_l は l 層目のユニット数）重みの初期化を行い，各層のバイアス（b_l）は 0 で初期化するものである．今回の実装においても He Initialization を用いた．

表 2.1 に活性化関数の組み合わせとその時の精度を示す．各 5 回ずつ実験を行った．2 クラス分類であることから識別精度が 50%付近である S-S-S-S の組み合わせは勾配が消失してしまい，上手く学習が行えていない可能性が考えられる．また，Leaky ReLU を多く用いた方が識別精度が高くなることが確認できる．これは Leaky ReLU 関数は微

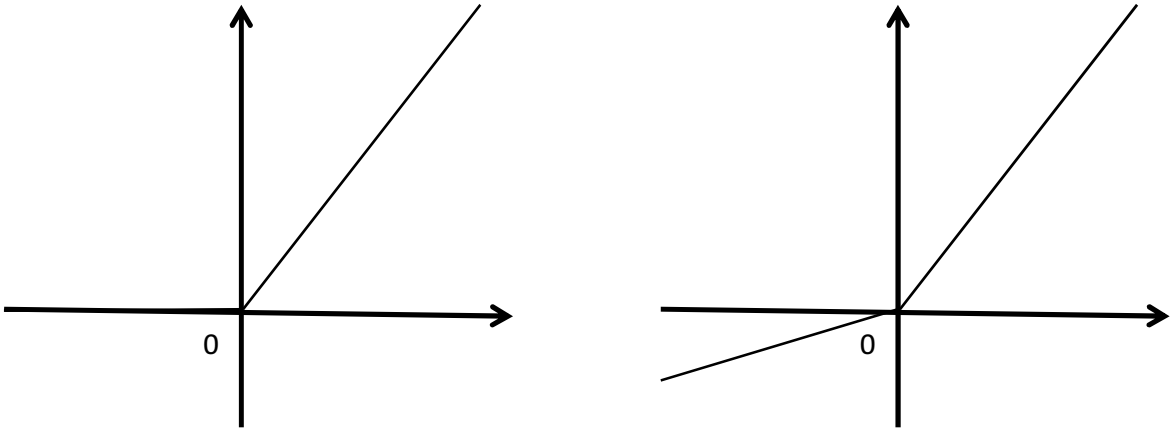

図 2.3　左 ReLU．ReLU は入力が負の領域にあるときは 0 を出力し，正の領域にあるときは線形に出力する．　右 Leaky ReLU．Leaky ReLU は入力が負の領域にあるときは 0.2 を乗算して出力し，正の領域にあるときは線形に出力する．

表 2.1　活性化関数の組み合わせと識別精度．活性化関数に Leaky ReLU を持つ層が増えるほど識別精度が増している．逆伝播には微分が行われていることから，Sigmoid 関数を多く用いると勾配が消失してしまい正常に学習が行われなくなる．Leaky ReLU は微分をすると定数になることから勾配消失の危険はない．

S-S-S-S	S-S-L-S	L-S-L-S	L-L-L-S	L-L-L-L
58.23%	64.28%	79.70%	81.52%	89.18%
61.88%	74.14%	78.64%	82.46%	90.45%
69.44%	70.46%	76.40%	82.54%	90.83%
64.99%	76.56%	77.69%	82.28%	90.93%
49.79%	74.32%	74.42%	81.95%	89.23%

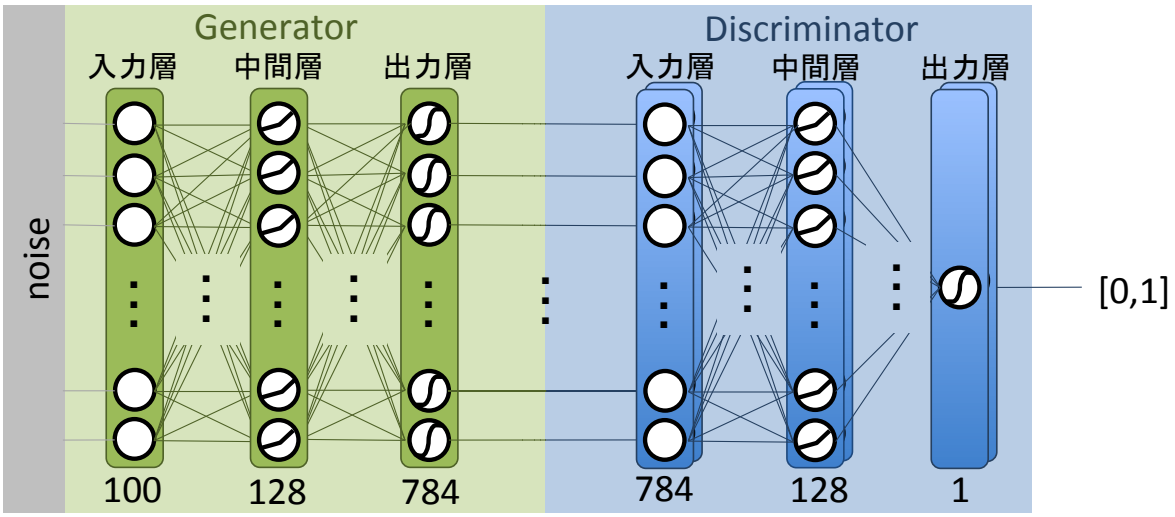

図 2.4 Numpy 実装による GAN の層構造．それぞれの中間層の活性化関数は Leaky ReLU を用いて，出力層の活性化関数は Sigmoid 関数を用いた．また，識別器は本物のデータに対する内部状態と偽物のデータに対する内部状態の 2 つの状態を必要とすることから，識別器の各層を 2 つずつ用意してある．

分をしても 0 にならないことから勾配が消えず，入力層側まで修正情報が伝わり重みの更新が予期した通りに行えていると考えられるためである．この実験結果から，活性化関数の組み合わせは L-S-L-S とした．識別精度が高ければ高いほどより良い性能を得る可能性があるが，GAN への実装を踏まえると全ての活性化関数に Leaky ReLU を使用することはできない．つまり，GAN であることを考えた時には，2 番目と 4 番目の中間層は生成器と識別器の出力層に相当し，[0,1] の範囲で値を出力しなければならないことから Sigmoid 関数を用いる必要がある．ただし，生成器と識別器の中間層数を増やし，3 層から 4 層の多層パーセプトロンへ変更する場合は，L-L-S-L-L-S とするのが望ましいと考えられる．

2.2.2 Numpy を用いたソフトウェアシミュレーション

図 2.4 に本シミュレーションで実装した GAN の層構造を示す．活性化関数は前述の通り生成器と識別器両方において中間層は Leaky ReLU，出力層は Sigmoid 関数を用いた．ミニバッチサイズは 64 とし，MNIST 画像を訓練データとして使用し，実装上の観点から 10 クラスあるうちから 1 クラスのみのデータを抽出し学習を行った．識別器の各層は本物である訓練データに対しての内部状態と偽物である生成されたデータに対

しての内部状態の二つが存在するため，これらの保存を目的に各層も二つずつ用意した．ただし，パラメータは共通のため一つしか用意しない．識別器の学習において最小化すべき誤差関数は式 (2.1) から次のように導出される．

$$-\nabla_{\theta_D}\frac{1}{m}\sum_{i=1}^{m}[\log(D(x^{(i)}))+\log(1-D(G(z^{(i)})))]s \tag{2.7}$$

m はミニバッチサイズである．式 (2.7) の第一項が本物の訓練データに対する識別結果に対応し，第二項が偽物の生成されたデータに対する識別結果に対応する．逆伝播を行う際には本物のデータに対する識別誤差と偽物のデータに対する識別誤差とのそれぞれから重みの更新量を算出し，二つの更新量を足し合わせたものを用いて重みの更新を行った．生成器の学習において最小化すべき誤差関数は式（2.1）から次のように導出されるが,

$$\nabla_{\theta_G}\frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(z^{(i)}))) \tag{2.8}$$

実際の学習では次に示す式（2.9）を最小化する方がより大きな勾配を得られるため良いとされている [54] ため，次の式を使用した．

$$\nabla_{\theta_G}\frac{1}{m}\sum_{i=1}^{m}\log(D(G(z^{(i)}))) \tag{2.9}$$

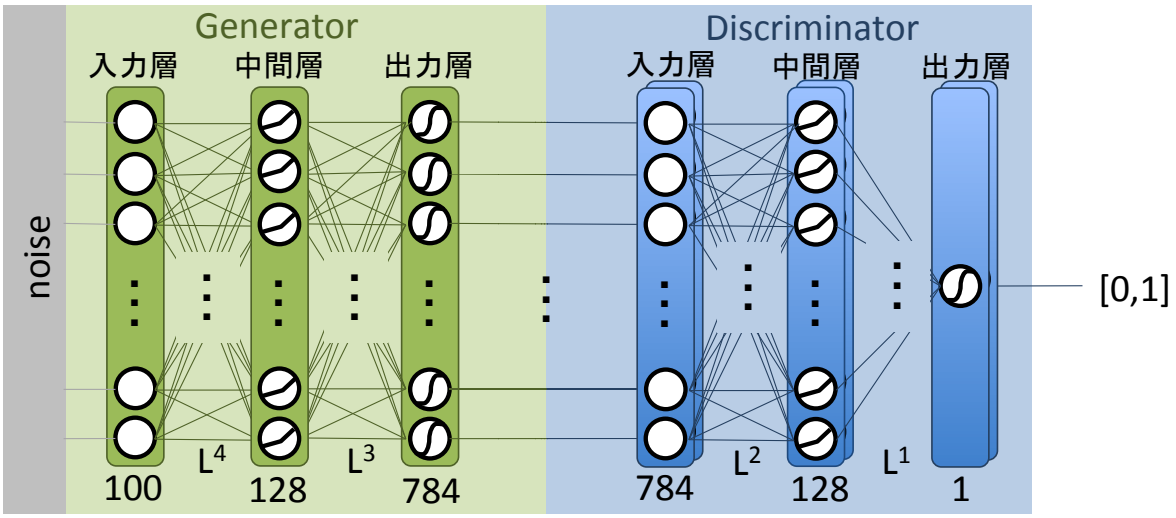

図 2.5 本シミュレーションにおける GAN のモデル構造と固定小数点化の施し方．推論処理時のビット精度を決定した後に BP 演算のビット精度を決定する．BP 演算のビット精度は識別器の出力層側から 1 層ずつ順に（L^1 から順に）決定した．

続いて，固定小数点演算に向けたビット精度探査について記述する．ハードウェア実装をする際に扱う数値表現方式は，リソースの観点から浮動小数点方式ではなく固定小数点方式の方が好ましい．この時固定小数点のビット精度が小さいほどメモリ容量が少なくなる．推論処理ではビット精度は小さくて済むと予想されるが，BP 演算では勾配の情報やパラメータの更新量は小さな値となるために高いビット精度が要求されることが予想される．そのため，本探査では，まず整数部と小数部ともに十分大きなビット精度で固定小数点化し，そこから徐々にビット精度を小さくしていくというような方式を取った．まずは，推論処理のみを固定小数点化し，その次に BP 演算を出力層側から順に固定小数点化するというように段階を経て行った．図 2.5 に BP 演算の固定小数点化について段階の踏み方について示す．L^n は出力層から n 層目までを示しており，BP 演算は出力層側から行うため識別器の出力層側から順に数字を振っている．

2.2.3 GAN アーキテクチャ

提案アーキテクチャは，以前本研究室において取り組んでいた推論処理を行うアーキテクチャ [55] と BP 演算を行うアーキテクチャ [56] の二つを用いて構成されている．これらを利用して GAN アーキテクチャを構成した際のリソース使用量やレイテンシやスループット等の性能の見積りを行った．専用ハードウェアによる処理の特徴の一つ

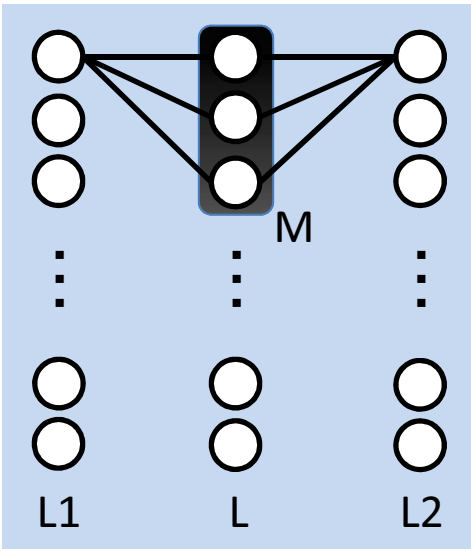

図 2.6　Brein Memory の演算方法．3 層のパーセプトロンを 1 組として考える．中間層のニューロンを同時に計算する個数をパラメータ M によって定義している．M が増えるほど要求リソースは増大するがレイテンシは小さくなる．

として挙げられるのは並列性である．今回の GAN でいうのなら生成器と識別器は同時に動作していても構わないことから，並列に動作させることでレイテンシの削減が見込める．また，識別器を 2 つ用意することで本物のデータと偽物のデータの識別を同時に行うことも考えられ，この 2 つもまた並列に動いていても問題はないはずである．このようにハードウェアの特徴を生かしたアーキテクチャを提案する．

ここで，推論処理を行うモジュールとして Brein Memory のシステムを応用することを検討している．Brein Memory は図 2.6 に示すような処理を行う構造を持ち，入力層から中間層へは逐次入力・並列出力を行い，中間層から出力層へは並列入力・逐次出力を行う．M は時分割に計算するニューロンの個数であり，時分割度は次のように求められる．

$$\lceil \frac{L}{M} \rceil \tag{2.10}$$

L は中間層のニューロンの個数である．式 (2.10) を用いて推論処理にかかるクロック数は以下の式（2.11）で求められる．

$$\lceil \frac{L}{M} \rceil (L_1 + L_2) \tag{2.11}$$

L_1 は入力層のニューロンの個数であり，L_2 は出力層のニューロンの個数である．M の値が大きくなるほどレイテンシは減少するが必要となる乗算器の個数が増大する．

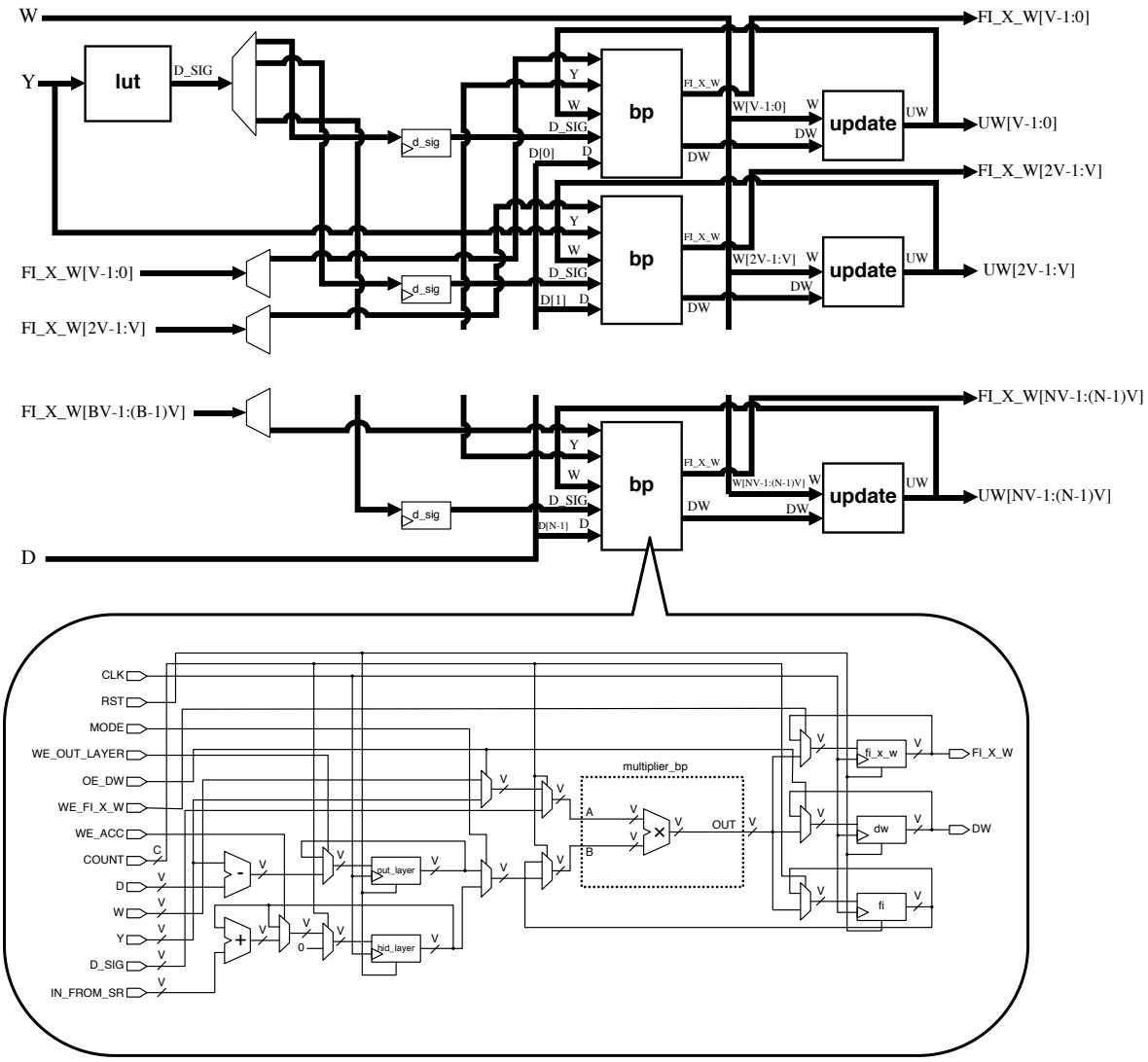

図 2.7　back 演算部の演算部分．BP モジュールによって出力 Y か前層の重みに関する値 FI_X_W と教師信号 D から重み更新の値が求められる．FI_X_W は次の層の重みの更新に利用されるためシフトレジスタに入力される信号である．

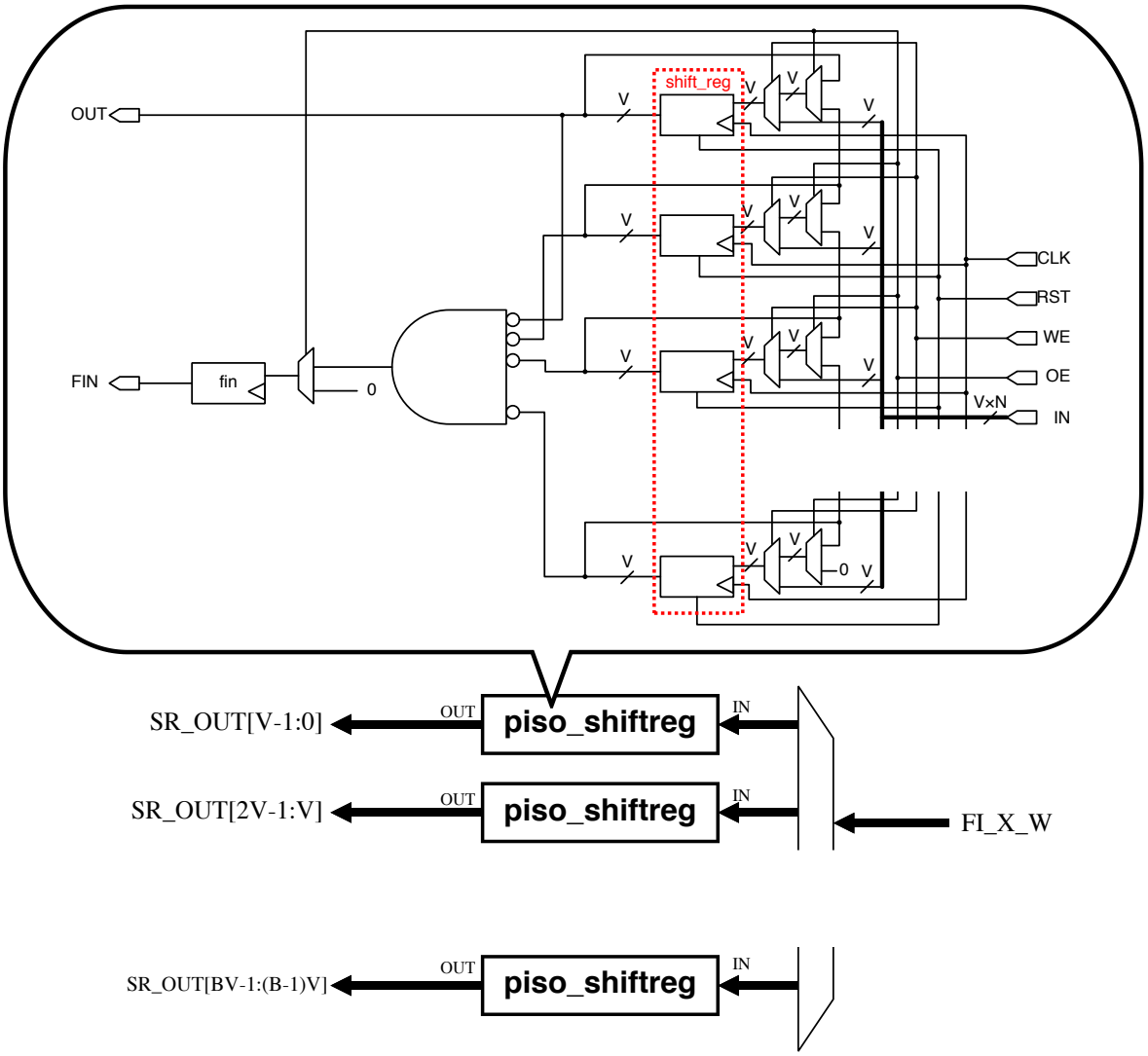

図 2.8　back 演算部のシフトレジスタ部分．BP モジュールによって求められた重み更新の値は次の層の重みの更新に利用されるためシフトレジスタによって保持される．このシフトレジスタを用意する個数がパラメータ B として定義されている．B が増えると要求リソースが増大するが，レイテンシは小さくなる．

続いて，BP 演算を行うモジュールとして図 2.7 と図 2.8 に示すようなアーキテクチャを用いる．本アーキテクチャはパラメータの更新量を算出する演算部分と重みの更新量を格納するシフトレジスタ部分に分かれている．図 2.7 に示す演算部分は各ニューロンからの出力を受け取り，シグモイド関数の導関数を通した結果を出力するルックアップテーブルである lut モジュールと，BP 演算を行う bp モジュール，現在のパラメータの値と bp モジュールから出力されるパラメータの更新量を加算して更新後のパラメータの値を算出する update モジュールで構成されている．また，bp モジュールおよび update モジュールは一層に含まれるニューロンの数だけ用意する．算出された更新後のパラメータの値とパラメータの修正情報は乗算されシフトレジスタ部へと送られる．シフトレジスタ部分は図 2.8 に示すようにパラレルイン・シリアルアウトのシフトレジスタを含む piso_shiftreg モジュールによって構成されている．piso_shiftreg モジュールは演算部分を構成している bp モジュールと同じ数だけ用意することが理想であるが，大規模なネットワークの実装を考えた場合膨大なリソースが必要となることを避けるため，piso_shiftreg モジュールの数を調整できるようになっている．この piso_shiftreg モジュールの個数を用いて BP 演算にかかるクロック数が以下の式（2.12）で求められる．

$$\sum_{l=1}^{L}\{(4B+N_l)\lceil\frac{N_l}{B}\rceil+4(N_l\%B)+N_l+4)\} \tag{2.12}$$

B は用意する piso_shiftreg モジュールの個数であり，N_l は l 層目のニューロンの個数である．

図 2.9 に今回提案するアーキテクチャのブロック図の概観を示す．各矢印に表示されている数字は稼働周波数が 50MHz としたときの必要バンド幅の最大値である．必要バンド幅は並列度 F_G，F_D や時分割度パラメータ M，バックワードモジュールのパラメータ B に関係しており，並列度やパラメータが増大しクロック数が短くなると必要バンド幅も増大する．各種モデルのパラメータや訓練データはメモリに保存されており，推論処理や BP 演算を行う前にロードを行うことや，BP 演算を行いパラメータの更新を行う際に逐次アクセスする必要がある．BP モジュールは生成器と識別器で別々に用意するのではなく一つのモジュールを共用することを検討している．

図 2.10 に今回提案するアーキテクチャのパイプラインチャートの概観図を示す．ミニバッチサイズは 64 とし，フォワードモジュールの時分割度は 1（M=128）とした．また，生成器のフォワードモジュールを 8 つ並列に動作するように用意し（この並列度を F_G とする），識別器のフォワードモジュールを本物のデータを処理するモジュールと生成データを処理するモジュールの 2 つに分け，それぞれを 4 つずつ並列に動作す

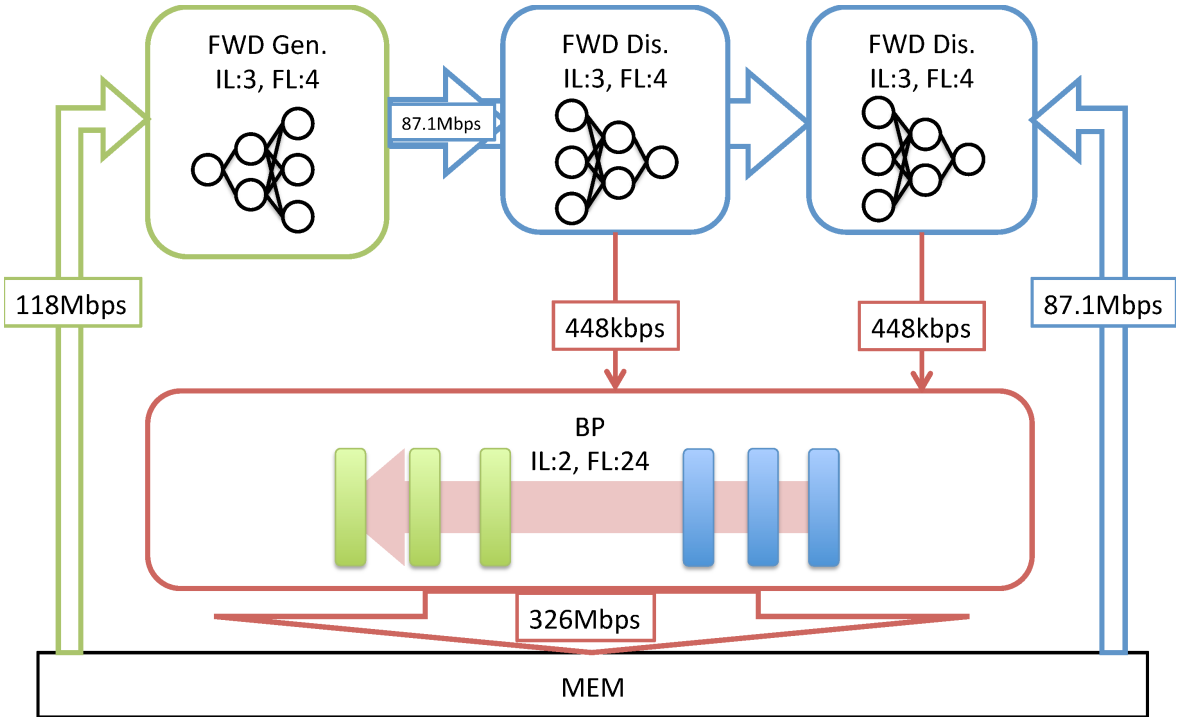

図 2.9　提案アーキテクチャのブロック図．データや重み，バイアスの値はメモリに保存されているため，演算を行うにはロードする必要がある．その時に必要となるバンド幅を稼働周波数を 50MHz として見積りを行った．なお，コントローラモジュールについては省略している．

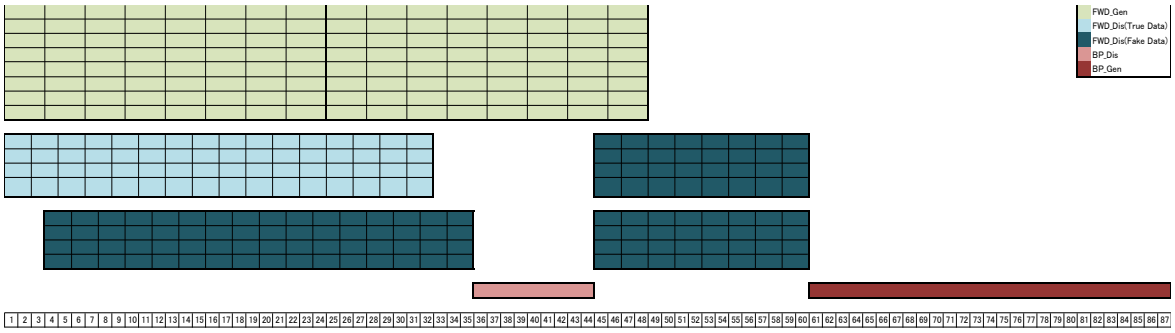

図 2.10　提案アーキテクチャのパイプラインチャート（F_G=8，F_D=4）．生成器の推論処理はモジュールを 8 つ用意し，8 並列でデータの生成を行う．識別器の推論処理は本物のデータに対するモジュールと偽物に対するモジュールを各 4 つずつ用意し，4 並列で演算を行う．BP モジュールは 1 つだけ用意しコントローラによって制御を行う．生成器の学習には識別器も 8 並列で演算を行う．

るように用意した (この並列度を F_D とする). バックワードモジュールの piso_shiftreg モジュールの個数を決める B の値は 99 とした．下部にある数字はクロック数を示しており（単位は 100），今回提案したアーキテクチャのレイテンシは 8700 クロックであると見積もることができた．GAN の学習は生成器と識別器いずれにおいても，データを生成して識別しその誤差を用いて学習するという手順を踏まなければならないために，モジュールが休む時間がでないようにする並列化を行うのが困難である．また，生成器か識別器が逆伝播を行っているときには推論処理を行えないことも並列化が困難となる要因になっている．フォワードモジュールの並列度 F_G，F_D や時分割パラメータ M，バックワードモジュールのパラメータ B によってパイプラインチャートは変化するため，生成器の学習時に識別器を 2 つずつ並列に動作させることができない可能性も存在する．

ソフトウェアシミュレーションと提案アーキテクチャの評価は次の節にて行う．

2.3　評価

本節ではハードウェア指向 GAN の評価と提案アーキテクチャの評価を行う．ソフトウェアによる GAN の研究は隆盛を極めているが，我々の調査の限りにおいて GAN のハードウェア化に関する先行研究は存在しない．また，GAN そのものが比較的新しいシステムのため学習に用いられる最適化手法も Adam 等の高度な最適化手法が用いられているが，本提案手法で用いる SGD は最も基本的な最適化手法である．そのため，Adam 等の高性能な最適化手法で学習されてきた GAN において，SGD を使用した際にも学習が可能であるかの調査が必要となる．また，SGD を用いて GAN の学習が可能であった場合でも，ハードウェア実装にあたって重要な固定小数点化した際に性能が出るのか，ビット精度を削った場合にはどれくらい性能が変化するのかも調査する必要がある．そこで，前述のシミュレーションによる GAN の実装を用いて，SGD による学習とビット精度の探査を行った．また，前節では GAN のアーキテクチャの提案も行った．提案したアーキテクチャのフォワードモジュール，バックワードモジュールには並列度や時分割度といったパラメータが存在し，このパラメータにより性能が変化する．ハードウェア実装を考える際にまず用いられるのは FPGA であり，FPGA にもリソースの潤沢なものから少ないものまで種々存在する．つまり，FPGA チップの性能により取り得るパラメータは変化し，前節で求めたパイプラインチャートは現実のボードを無視した提案である可能性も存在する．そのため，Intel の FPGA チップの使用を前提としてパラメータによってリソースや性能がどのように変化するかの評価を行った．また，各 FPGA チップを用いた場合に期待される性能と要求リソース量を評価した．

図 2.11　Numpy 実装での生成画像．学習回数が 2epoch（約 12000 回学習）の時にはぼやけているものの数字が生成されている．学習回数を増やすにつれて画像は崩壊していった．

2.3.1　ソフトウェアシミュレーションを用いた評価

Numpy を用いて実装した GAN の評価を以下に示す．ミニバッチサイズは 64，訓練データとして MNIST を用いた．MNIST には 0〜9 までの 10 クラスの手書き数字画像が存在するが，入力データのクラスが 2 クラス以上の場合に学習が発散することが確認できた．単純な識別問題において，前節の MLP を用いた予備実験では TensorFlow と同等の精度が出ていた．そのため，GAN 特有の層の構造などによって多クラスでは学習ができていないのではないかと考え，1 クラス（特定の数字）のみで学習を行った．

図 2.11 に浮動小数点方式を用いたシミュレーションによる生成器の出力画像を示す．本シミュレーションではミニバッチサイズを 64 としていることから，出力データも 64 個存在する．ここでは，64 個の画像から任意の 1 つを選択し，表示した．1epoch は使用する訓練データ全てに対して学習を行ったことを示しており，1 クラスの MNIST だと約 6000 回の学習である．2epoch（≈12000 回学習）目には本物と見紛うような画像が生成されているが，学習が進むにつれて 10epoch 目の出力のようにぼやけたような数字を生成するようになった．これはある種の Mode Collapse[54] であると考えることができる．つまり，いずれの数字においても画像の中央に白として存在しているため生成器は画像の中央に白っぽいものを出力すると識別器を騙すことができると学習した結果，ぼやけた数字を生成するようになったものと考えられる．

表 2.2　固定小数点化に関して必要なビット精度．推論処理はビット精度を多くは必要としないが，BP 演算は重み更新の値が極めて微小であることから，ビット精度を多く必要とする．

		整数部（符号含む）	小数部
FWD		3bit	4bit
BP(L^1)	δ	2bit	4bit
	Δ	1bit	16bit
BP(L^2)	δ	1bit	18bit
	Δ	1bit	24bit
BP(L^3)	δ	1bit	18bit
	Δ	1bit	24bit
BP(L^4)	δ	1bit	14bit
	Δ	1bit	20bit

続いて，固定小数点方式でシミュレーションを行った際の結果について記述する．ただし，固定小数点方式での実験においては実行時間の都合により通常 28x28 サイズの MNIST 画像を縦横半分に縮小し，14x14 サイズの MNIST 画像を用いて行った．また，ビット精度を狭めていく前に性能が落ちない十分大きなビット精度で実験を開始した．この時のビット精度は整数部（符号含む）32bit，小数部 32bit である．表 2.2 に推論処理時に必要なビット精度と BP 演算時に必要なビット精度を示す．δ は活性化関数の微分値などから求められる修正情報であり，Δ はパラメータの更新量である．推論処理に比べると BP 演算に必要なビット精度は膨大となることが確認できる．これは，BP 演算で取り扱う数値は誤差や微分値といった小さな数値であるため，これを表現するため高いビット精度が要求されるためであると考えられる．図 2.12 にミニバッチサイズを 64, 訓練データを 14x14 に縮小した 1 クラス MNIST 画像とし，BP 演算を浮動小数点と完全固定小数点とで学習を行った結果を示す．浮動小数点に比べると固定小数点の方が早い段階で数字の生成を行えている．これは，以下のことを要因として副次的に得られた効果だと考えられる．一般に，演算の固定小数点化は取り得る数値の離散化を意味する．ニューラルネットの学習は未知の関数へのフィッティングであるが，重みやバイアスといったパラメータの数が膨大であるために探索する空間が巨大である．固定小数点化を行うことはこの探索空間の制限につながるので，学習が速くなると考えられる．

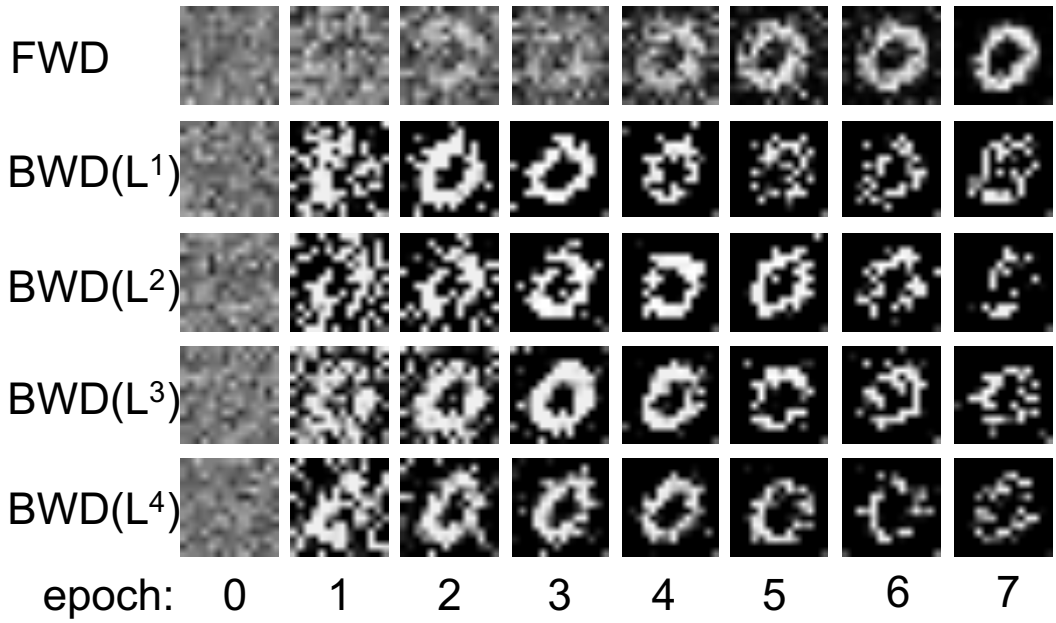

図 2.12　小数点方式の違いによる生成画像の変化．浮動小数点方式で学習を行った場合と，固定小数点方式で学習を行った場合との比較である．ビット制限を行うことによってパラメータ探索の範囲が限定され，学習が速く行えると予想される．

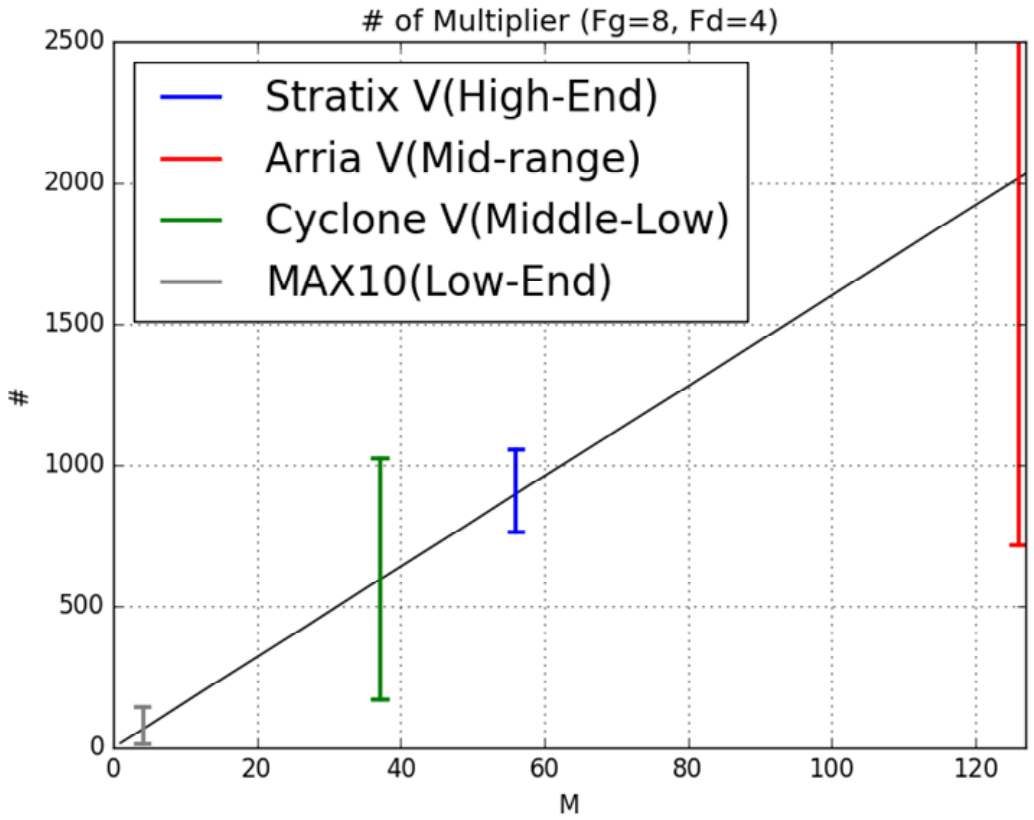

図 2.13　M の増加に伴い必要となる乗算器数見積もり．ニューロンの計算に関するパラメータ M によって必要となる乗算器数が変化する．M はニューロンの個数以上に用意しても意味がないため，今回の場合の最大値は 128 である．各ボードの乗算器数の範囲をバーによって示している．グラフとの交点は各バーの平均値である．Fg は生成器のフォワードモジュールの並列度，Fd は識別器のフォワードモジュールの並列度である．

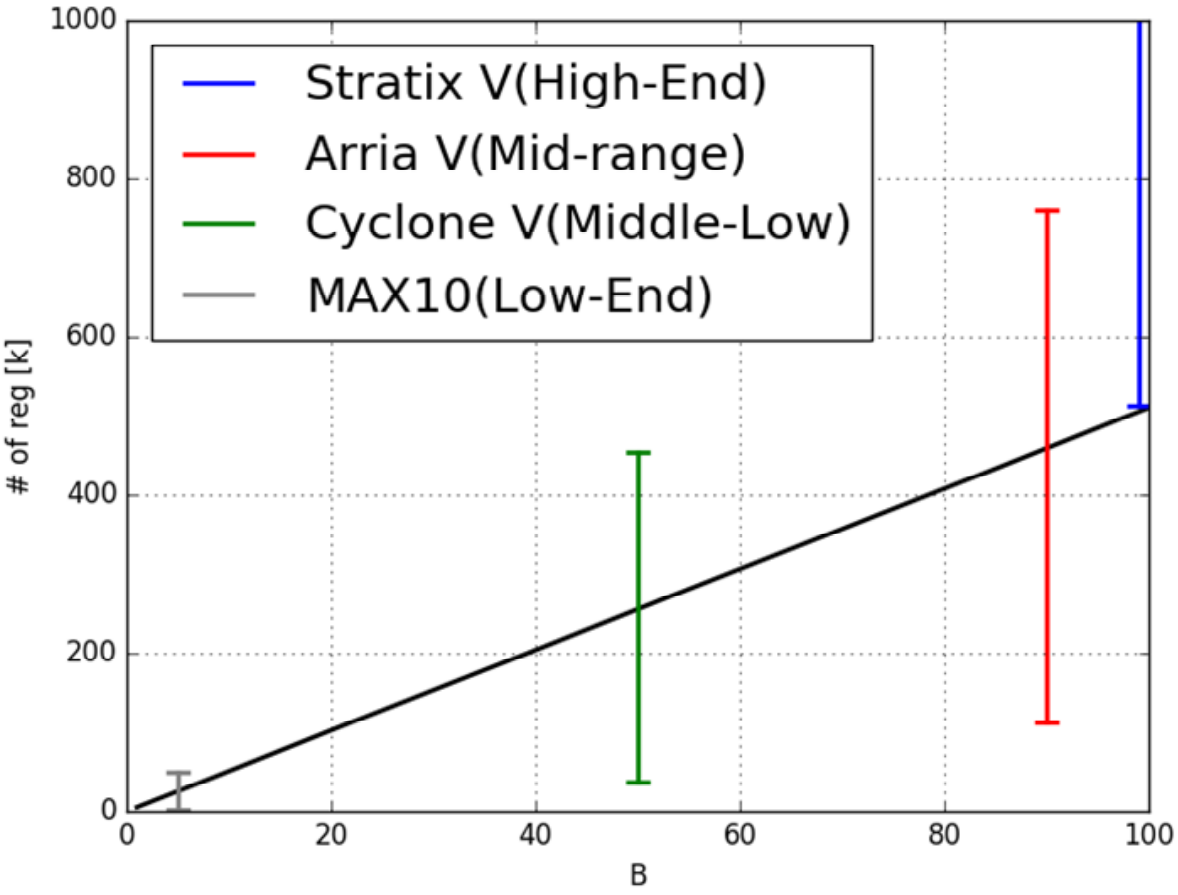

図 2.14　B の増加に伴い必要となるレジスタ数見積もり．シフトレジスタに関するパラメータ B によって必要となるレジスタの数が変化する．こちらも同様に各ボードのレジスタ数の範囲をバーによって示している．グラフとの交点は各バーの平均値である．B はニューロンの個数以上に用意しても意味がないため，今回の場合の最大値は 196 である．

2.3.2　アーキテクチャ評価

前節では，GAN のアーキテクチャを提案し，パイプラインチャートやブロック図を作成した．フォワードモジュールや BP モジュールに関するパラメータを全て考慮に入れた上で評価を行うのはその組み合わせの膨大さから非常に厳しく，今回はパイプラインチャートを作成する際に仮決めしたフォワードモジュールの並列度パラメータを F_G=8，F_D=4 に固定し，時分割度パラメータ M と piso_shiftreg の個数に関するパラメータ B によって性能やリソースがどのように変化するかを評価した．図 2.13 に M を増やすことで必要になる乗算器数を示す．ここで，時分割度パラメータ M の最大値は

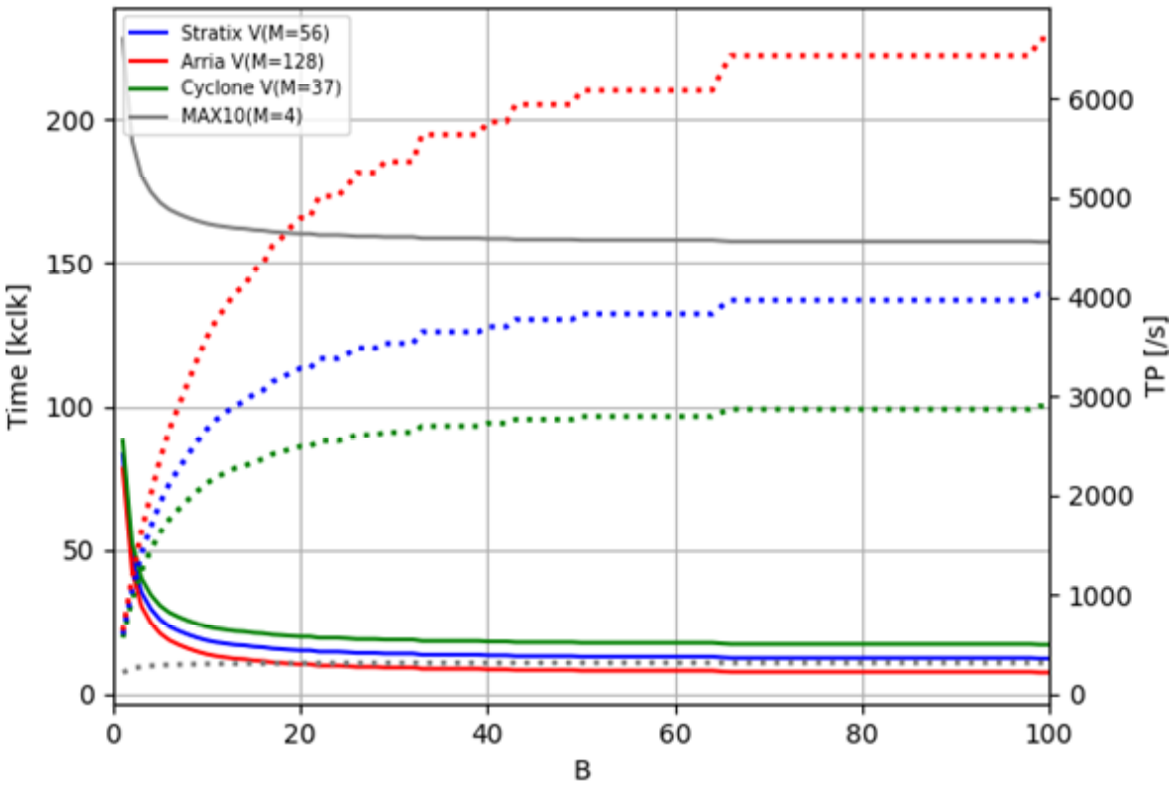

図 2.15　M と B による性能見積もり（実線がレイテンシ，点線がスループット）．スループットは稼働周波数を 50MHz として見積もりを行った．レイテンシやスループットは M と B によって変化し，ボードにはよらない．各ボードを選択した時の値 M を図 2.13 から仮決めしていることに注意されたい．（Cyclone V でも $M = 56$ にすることはできる．）

中間層のユニットの個数であるため無尽蔵に大きくすることはできないことに注意されたい．今回実装する FPGA チップとして Intel FPGA シリーズから 4 つを候補として選定した．高性能なものから述べていくと Stratix V（青色で示す），Arria V（赤色で示す），Cyclone V（緑色で示す），MAX10（灰色で示す）である．使用できる乗算器数が多いほど同時に計算できるニューロンの個数が増えるため，レイテンシは小さくなる．ボードによって取り得る M の値は変化する．例えば，Cyclone V の場合 M はおよそ 10〜60 の範囲に収まらなければならない．同様に，図 2.14 に B を増やすことで必要になるレジスタ数を示す．こちらもシフトレジスタの個数を増やすほどレイテンシは小さくなる．B の最大値はある層のニューロンの最大個数である（今回は 196）．ボードによって取り得る B の値は変化する．例えば，Cyclone V の場合 B はおよそ 5〜90 の範囲に収まらなければならない．図 2.16 に性能からボードを選択する方法を示す．図

2.15 にて求めたい性能で線を引きレイテンシとの交点で垂線をおろし，その時の B の値と図 2.14 からボードの候補が求められる．図 2.14 にてボードを示すバーと一次直線の交点の B より大きい場合はそのボードは候補から外れる．図 2.17 にボードから性能を簡単に見積もる方法を示す．図 2.13 のボードを示すバーと一次直線の交点から M の値を求める．同様にして図 2.14 から B の値を求め，図 2.15 にて求めた M と B の値からレイテンシとスループットを見積もる．図 2.15 にある M を選んだ時のレイテンシとスループット見積もりの概略図を示す．性能は，凡例に示してある M を取り得るボードで色分けされており，M によって変化することに注意されたい．例えば，Cyclone V は $M = 60$ とすることもできる．

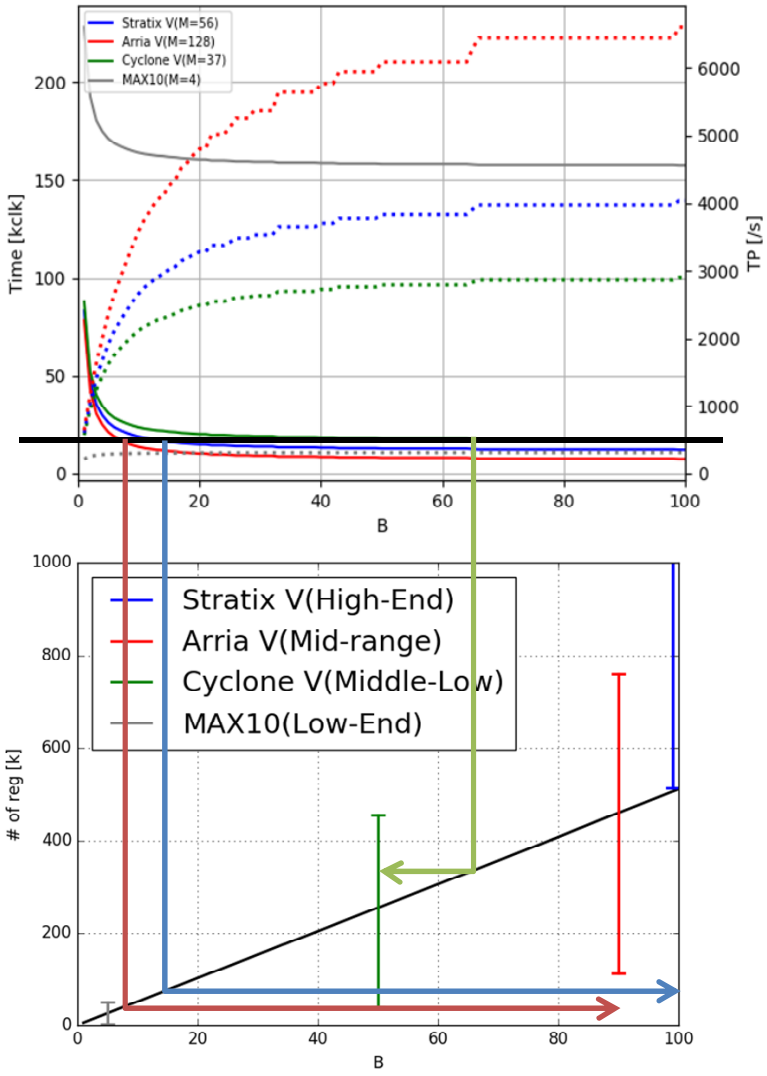

図 2.16　性能からボードの選択方法．図 2.15 の求めたい性能で横線を引き交点からそのときの B の値を求める．図 2.14 のグラフと B から求められるレジスタ数がバーの下限より少ないボードが選択できる．バーの上限より上にあった場合には選択できないが，下限より上にあった場合にはボードの性能によっては実装できない可能性も生じる．

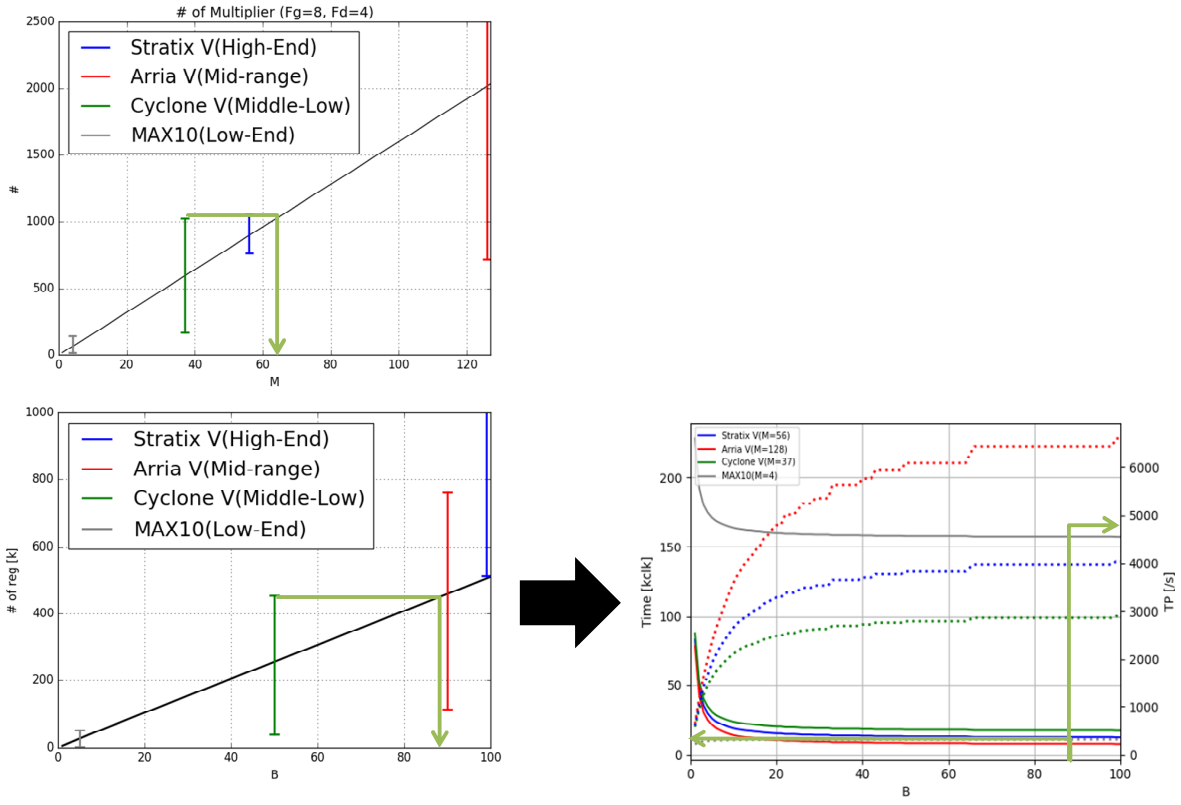

図 2.17　ボードによる性能見積もり方法．ボードの性能と図 2.13 から M の値を決め，図 2.14 から B の値を決める．求めた M と B の値から図 2.15 を用いて性能を見積もることができる．

2.4 結論

本章では，NN の学習アルゴリズムである誤差逆伝播法 (BP) と確率的勾配降下法 (SGD) のハードウェア指向アルゴリズム開発を目的とした．学習を含めたアーキテクチャの構築が重要となる GAN へ応用を考慮に入れ，ハードウェア指向の GAN の実装を行った．GAN の学習を行うハードウェアを考えた場合，従来のような演算の加速化や，高効率化を目指したニューラルネットワーク実装のような推論処理のみを行うアーキテクチャではなく，BP や SGD といった学習処理をも行う必要がある．しかし，近年利用されているような高度な最適化手法を用いることができないことが予想されるため，Python の数値計算ライブラリである Numpy を用いて最適化手法を SGD とした GAN の実装を行った．Python には TensorFlow などの機械学習ライブラリが存在するが，最適化部分がブラックボックス化されており，ソフトウェア上の工夫などによってこちらの予期している動作以上の事をしているかもしれないことから，ハードウェア実装時に障害となることが予想されるので Numpy を使用して実装を行った．まずは，この Numpy を用いた実装によって SGD の実装を通してその演算内容を理解するとともに，SGD を用いた際にも学習が可能であるかを確認した．また，GAN は 2 つの NN から成り立っておりそのどちらとも 3 層の多層パーセプトロンを使用した場合でも，生成器は識別器を介して学習を行うことから 5 層の多層パーセプトロンとして扱う必要がある．しかし，5 層の多層パーセプトロンでは活性化関数の組み合わせ次第によっては勾配消失が生じて学習ができない可能性が出てくる．そのため，予備実験において GAN を模した 5 層の多層パーセプトロンによる 2 クラス分類を行い，活性化関数の組み合わせを変更し，十分に学習が行えているかを検証した．効率的な演算を行うハードウェア実装を考えると数値表現に係る小数点方式は浮動小数点方式から固定小数点方式の方が望ましい．固定小数点方式へと変更した場合に必要となる推論・学習処理時のビット精度を求めた．これらを踏まえてハードウェアアーキテクチャを考案し推論処理と BP 演算に既存のモジュールを利用した場合のクロック数を求め，パイプラインチャートを作成した．また柔軟な実装ができるように上記モジュール内にハイパーパラメータを 2 つ用意し，このパラメータによって性能や要求リソースがどのように変化するのかを示した．本研究を通じて学習アルゴリズムを固定小数点化しただけではハードウェアアーキテクチャの観点から改善点が多いことが判明した．次章からはこれを解決するようなハードウェア実装を前提とした学習アルゴリズムの提案を行う．

参考文献

[12] Z. Zou, Y. Jin, P. Nevalainen, et al. “Edge and Fog Computing Enabled AI for IoT-An Overview”. In: *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 2019, pp. 51–56. DOI: 10.1109/AICAS.2019.8771621.

[15] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[16] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). arXiv: 1412.6980.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, et al. **27**. Curran Associates, Inc., 2014.

[50] D. Pathak, P. Krahenbuhl, J. Donahue, et al. “Context Encoders: Feature Learning by Inpainting”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[51] Y. Li, S. Liu, J. Yang, et al. “Generative Face Completion”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[52] S. Iizuka, E. Simo-Serra, and H. Ishikawa. “Globally and Locally Consistent Image Completion”. In: *ACM Trans. Graph.* **36**.4 (July 2017). DOI: 10.1145/3072959.3073659.

[53] K. He, X. Zhang, S. Ren, et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[54] I. Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1701.00160.

[55] K. Ando, K. Ueyoshi, K. Orimo, et al. “BRein Memory: A 13-Layer 4.2 K Neuron/0.8 M Synapse Binary/Ternary Reconfigurable in-Memory Deep Neural Network Accelerator in 65 Nm CMOS”. In: *2017 Symposium on VLSI Circuits*. 2017, pp. C24–C25. DOI: 10.23919/VLSIC.2017.8008533.

[56] 伊藤 健之. 動画像のシーン分類を可能にする新規特徴抽出法とそのニューラルネットワークアーキテクチャに関する研究. 北海道大学大学院情報科学研究科修士論文, 2016.

第3章　エッジAIに向けた省電力・省リソース学習法とそのアーキテクチャ

3.1　導入

第 2 章では，今日のニューラルネットワーク (NN) の学習に使われる手法で最も基本的な誤差逆伝播法 (BP) と確率的勾配降下法 (SGD) のハードウェア指向アルゴリズムとアーキテクチャについて述べた．本章では，実空間での AI 応用の観点から重要性を増しているエッジ AI コンピューティングに向けた新規オンライン学習アルゴリズムについて提案を行う．現在，最も成功している機械・深層学習システムの多くは、グラフィックスプロセッシングユニット（GPU）を用いて構成されるサーバ等のクラウドデバイスに依存している．NN は、SGD[15]、Adam[16] などの学習アルゴリズムを用いて望ましい出力が得られるまで学習を行っている．近年の目覚ましい AI 技術の発展にこれらの学習アルゴリズムが寄与した部分は非常に大きなものである．以下に例を挙げると，Squeeze-and-Excitation ネットワークは [57] 画像認識の最良手法の一つであり，CycleGAN [58] もまたデータ生成の，Parallel WaveNet [59] は音声合成で最良の手法の一つであり，これらは高度な学習アルゴリズムに下支えされている．また，これらの成果は GPU などの高性能で高消費電力のデバイス，つまり強大なマシンパワーに依って得られた部分が非常に大きい．昨今の NN は膨大なモデルサイズということもあり，学習には多大な時間と電力が必要となる．これを解決するために AI 演算に特化したドメインスペシフィックアーキテクチャが登場している．その多くは推論処理をターゲットにしたものであるが，TPUv2 [60]，v3 [61]，および DLU [62] は従来の AI ハードウェアと異なり推論処理だけでなく，学習処理もターゲットにしている．このような AI ハードウェアでは効率の良い演算のために数値表現に工夫を凝らしていることが多い．TPU では NN の学習で実行される計算の大部分は浮動小数点乗算であるものの，DLU では “Deep Learning Integer”（DL-INT）と呼ばれる独自の精度を使用して

学習を行っている．このDL-INTを用いることで精度を維持しながら消費電力の削減を可能としている．ただし、これらのAIハードウェアはクラウド集約型のAIシステムにおいて増加する処理時間と消費電力に対処することを目的としていることからエッジデバイスまではサポートしきれていない．そのためにエッジデバイスを利用したオンライン学習で例に挙げたAI処理を行うことは実質不可能である．

一方で，ユーザ自身用途に合わせて回路を構成できるフィールドプログラマブルゲートアレイ（FPGA）を使用することで，GPUよりも高いエネルギー効率やAI処理の加速化を達成することができる．FPGAはユーザがプログラム可能なハードウェアであることから，画像認識に特化した畳み込みNNや時系列データの扱いに優れる再帰型NN等，各NNモデルに応じた高エネルギー効率の回路を構成可能である．そのため，FPGAを用いることで低消費電力化が可能であり，また，ローエンドからハイエンドまでさまざまなタイプのFPGAが存在している．ハイエンドFPGAはクラウドサーバーでも使用されることがあるが，本章で提案する学習手法は，ローエンド向け，またはローエンドFPGAよりも少ないリソースのデバイス，いわゆるエッジデバイスを対象としている．GPUよりも遥かに低い消費電力と演算リソースが求められるエッジデバイスでは，推論処理の演算を高速化，高効率化するような研究が多くなされている．このような研究では，前もってCPUまたはGPUを備えたサーバ上で学習されたNNのパラメータを使用している．こうしたFPGAベースのAIアーキテクチャ研究の概要は、Guoら[63]によってまとめられている．活発に行われている推論処理AIハードウェア研究とは対照的に，学習処理に焦点を当てたハードウェア研究事例は非常に少ない．学習処理を含むAIハードウェアは，推論処理に特化したAIハードウェアと異なり事前にサーバで学習したパラメータを使用せず，そのデバイス単独で学習することによって望ましいパラメータを獲得する．ただし，ハードウェアを見据えた研究例はいくつか存在し，DoReFa-Net [35]は，低ビット精度のパラメータや活性値，勾配を使用して，畳み込みNNを学習する手法である．この手法は，量子化または二値化によってNNの消費電力を削減、演算の加速化を目的としている．同様にF-CNN [64]は，実行時にストリーミングデータパスを再構成することにより，NNの消費電力を削減，演算の加速化を目的としている．ただし，F-CNNは32ビット浮動小数点演算を使用するため，この手法がエッジデバイスに適しているとは見なせない．

エッジデバイス上での学習に関する研究が不足している理由の一つとして学習の処理が推論の処理よりも煩雑であるためと考えられる．学習段階でのNNは非常に小さな値の演算を行うため，ネットワークは数値表現のためにより高いビット精度を必要

とする．ビット精度の増加は必要なリソースの増加も招く事になる．前章の結果からも，BP の量子化という従来手法を用いてエッジデバイス上での学習を行った場合，リソースと性能間のバランスが取れていない問題点が生じていた．本章では，これを解決する新規学習アルゴリズムを提案する．近い将来，学習可能なエッジ AI の重要性が高まると考えられる．なぜならば，AI 技術が企業だけでなく民間にも普及したとき，遍く全ての AI デバイスの学習をクラウドサーバ上で行うことは不可能であるために，全てのタスクにクラウドサーバを用いるのではなく，デバイス上で処理できることはデバイス上で処理を行うということの重要性が増すためである．

本章では，学習も可能なエッジ AI ハードウェア構築に向けて以下の提案を行う．

1. エッジ AI に向けた誤差逆伝播法を行うアーキテクチャと低電力および低リソースの学習アルゴリズムを提案する．アーキテクチャは SGD 法に基づいており，固定小数点乗算を使用して推論処理と学習処理の演算を行う．このアーキテクチャのビット精度は予備実験によって定義した，提案する学習アルゴリズムは，演算リソース削減のために三値化された勾配を使用して学習処理の演算を行う．

2. 提案学習アルゴリズム “Ternarized Backpropagation”（TBP）の説明を行う．低消費電力化，低演算リソース化に寄与する三つのアイディアを示す．同様に TBP アーキテクチャの説明を行う．提案するアーキテクチャは三つのモジュールから構成されており，個々の動作を示す．

3. 提案手法に関してアルゴリズムとアーキテクチャの観点から評価を行う．従来手法と比較して性能を維持しながらも各種演算リソースの削減に成功していることを示す．

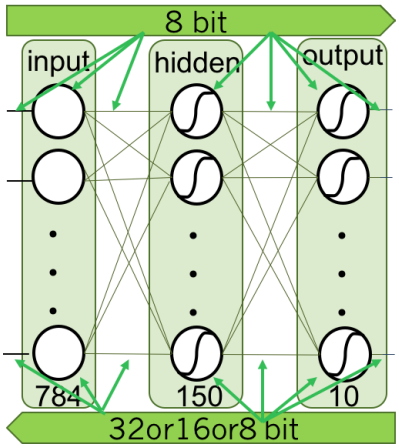

図 3.1　使用した NN モデル図．3 層の多層パーセプトロンを用いた． 中間層は一つであり，推論中のビット精度は 8 ビット，学習中では 32,16,8 ビットとした．

3.2　提案手法

本節では，予備実験の評価とそれを踏まえてエッジデバイスに向けた低消費電力・低演算リソースの学習ハードウェアについて述べる．予備実験は従来の学習アルゴリズムである BP と SGD を 16 ビットに量子化し (Fixed-BP)，推論処理と共に実装を行ったオンライン学習用アーキテクチャである．これを基にしてエッジ AI に向けた提案学習アルゴリズムである三値化バックプロパゲーション法とそのハードウェアアーキテクチャを構築した．まずは，以下の Fixed-BP について説明を行う．

3.2.1　16 bit 量子化 BP アーキテクチャ

機械翻訳，画像認識，データ生成など最も成功している AI システムの多くは，ソフトウェア上で処理されるアルゴリズムの工夫によってなされてきた．これらの AI システムでは，NN の学習に使用される学習アルゴリズムは，浮動小数点演算に基づいている．ここで，ハードウェア実装においては浮動小数点演算には固定小数点計算よりも多くの電力と演算リソースが必要となる．そのため，低電力かつ低演算リソースが要求されるエッジデバイス上で AI 処理を実現しようとした場合に浮動小数点演算の使用は適していないと言える．現在 NN の学習手法は種々存在しているものの，推論処理の研究に見られるようなそれらを効率よく実行するアーキテクチャの研究は少なく，エッジデバイス上での学習処理の実装について課題が多く残っている．

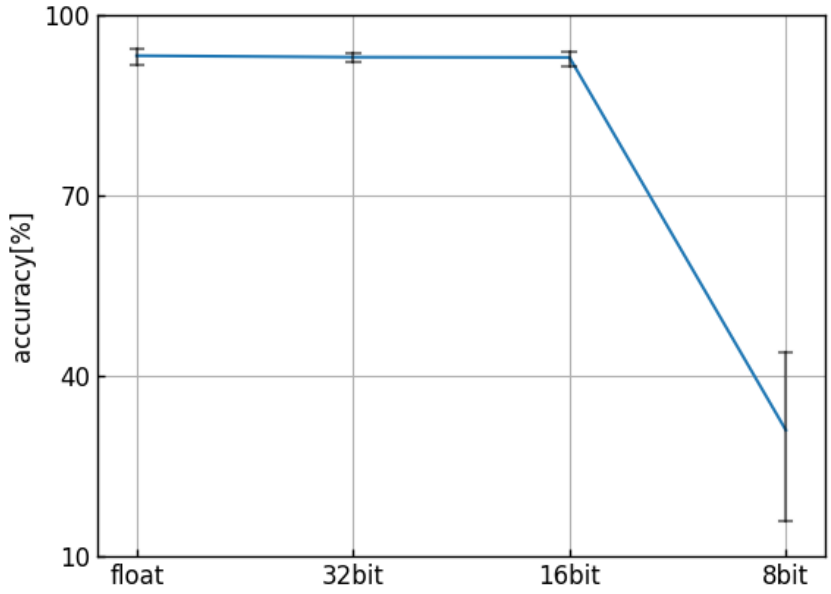

図 3.2 各ビット精度ごとでの認識精度．実線は 30 回試行した結果の平均を表している．

よって，エッジデバイス上で学習処理を実装する最も簡単な方法として挙げられるのは，最も単純な学習アルゴリズムである SGD 法を量子化し，それを効率よく実行するアーキテクチャの創出だと考えられる．量子化によって固定小数点化した計算を使用する際の最も重要な問題の 1 つはビット精度とモデルの性能 (認識精度等) 関係である．そのため，MNIST データセット [65] で MLP を使用したソフトウェアシミュレーションを用いてビット制限を課した際の認識精度の変化を観察することにより，最適なビット精度を探査した．図 3.1 に NN モデルを，図 3.2 にビット精度と認識精度の関係を示す．ニューロンの数は 784–150–10（このパラメータはターゲットハードウェアデバイスのメモリサイズ等によって制限される），推論段階のビット精度は 8 ビットであり内訳は 1–2–5 ビット（符号–整数–小数部）である．各層の活性化関数はシグモイド関数とし，学習率は 0.4，ミニバッチサイズは 1，学習回数は 0.3epoch とした（18,000 回）．メモリに保存する際には 2 のべき乗のビット精度が適していることから，各 2 のべき乗のビット精度について精度がどのように変化するかを確認した．この結果に基づき MNIST テストでより高い認識精度を達成するには，NN が学習処理で 16 ビットより高いビット精度を必要とすることが確認できた．従って，学習処理時のビット精度として 16 ビットを使用する．上記を踏まえて，推論処理と学習処理を単一デバイス上

で完結する AI ハードウェアの構築を行った．

ここで演算方式について述べると，大量のデータが用意され高速なスループットが求められるクラウドサーバによる AI 処理とは異なり，実空間での運用が前提となるエッジデバイスには並列演算等による高スループットの達成は必須ではないと考えられる．つまり，運用先の環境の変化をセンシングしデータの取得を行わなければならず，そこで現在のディジタルコンピュータに比べてこのデータの取得にかかる時間というのは非常に遅い．そのため，高性能プロセッサや高速インターフェイスを備えていないエッジ AI デバイスはアクセラレータとしての運用はそれほど求められていないと考えられる．例え並列性を高めて高速演算を達成した場合でも，低速インターフェイス（SPI、I^2C [66]）によってスループットが制限されることが予想される．従って，推論処理および学習処理では演算を時間方向へと展開し逐次的に行う方式となる時分割シリアル演算による低演算リソースでの処理を重視してアーキテクチャを設計した．

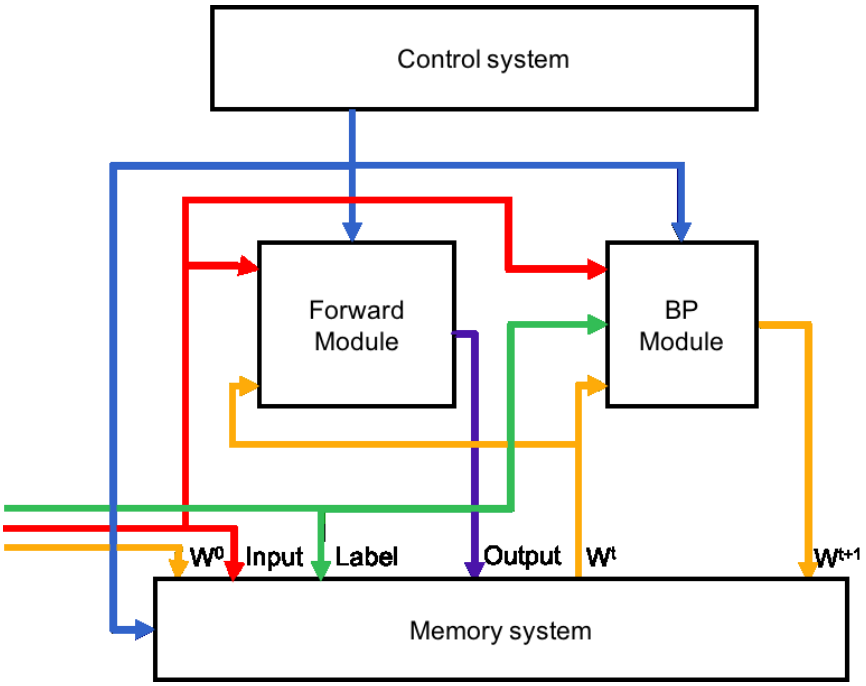

図 3.3　構成システム図. 本アーキテクチャは 4 つの要素から構成されている．Forward モジュールは推論を，BP モジュールは学習を行う．メモリには各種パラメータが保存されメモリアクセスはコントローラによって制御される．

図 3.3 に本ハードウェアアーキテクチャのブロック図とデータフローの概要を示す．本アーキテクチャは，推論モジュールと学習モジュールとが交互に動作する．推論処

理が終了し出力データがバッファからSRAMへと格納されると，学習処理が開始される．ここで，推論処理と学習処理共に計算は層毎に行われる．本アーキテクチャは，推論処理を行うForwardモジュールと学習処理を行うBPモジュール、各モジュールのステートやメモリのアクセスの制御を行うControlモジュール，そしてパラメータや入力，教師データ等を保存するメモリシステムの4つの要素に分割可能である．本アーキテクチャは動作開始時に初期化として，学習を行うNNに入力データ，教師データ，パラメータの値（W^0）を与える必要がある．認識精度等に当たる出力データは，推論処理終了後に外部メモリであるSRAMへと格納される．学習モジュールは、更新パラメータ（W^{t+1}）を出力する．学習前のパラメータに相当するW^tは，この新しいパラメータW^{t+1}によって上書きされる．推論モジュールと学習モジュールのメモリアクセスや演算は，コントローラによって制御される．

図3.4にアーキテクチャのより詳細なブロック図を示す．入力データと教師データはSRAMアクセス回数削減のために一時的にバッファ（x、t）に保存される．一方で，その数が膨大なものであるため重みやバイアスといったパラメータにはFPGA内部にバッファを用意せず，SRAMにのみ保存される．中間層と出力層の出力は、それぞれバッファ$g(h)$と$g(y)$に保存される．学習処理では，これらの各層の出力を用いて誤差の計算を行ったり，パラメータの更新を行う必要があるために学習処理が終了するまで各層の出力値は保存される必要がある．そのため，推論処理のみを行うAIハードウェアでは，この両バッファは必要ではない．さらに，活性化関数に入力する前の値である重み付き入力もまた学習処理に用いられることからバッファ(h、y)に保存する必要がある．コントローラモジュールは，メモリアドレスなどの信号を生成する．SRAMまたはバッファへのアクセスは，マルチプレクサ（MUX）を介して制御される．

推論モジュールと学習モジュールの詳細を以下に示す．推論モジュールおよび学習モジュールは，各種演算器を使い回す時分割シリアル演算方式を使用してハードウェアリソースを削減する．1つの演算サイクルで，SRAMから1つのパラメータの読み出しを行う．活性化関数とその導関数は．ルックアップテーブル（LUT）によって実装される．図3.5に推論モジュールの詳細を示す．本モジュールは，待機，中間層の演算，および出力層の演算の3つの状態で構成されている．推論処理は内積計算が必要となるが，これを乗算モジュール（mul）による乗算と加算モジュール（add）による累算によって実装を行った．乗算モジュールはSRAMに格納されているパラメータとバッファxまたはバッファ$g(h)$に格納されているデータを入力とし演算を行う．中間層の計算を行う場合，入力データの一つはバッファxに格納されている値となり，同様に、

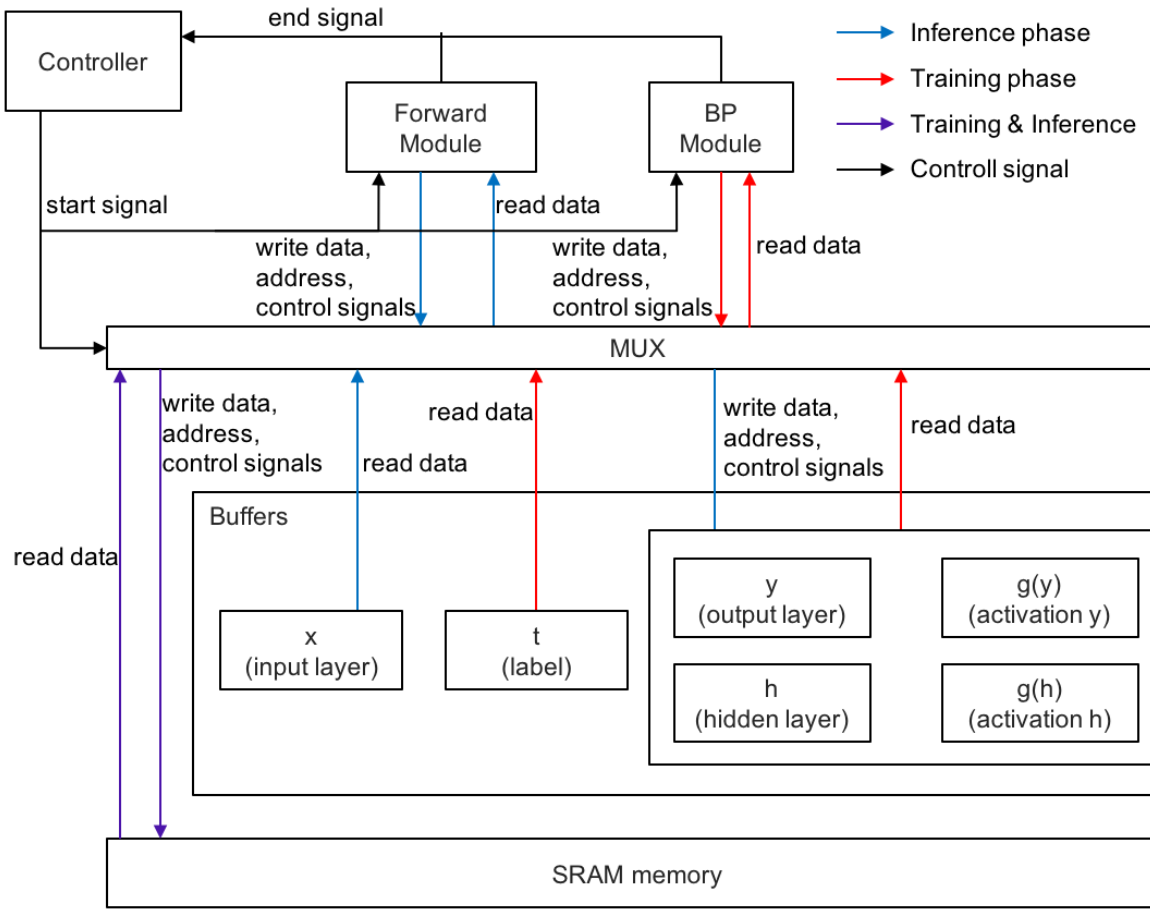

図 3.4　トップモジュール概要図. メモリアクセスはコントローラで制御される．読み出し，書き込みされるデータはマルチプレクサを通じて各モジュールへ入力される．推論処理中各層の出力はバッファへ保存される，出力バッファ $g(y)$ の値は推論処理終了後 SRAM へと書き戻される．学習処理が終わると更新されたパラメータはバッファを介さずに直接 SRAM へと書き戻される．

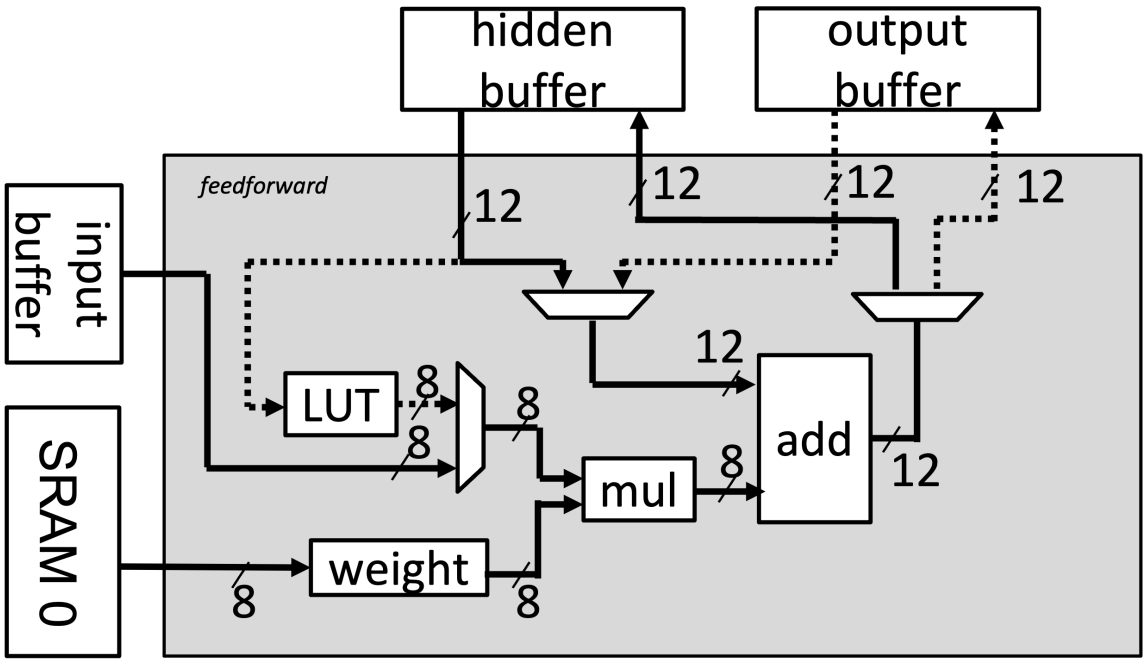

図 3.5　推論モジュールブロック図．計算結果はそれぞれのバッファへと格納される．バッファに格納される値は活性化関数に通す前の値であり，その後の演算に応じて活性化関数である LUT へと入力される．出力層の演算時には破線部の流れに従う．

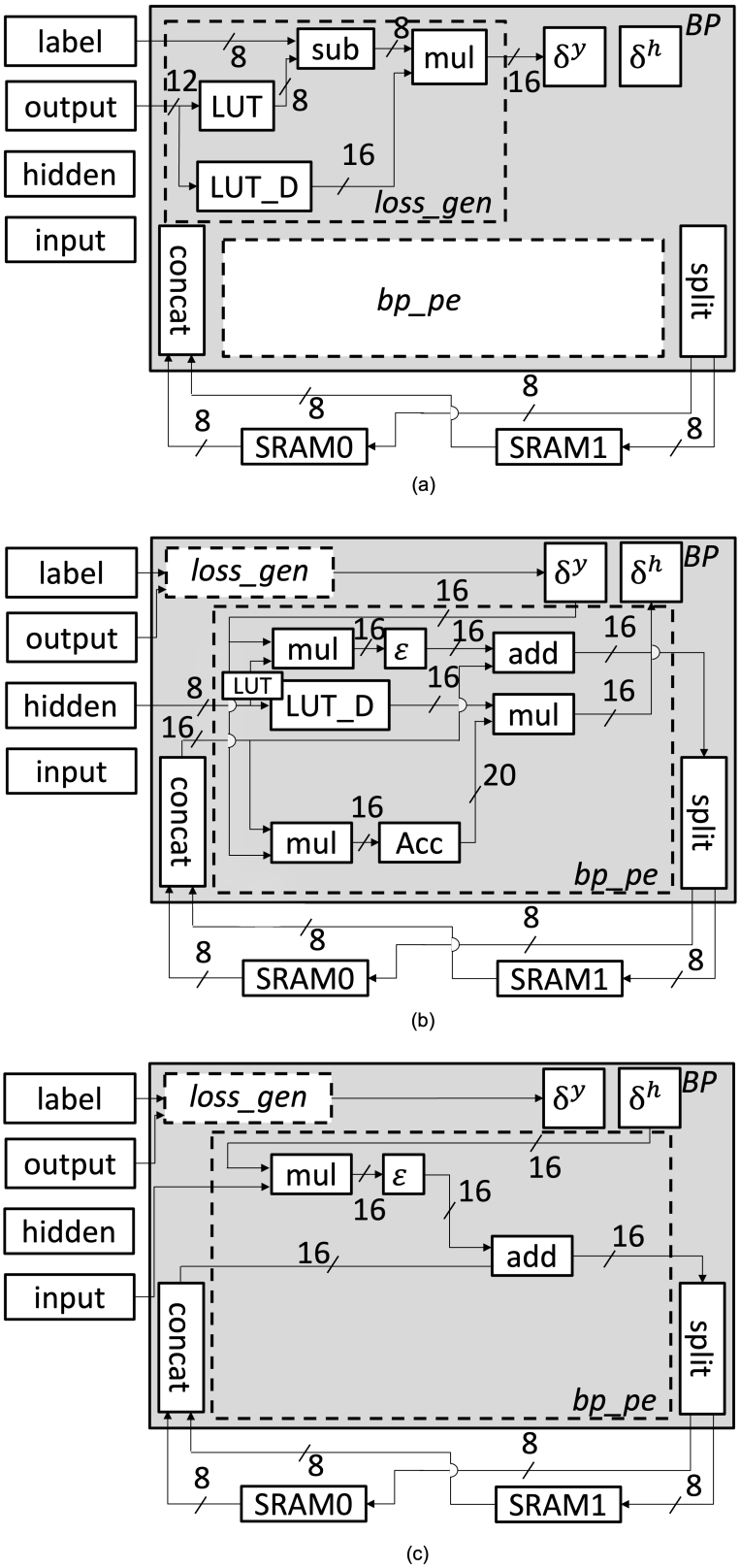

図 3.6　学習モジュールブロック図．(a) 出力層の誤差計算時の回路構成．(b) 中間層と出力層間のパラメータ更新と誤差伝播時の回路構成．(c) 入力層と中間層間のパラメータ更新時の回路構成．

出力層の計算を行う場合，入力データの一つはバッファ $g(h)$ に格納されている値となる．時分割シリアル演算のため出力層側の一つのニューロンを対象として計算を行い，終了次第次のニューロンを対象とする．一つのニューロンに対しての演算が完了すると，出力値はバッファ h または y に格納される．さらに、バッファ $g(h)$ または $g(y)$ へは LUT を介して出力値を保存する．全ての演算が完了した後、推論モジュールは学習モジュールの演算が完了するまで待機状態へと移行する．図 3.6 に学習モジュールの詳細を示す．本モジュールは，待機，出力層の誤差計算，中間層と出力層間のパラメータ更新，および入力層と中間層間のパラメータ更新という 4 つの状態で構成される．誤差 (δ^y) を計算する際，学習アルゴリズムの多くの場合において損失関数の導関数から減算形式で導出できる，誤差の計算には，label バッファ内の教師データと output バッファ内の NN 出力が必要となる．誤差計算が終了次第，更新値と前層へと伝播を行う誤差 (δ^h) の計算が可能となる．この更新値の計算には，対象としている層の誤差 (δ^y) とパラメータおよび hidden，input バッファに格納されている入力が必要となる．パラメータ更新の際に値が大きくなりすぎないよう学習率が設定されるが，この学習率は本アーキテクチャではビットシフトによって実装を行った．誤差計算に必要となる内積演算は，乗算（mul）および累積（accumulator）モジュールによって実装される．ここで時分割シリアル演算方式を採っている本アーキテクチャであるが，図 3.6(b) に示すように更新値と誤差伝播の演算はデータの再利用の観点から並行して行われる．また，逆伝播演算では入力層が最終層となるために誤差の伝播は中間層までとなる．つまり，図 3.6(c) に示すように入力層と中間層間のパラメータを更新する際には誤差伝播の演算を行う必要はない．

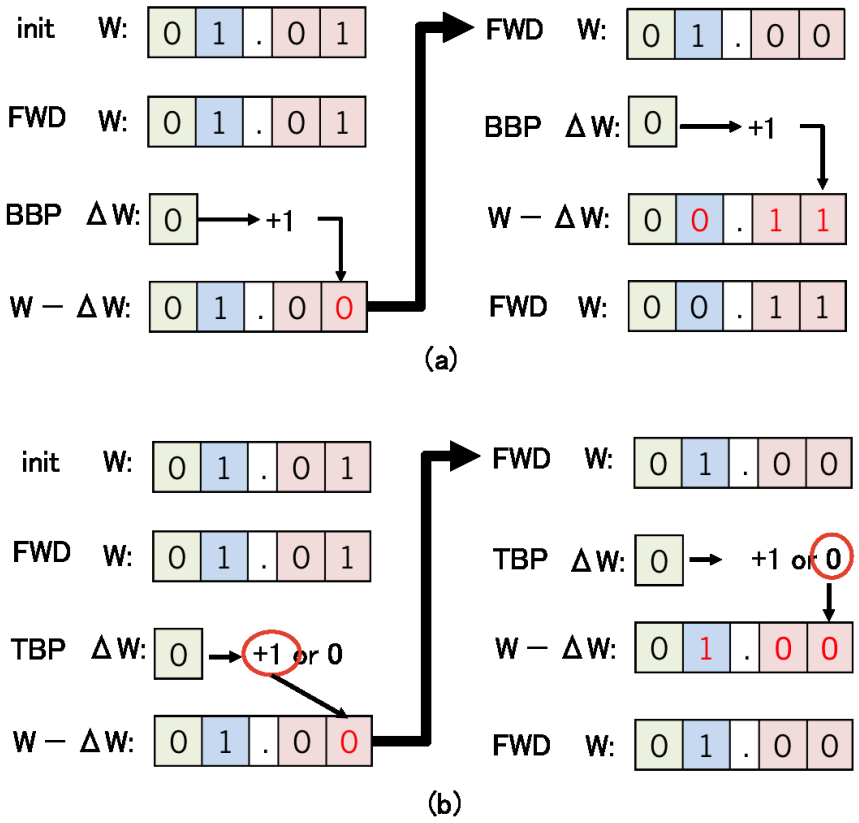

図 3.7　BBP，TBP アルゴリズムの概要図．両提案アルゴリズムは更新値 ΔW の値に応じて最下位ビットを +1，−1，0 (TBP) で更新する．(a) BBP の場合，全てのパラメータの LSB が更新される．(b) TBP の場合，一部のパラメータの LSB が更新される．更新されるパラメータは確率的に選択される．

3.2.2　Binary/Ternary backpropagation アルゴリズム

前節で明らかにしたように学習処理では，望ましい演算結果を得るためには推論処理よりも高いビット精度を必要とする．また，NN の学習に固定小数点方式を用いた場合，特に推論処理と学習処理とで異なるビット精度を持つ場合において NN のパラメータである重みやバイアスは学習によって必ずしも出力に影響を与えるとは限らない．これは推論処理と学習処理間でのビット精度の違いに起因している．たとえば，NN が推論処理で 4 ビット，学習処理で 8 ビットを必要とする場合，NN のパラメータは 8 ビットで保存される必要がある．このとき推論処理では，メモリに保存されている 8 ビットから上位 4 ビットのみをロードし演算を行う．もちろんそのためにはパラメータは

表 3.1 MNIST データセットでの学習結果．精度は 1epoch 学習した後のもの．

	BBP [%]	TBP [%]
#1	8.57	76.2
#2	2.59	77.0
#3	7.00	76.6
#4	8.21	76.0
#5	7.19	78.0

上位 4 ビットと下位 4 ビットで別々にメモリに保存されている必要がある．学習処理が終了しパラメータの更新が成された後，全てのパラメータが下位 4 ビットのみを更新されていた場合，次に推論処理を行ったとしても上位 4 ビットが変化していないことから学習前と同じ認識精度を出力することになる．推論と学習とで最適なビット精度が異なっているものの，ハードウェアコストを削減するために学習処理を 4 ビットした場合では NN は上手く学習できず低い性能となってしまう．SGD，Adadelta[67]，RMSProp[68]，または Adam などを利用するソフトウェア実装において AI 技術を牽引してきた学習アルゴリズムを単純に量子化した手法というものは，推論処理に比べるとビット精度が高くなることからエッジデバイスには適していないと言えるだろう．したがって，この問題を解決するにはソフトウェアの知見だけではなくハードウェアの知見に基づき，協調設計により生み出される新しい学習アルゴリズムが必要となる．

図 3.7 に本節で提案する学習アルゴリズムの概要を示す．これは，二値化バックプロパゲーション法 (BBP) および三値化バックプロパゲーション法 (TBP) と名付けたものである．両アルゴリズム共にアイディアの基礎は，更新値 (ΔW) の符号ビットに従って最下位ビット (LSB) の更新を行うというものである．BBP および TBP では，それぞれ二値または三値で学習処理の演算が行われている．したがって，乗算器の代わりに XNOR ゲートまたはマルチプレクサを使用できるため，学習処理での演算コストの削減が可能である．さらに，活性化関数の勾配次第ではあるが，活性化関数の導関数を求める演算を省略可能である．表 3.1 に BBP および TBP を用いた MNIST 分類の精度を示す．NN モデルは、入力層に 784 個，中間層に 128 個，出力層に 10 個のニューロンを持つ MLP である．また，各層はバイアス項を持ち，ミニバッチサイズ 64 で学習回数は 1epoch とした．活性化関数として中間層に LeakyReLU[69]，出力層にシグモイド関数を用いている．BBP の認識精度が低いことを確認できるが，これはすべてのパ

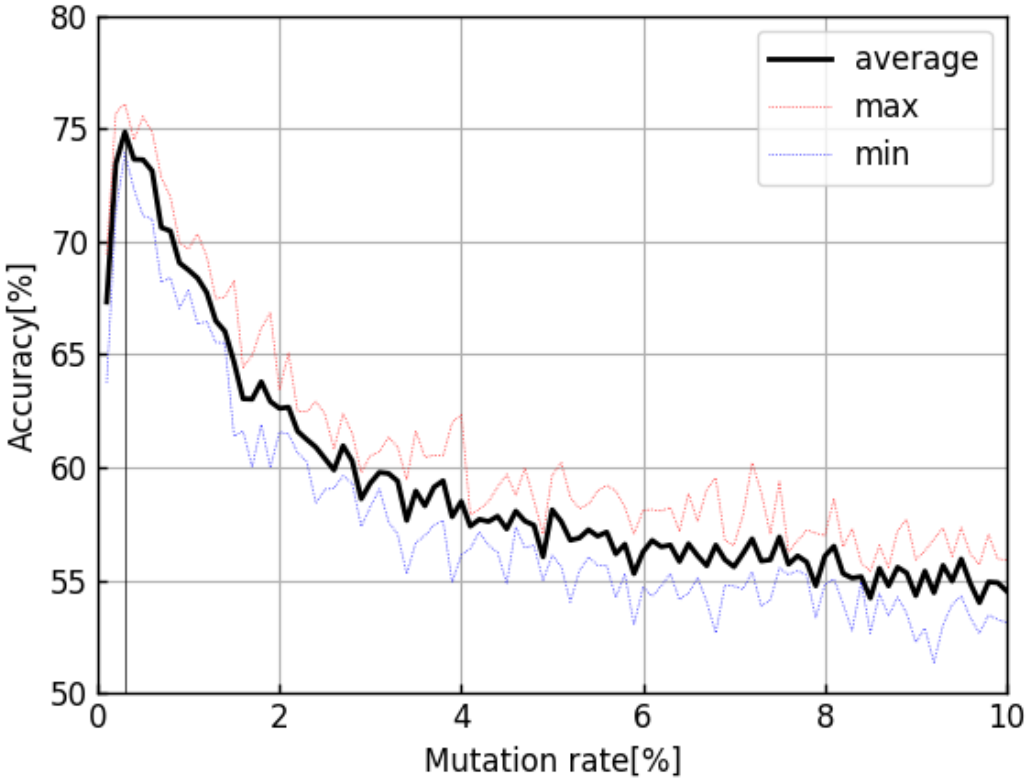

図 3.8 Mutation の比率と精度の関係．図中黒色の線は 5 回試行した平均値を示す．同様に赤色は最大値を，青色は最小値を示す．Mutation の比率は 0.3%の時最適ということが判明した．つまり，全てのパラメータの内 0.3%を更新した時に最大の精度が得られた．

ラメータを更新するため学習率が非常に高くなるということに起因していると考えられる．そのため．学習率を下げる必要が生じるが最下位ビットを更新している関係から単純に値を小さくすることは不可能である．そのため BBP では +1 または −1 で必ず更新されていたパラメータに，TBP では 0 を加えることで見かけの学習率を削減することで精度の著しい低下を抑制することに成功した．

Mutation

TBP では見かけの学習率を下げるために 0 で更新を行うが，そのパラメータの選択を行う操作を Mutation と定義する．ここで，Mutation の比率はハイパーパラメータとなる．Mutation が 0% の場合，パラメータは更新されないため，認識精度は NN の初期

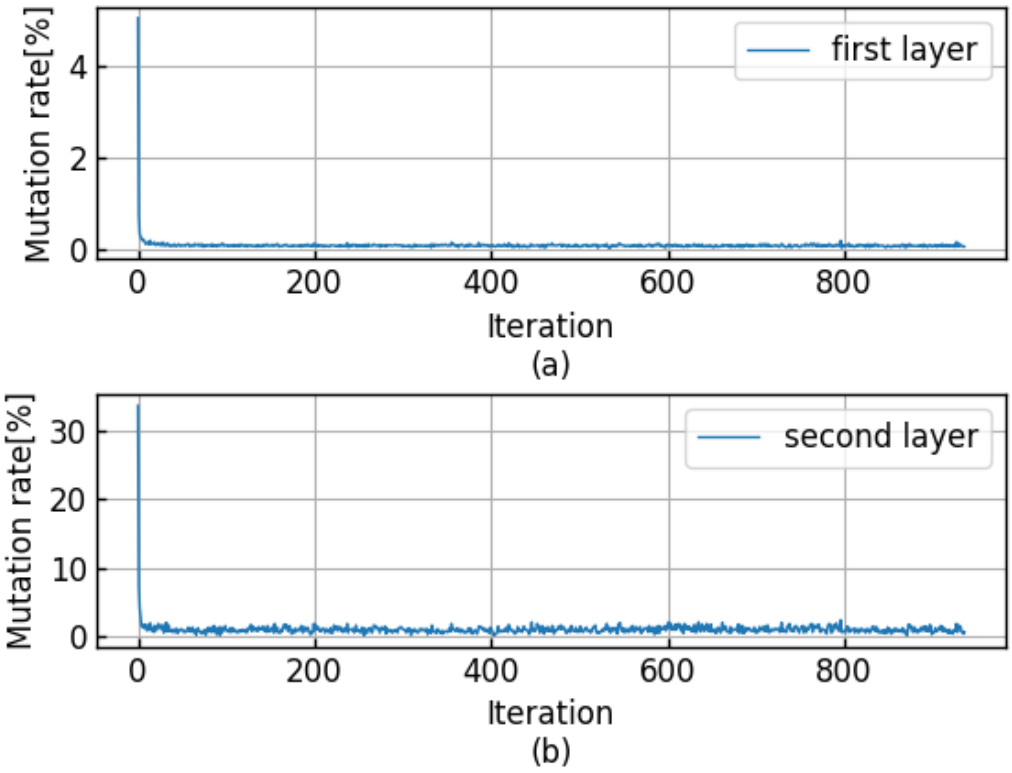

図 3.9　16 ビットに量子化した SGD 手法を用いて 1 epoch 学習した後のパラメータ変化率．ここでの変化率とはパラメータの上位 8 ビットの値が学習前後で変化した割合と定義している．(a) 入力層と中間層の間のパラメータ変化率．(b) 中間層と出力層の間のパラメータ変化率．

パラメータによる認識精度と同じスコアになる．同様に，Mutationが100%の場合，認識精度はBBPと類似したスコアになる．つまり．ΔWが0になることはほぼないため，Mutationが100%の場合TBPとBBPは等価となる．厳密に言うならば，TBPでは誤差の逆伝播演算に0の値を用いることからBBPとはほぼ等価である．一般に，消費電力の観点からは低いMutationの比率の方が優れている．パラメータはメモリに保存されているため更新を行う際にはメモリアクセスの必要がある．この時Mutationの比率が低いと更新を行うパラメータ数が少なくなるためメモリアクセス回数も少なく済むためである．図3.8は，Mutation率が0.3%の場合にNNが最高の認識精度を得たことを示している．また，この結果からMutation率が高いほど認識精度が低いことが確認できる．図3.9に従来手法として比較対象に用いるFixed-BP (16ビット固定小数点化SGD法)のパラメータ上位8ビットの変化率を示す．ここでの変化率とはパラメータの上位8ビットの値が1回の学習前後で変化した割合と定義した．この変化率はMutationによる演算と同様なものであると考えることができる．Fixed-BPにおいて変化率は学習最初期を除き一定の割合となることから，層と学習回数をパラメータに持つと考えられる．対照的に，現在提案しているTBPのMutationは，層と学習回数によってパラメータ化されておらず，さらに，TBPのMutation率は非常に低いためNNが十分に学習されていない可能性がある．ただし，TBPをFixed-BPで学習した際に得られた変化率を基にしてMutation率を設定したとしても，認識精度は高くなるとは限らない．入力層と中間層間のパラメータ更新に係るMutation率と中間層と出力層間のパラメータ更新に係るMutation率とそれぞれの最適なMutation率を探索することは非常に困難であるため，本項，また次の項では，Mutation率は経験的に定義し一定のものとした．従って，Mutationの方法等の再検討が今後の課題として残っている．

更新手法

学習処理は誤差や勾配の計算とパラメータの更新に分割できる．提案手法であるTBPは誤差や勾配の計算プロセスに関連し，Mutationはパラメータの更新プロセスに関連している．TBPはBPに基づいているため，演算プロセスは基本的に同じである．ただし，演算に用いられるすべての値は量子化または三値化されている．Mutationには3値化が含まれており，ΔWは+1，0，または–1となる．更新処理の際には，この三値化されたΔWが各パラメータのLSBに加算される．図3.10に，更新を行うパラメータ選択に乱数を使用する手法，更新の重みを選択するためにカウンタを使用する手法の2種類の更新手法を示す．ここで，乱数を用いる手法では一様乱数を使用する．カウンタ

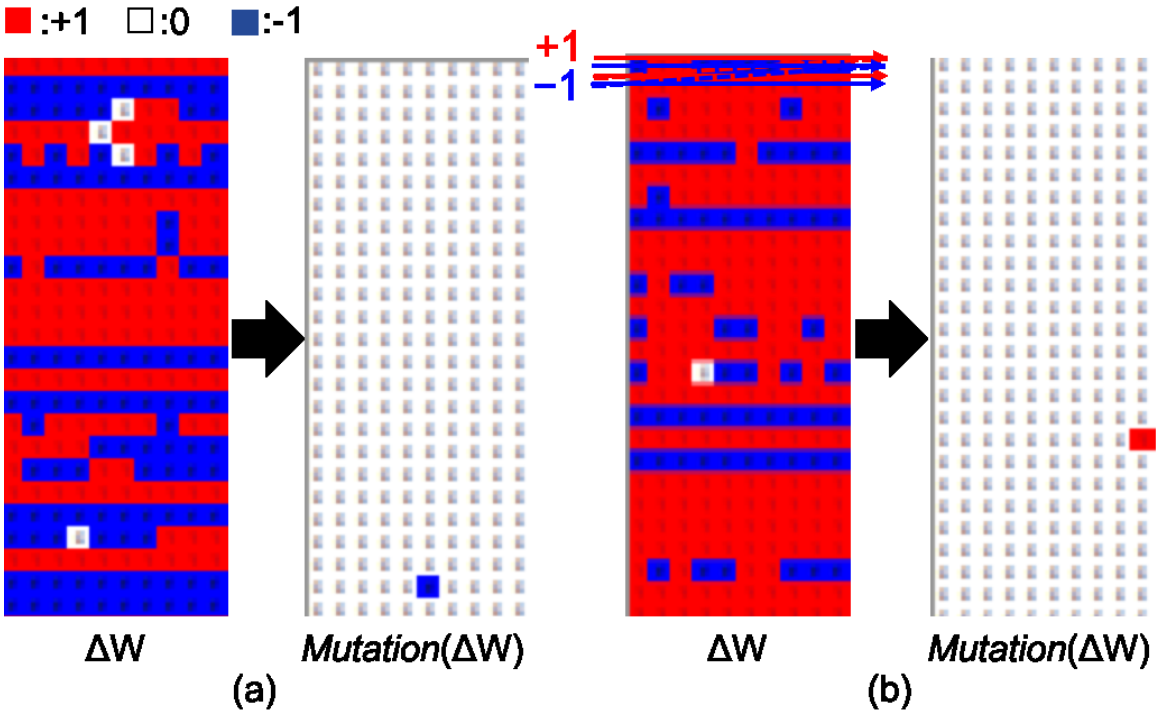

図 3.10 TBP の更新規則．両規則とも更新されるパラメータは一様に選択される．(a) と (b) の左側は算出された ΔW の値を示している．同様に (a) と (b) の右側は Mutation の結果を示している．この時赤もしくは青色の部分のパラメータが更新されるが，赤色は +1 を，青色は −1 の値を，白色は 0 を示している．(a) 更新パラメータの選択に乱数を用いる方式． (b) 更新パラメータの選択にカウンタを用いる方式．正の値，負の値それぞれをカウントするカウンタを用意し，あらかじめ設定された閾値を超えた時のパラメータを更新する．

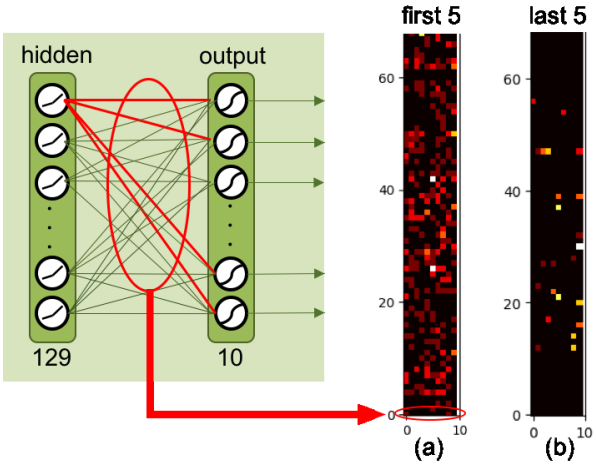

図 3.11　NN モデルの一部（図左）と学習回数と更新されたパラメータの関係（図右 (a)(b)）．NN は全結合型のもので各ニューロンは次の層の全てのニューロンに結合している．赤丸で示すように図左と図右 (a)(b) は対応しており，(a)（b）の縦軸が入力層のニューロンを横軸が出力層のニューロンを示している．同一箇所の更新回数が増えるにつれ赤色から黄色へと変化している．(a) は学習初期から 5 回までを，(b) は学習終わりまでの 5 回をグラフ化したものである．(b) から更新されるべきパラメータが存在することが示唆されている．

を用いる手法ではモジュール内に +1 および –1 を担当するカウンタを備えており，カウントアップによりハイパーパラメータとなる閾値を越えた箇所のパラメータを更新する．内部のカウンタは学習終了によって初期化を行わないため，最初にカウントされるパラメータの更新も保証されている．上記二種類の更新手法を用いて，同一の NN モデルと Mutation 率で学習を行ったが，認識精度に大きな差は見受けられなかった．図 3.11 に更新されるパラメータと学習回数の関係を示す．図中左には NN モデルの一部を，図中右 (a)(b) には学習回数と更新されたパラメータの関係を示している．NN は全結合型のもので各ニューロンは次の層の全てのニューロンに結合している．赤丸で示すように図左と図右 (a)(b) は対応しており，(a)(b) の縦軸が入力層のニューロンを，横軸が出力層のニューロンを示している．同一箇所の更新回数が増えるにつれ図中右のグラフが赤色から黄色へと変化している．(a) は学習初期から 5 回までを，(b) は学習終わりまでの 5 回をグラフ化したものである．(a) の結果から学習開始から 5 回目まで，パラメータ全体の約 80%が少なくとも 1 回更新される．次に，パラメータの一部は数回更新されることも確認できる．続いて (b) の結果から学習最後までの 5 回で，いくつかのパラメータが更新される．更新されたパラメータの総数は (a) より少ないが，

表 3.2 MLP を TBP で学習した際の L2 正則化を使った場合 (w/) と使わなかった場合 (w/o) での認識精度，合計 5 回の試行回数である．

	w/o L2 regularization					w/ L2 regularization				
epoch↓	#1	#2	#3	#4	#5	#1	#2	#3	#4	#5
1	76.2	77.0	76.6	76.0	78.0	75.2	74.1	75.0	75.0	75.3
2	73.9	74.7	72.4	74.1	74.3	76.0	75.7	74.1	75.1	75.4
3	71.5	72.2	70.1	71.1	73.5	74.1	75.3	74.0	74.3	75.5
4	68.7	69.2	69.2	70.2	70.3	73.5	75.0	73.2	73.9	74.7
5	67.9	66.7	67.4	68.6	68.6	72.9	74.7	74.6	72.2	73.0

同一パラメータが更新された割合は数倍となっている．以上の結果に基づいて考察を行うと，パラメータ更新は一様に行われずに更新されるべきパラメータが存在していることが示唆されている．ただし，本章で提案する更新手法は乱数を用いる手法とカウンタを用いる手法と共にパラメータの更新を一様に行う．そのため，演算コストの増加を抑えつつ更新されるべきパラメータを更新するような手法の考案が課題として残っている．

L2 正則化

Fixed-BP を用いて学習した場合とは対照的に，TBP を用いて学習を行った NN は，エポック数の増加とともに認識精度の低下が確認できた．表 3.2 に MNIST データセットの学習に TBP と L2 正則化を用いた時と用いない時での認識精度を示す．L2 正則化を用いない場合最初のエポック後の認識精度は約 76%であるが，学習が進み 5 エポック後には認識精度が約 68%にまで低下していることが確認できた．一般に，学習回数が多いほど NN の認識精度は高くなるが，長時間の学習によって過学習 [70] が引き起こされた場合は学習を続けるにつれ認識精度の低下が生じる．ただし，学習の初期段階では過学習は問題とならないことから，これは TBP の更新方法に起因している可能性があると考えられる．そこで学習によってパラメータのヒストグラムがどのように変化するのかを確認した．

図 3.12 に各層間のパラメータのヒストグラムを示す．5 エポック後にはパラメータの分散が大きくなっていることが確認できる．入力層と中間層間のパラメータは，+0.5 と −0.5 の間に分布している．また，中間層と出力層間のパラメータは，主に +1.0 と

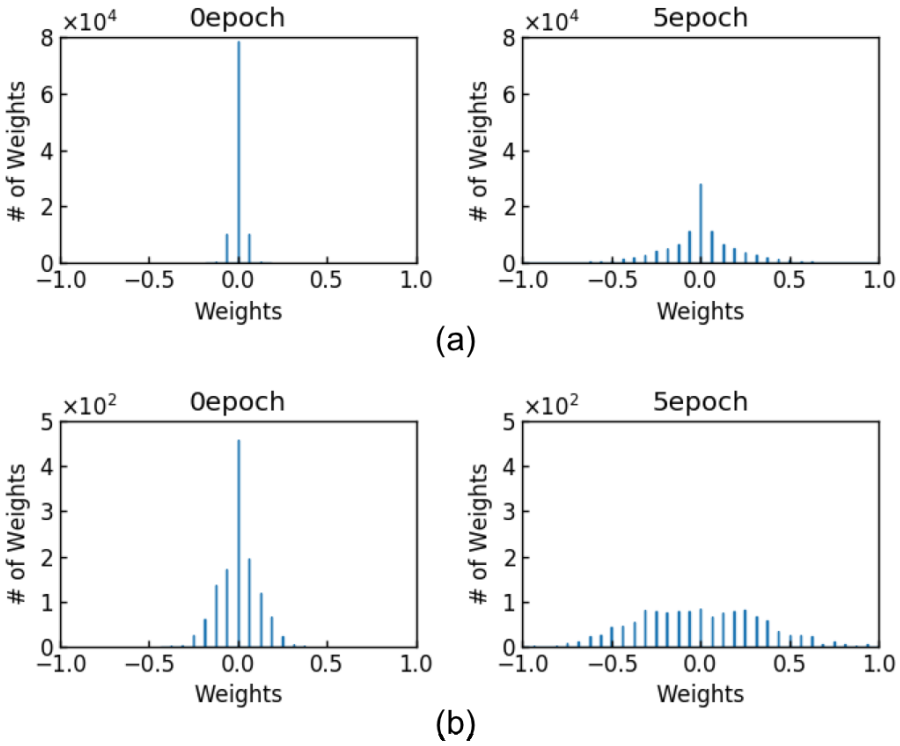

図 3.12 L2 正則化を用いなかった場合のパラメータのヒストグラム. (a) 入力層と中間層間のパラメータ. (b) 中間層と出力層間のパラメータ.

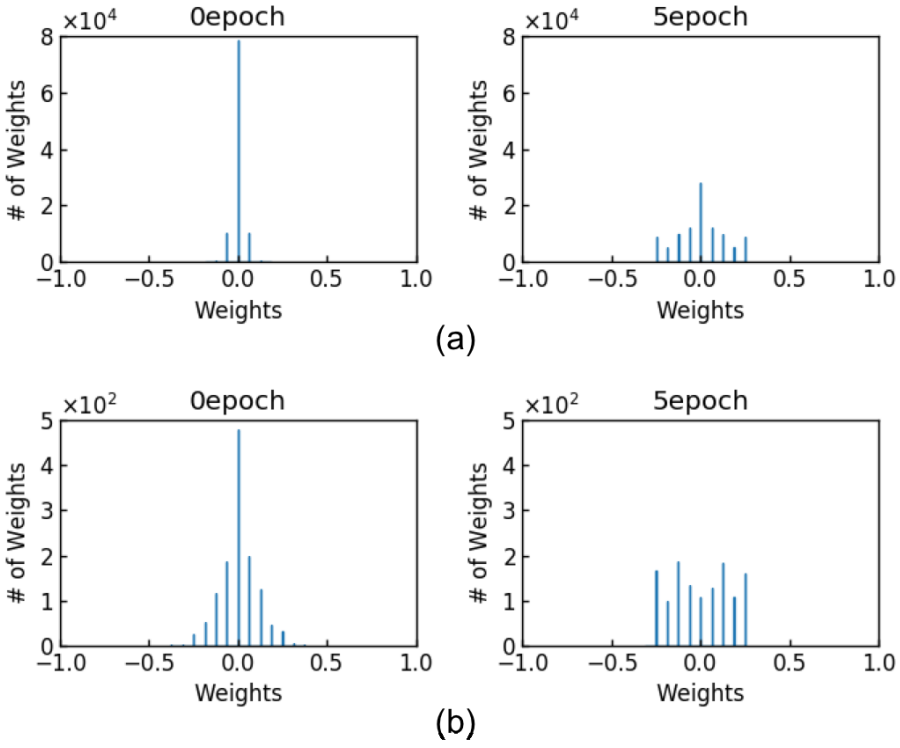

図 3.13 L2 正則化を用いた場合のパラメータのヒストグラム. (a) 入力層と中間層間のパラメータ. (b) 中間層と出力層間のパラメータ.

表 3.3 誤差のビット精度を変更した時の認識精度

	3bit	4bit	5bit	6bit
epoch: 1	77.01%	82.46%	82.84%	84.23%
epoch: 2	78.49%	81.73%	83.14%	83.29%
epoch: 3	77.91%	81.97%	82.71%	83.71%
epoch :4	78.16%	81.78%	81.56%	82.00%
epoch: 5	76.88%	83.07%	82.08%	80.57%

–1.0 の間に分布している．以上のことから，認識精度の低下を引き起こす原因として，パラメータの分散が大きくなっていることが一つ挙げられる．したがって，パラメータの分散の増加を防ぐために，L2 正則化を導入した．L2 正則化の結果を図 3.13 に示す．L2 正則化を導入することで認識精度は最初のエポックの後は約 75%，5 エポックの後は約 73%となることが確認できた．精度の低下と分散の増加を抑えることに対して一定数の効果を上げたものの不完全であると考えられる．

以上の結果に基づいて，学習に用いる値の全てを三値化した TBP では，高精度の性能を達成することが厳しいことが判明した．したがって学習に用いる全ての値を三値化するのではなく誤差のみを多ビットで持つことにより保持する情報量を増加させることとした．表 3.3 に誤差を多ビットで使用した時の認識精度の推移を示す．結果から，誤差に 4 ビット以上を使用した時認識精度の向上が確認できた．この誤差のみを多ビットで保持する TBP 手法では，誤差の厳密な値ではなくニューロン間の誤差の大小関係が保存されているため，ハードウェアリソースの増加を抑えることができている．そのため要求リソースと性能のバランスが取れていると判断した．実際には，誤差の量子化は 1 ビット右へビットシフトした値として定義している．これにより，性能とリソース要件のバランスの観点からも妥当な認識精度を達成可能である．

Algorithm 1 Ternarized BackPropagation (TBP). E は誤差関数を，y は各層の出力を示す．ternarize(X) は三値化の操作を行い，mutation(X) は更新するパラメータの選択を行う．L はパラメータの層の数を示し，θ はパラメータを，λ は L2 正則化の正則化項であることを示す．

Require

Network parameters $W, b \in \theta$

Require

Input data x at each layer

Require

bitwidth n of fractional part in inference phase.

$\delta^L = \frac{\partial E}{\partial y^L}$

for $l = L$ to 1

$\delta_q^l = quantize(\delta^l \gg 1)$

if $l \neq 1$ **then**

$\delta^{l-1} = ternarize(W^l) \cdot \delta_q^l$

$\Delta\theta = ternarize(x) \cdot \delta_q^l$

$\Delta\theta_t = ternarize(\Delta\theta)$

$\theta_{t+1}^l = \theta_t^l - Mutation(\Delta\theta_t) \gg n - \lambda \times Mutation(\theta_t^l)$

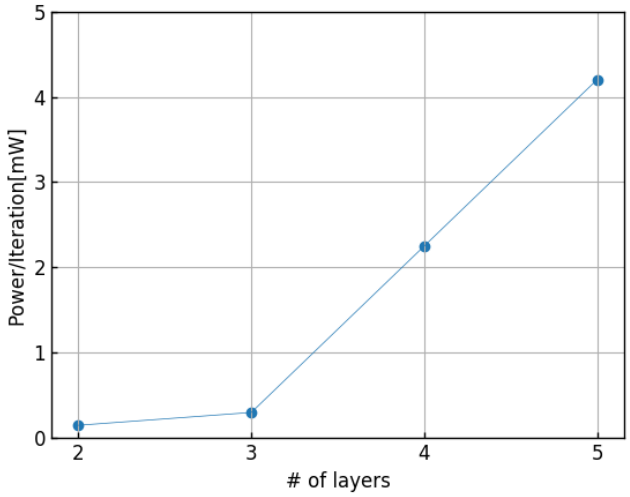

図 3.14　層の数と消費電力の関係．NN の層の数が 4 以上になった場合，誤差の逆伝播の関係から消費電力が増大する．

アルゴリズム

Algorithm 1 に TBP の擬似コードを示す．本提案手法は BP および SGD に基づいている．ここで，L はパラメータの層の数である，たとえば，$L = 2$ の場合，MLP は入力層，中間層および出力層の 3 層で構成される．誤差のみは多ビットで計算が行われることから $\delta^l_q = quantize(\delta^l \gg 1)$ では，誤差を適当なビット精度への量子化を行っている．この方程式中にある値 1 はハイパーパラメータであり，この値が大きいほど誤差の保存や計算に必要となるビット精度が高くなる．また，パラメータ更新を行うターゲットが入力層と中間層間のものであることを示す $l = 1$ の場合，誤差をさらに前の層へと伝播する必要は生じない．誤差のみを多ビットで表現している場合，内積に係る乗算は乗算器を利用するのではなくマルチプレクサによる代用で実現可能である．パラメータの更新を行う式である $Mutation(\Delta\theta_t) \gg n - \lambda \times Mutation(\theta^l_t)$ は，$\Delta\theta_t$ と θ^l_t の間の Mutation は独立していないため，L2 正規化は全てのパラメータに適用されるのではなく θ_t の更新箇所のみに適用される．つまり Mutation の結果，一部の $\Delta\theta_t$ が 0 となった場合，そのパラメータには L2 正規化は適用されない．ここで，λ はハイパーパラメータであり，MNIST データセットでの実験に従って，このハイパーパラメーターを 0.5 として定義した，

図 3.14 に層の数を変更することによる消費電力の推移を示す．なお，消費電力は SRAM アクセスへの回数から見積もりを行っている．NN が中間層を二つ以上持つ場合，前層へ誤差を逆伝播する際にパラメータを読み出し計算する必要があることから消費電力が増大してしまう．以上のことから，エッジデバイス上でのオンライン学習には入力層，中間層，出力層からなる構成が最適であると考えられる．しかし，NN が 4 つ以上の層により構成されている場合でも，更新するパラメータの選択次第では全てのパラメータを読み出す必要がなくなることから消費電力の削減を期待できる．

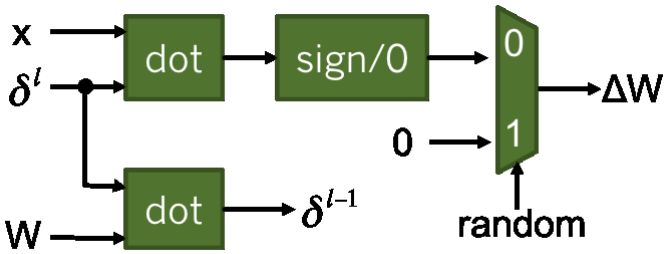

図 3.15　乱数型の TBP アーキテクチャのブロック図.

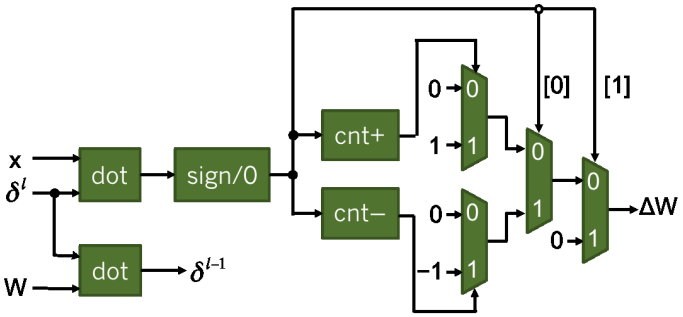

図 3.16　カウンタ型の TBP アーキテクチャのブロック図.

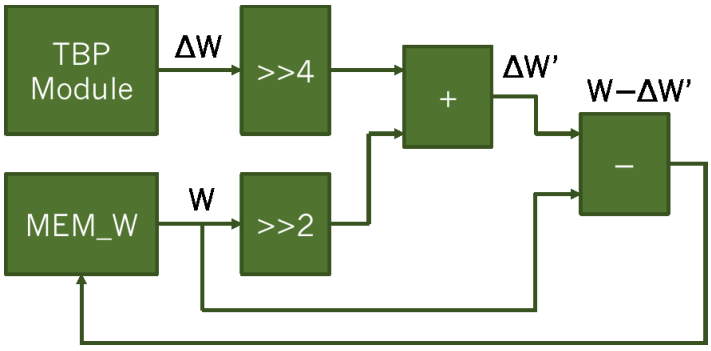

図 3.17　更新モジュールのアーキテクチャのブロック図.

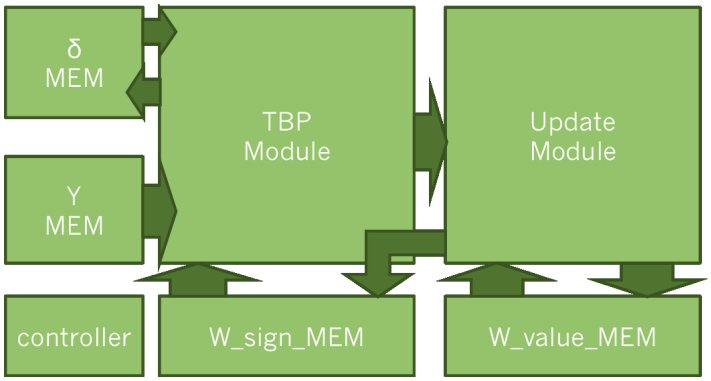

図 3.18　学習モジュール全体のアーキテクチャのブロック図

TBP アーキテクチャ青写真

図 3.15 に乱数を用いる手法の TBP モジュールのアーキテクチャブロック図を示す．本モジュールは内積モジュール（dot），三値化モジュール（Sign/0），およびマルチプレクサで構成される．ここで，誤差 δ^l とターゲットとしている層への入力 x は，更新値 ΔW を計算するために内積モジュールへと入力される．計算された ΔW は，乱数によって確率的に更新モジュールへと入力される．TBP は，活性化関数にシグモイドや Leaky ReLU 等の正や負に一様な傾きを持つ場合に活性化関数の導関数を求める計算を省略可能である．また，同様に誤差 δ^l とパラメータ W は，前の層へ誤差 δ^{l-1} を計算するために内積モジュールにも入力される．この操作は，ΔW と同時に行う必要がないことから，時分割演算での実行が可能である．

図 3.16 に，カウンタを用いる手法の TBP モジュールのアーキテクチャブロック図を示す．本モジュールも同様の構成を持つが，乱数を用いる手法よりも多くのマルチプレクサを備えている，これは，+1（cnt+）と −1（cnt-）の二種類のカウンタを用意したことに起因する．カウンタを用いる手法の TBP モジュールも，内積と三値化のプロセスは上述した乱数を用いる手法の TBP モジュールと同様である．三値化モジュールの後の最初のビット “[0]” は +1 または −1 を意味する符号ビットであり，2 番目のビット “[1]” は 0 か 0 ではないかを示すゼロフラグビットであることを意味する．つまり，+1 は，三値化モジュールの出力では 00 であることを意味する．各カウンタモジュール (cnt+，cnt-) でカウントが予め設定されたしきい値を超えている場合，カウンタモジュールはマルチプレクサを介して +1 または −1 を ΔW として出力する．

図 3.17 に更新モジュールのアーキテクチャブロック図を示す．外部メモリとなる SRAM メモリ（MEM_W）からパラメータを読み取る回数は増加しないために少ないオーバヘッドで L2 正規化の実装が可能である．ここで，“≫4” 中の 4 は，推論処理での小数部のビット精度を，同様に “≫2” 中の 2 は，L2 正則化の正則化項を意味するハイパーパラメータである．更新されるパラメータは全体の 1%未満であることから，メモリアクセスに係る消費電力の大幅な削減を見込める．

図 3.18 に学習モジュール全体のアーキテクチャブロック図を示す．パラメータは演算コストを削減できるように，符号部分（W_sign_MEM）と値部分（W_value_MEM）を分割し保存することを提案した．誤差伝播の計算や更新値 ΔW を求める計算では，パラメータの符号部分のみを使用することで高効率な演算を可能とする．誤差は δ-MEM に保存され，中間層の出力は Y-MEM に保存される．このメモリのアドレスは，内積計算や更新パラメータの状況に応じてコントローラモジュールによって制御される．

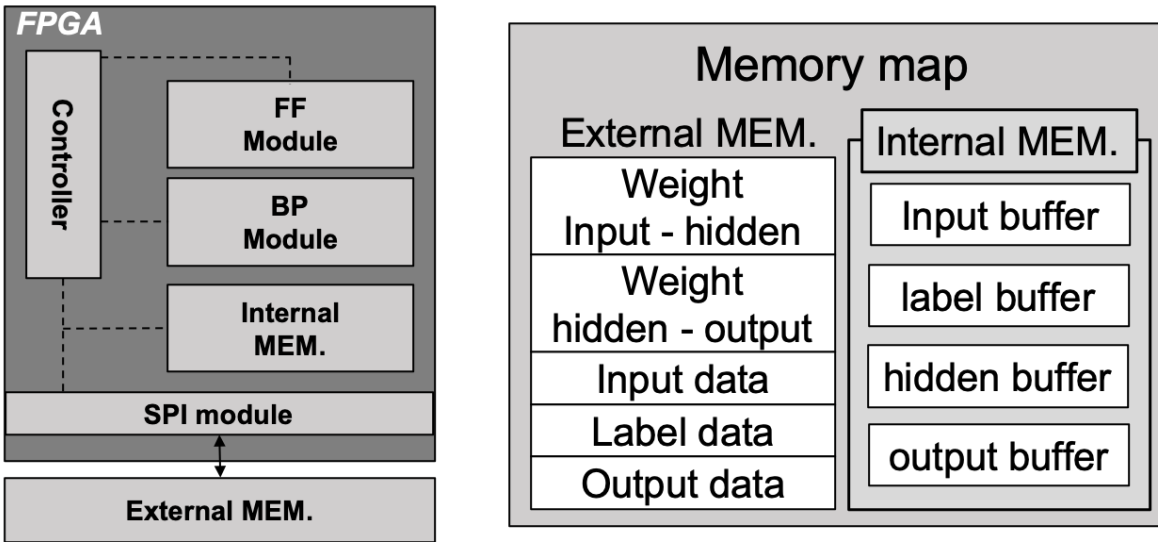

図 3.19 アーキテクチャ全体の概要図と各メモリのメモリマップ．エッジ上に AI を実装する場合には AI 演算を行うコアの側に外部メモリを置く方式が適している．外部メモリは NN のパラメータや入出力，教師データの保存に使用される．内部メモリはバッファとしての使用が主たる役割となる．

3.2.3 TBP アーキテクチャ

本節では，前節で提案した TBP アルゴリズムを FPGA へと実装した，その内容について説明する．本節ではエッジ AI へ焦点を当てた実装を行った．一般に豊富な電力や演算リソースを持ち，大量のデータを用いた高速処理が求められるクラウド AI では並列処理による高速化が図られている．しかし，エッジ AI では演算リソースに制限があることに加えて，AI 演算コアとデータのサンプリングを行うセンシングデバイスとの通信がボトルネックになり高速処理を行ったとしてもその大部分が待機時間となることが予想される．これら二つの点から，提案アーキテクチャでは並列処理ではなく時分割にシリアル演算を行うという方式で演算を行っている．つまり，一回の演算で一つのパラメータを読み出し演算器をシェアする方式である．

図 3.19 にアーキテクチャ全体の概要図と内部メモリと外部メモリにおけるメモリマップを示す，FPGA 内部に推論モジュール (FF Module) と学習モジュール (BP Module)，内部メモリ (Internal MEM.) と外部メモリ (External MEM.) 等との通信用モジュール (SPI Module) を持ちそれらの制御を行うコントローラ (Controller) を備えたアーキテクチャがエッジ AI には適していると考えられる．内部メモリとしては Block RAM やレジス

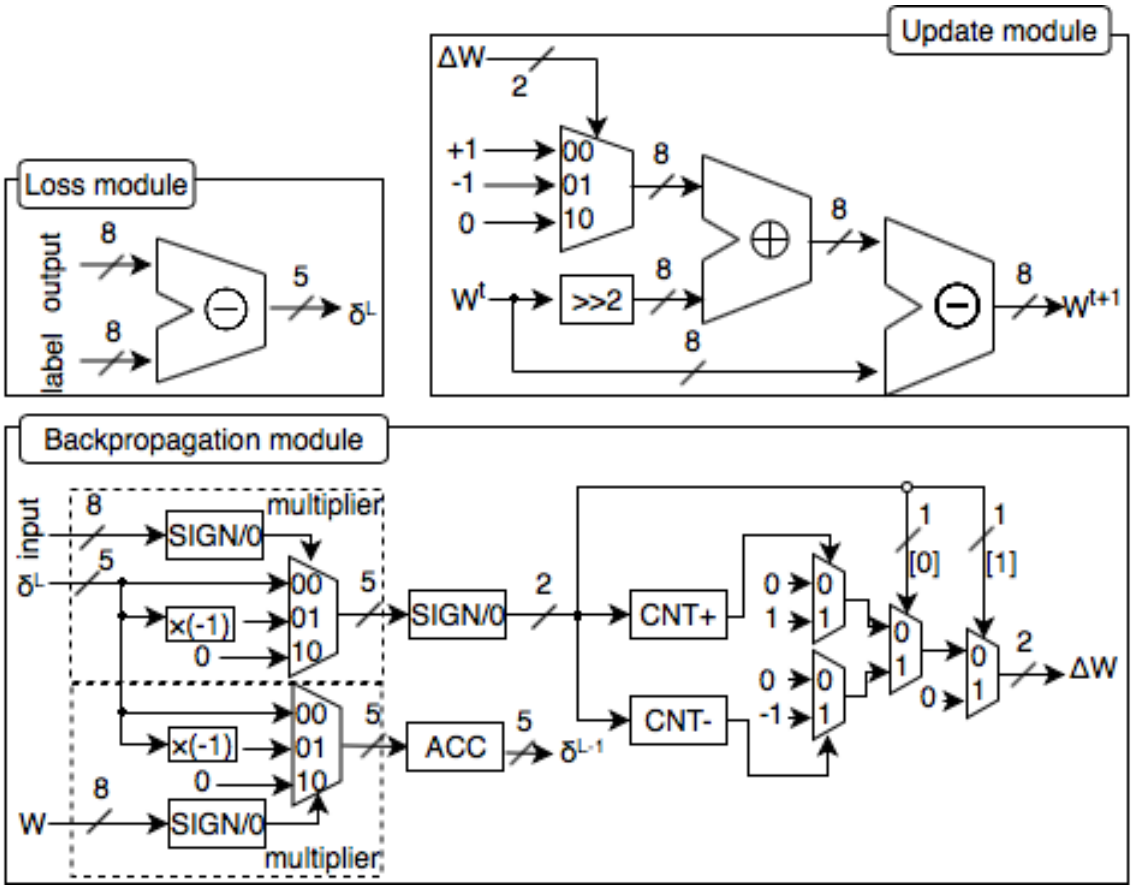

図 3.20　TBP モジュールの全体図．三つのモジュールに大別することが可能であり，それらは Loss 演算を行うモジュール，逆伝播と更新値を算出するモジュール，パラメータの更新を行うモジュールとなっている．

タをバッファとして使用し，同様に外部メモリとしては SRAM 等をパラメータや入出力，教師データの保存に使用することで，NN モデルの規模が大きくなった場合でも外部の SRAM 容量を増大することで対処できることからスケーラビリティについての担保ができている．外部メモリにはアドレスの上から順に入力層と中間層間のパラメータ，中間層と出力層間のパラメータ，入力データ，教師データ，出力データが保存されている．出力データを除き，各パラメータやデータは推論が実行される前に初期化や値の書き込みが行われる．この時に入力データと教師データは SRAM アクセス回数削減のために内部メモリ (Input buffer, label buffer) へと書き込まれ，また推論途中に中間層と出力層の活性化関数へ入力する前の値も内部メモリ (hidden buffer, output buffer) へと書き込まれている推論を終えると出力値が外部メモリの適切なアドレス箇所へと

書き込まれ，学習が開始される．本節では学習モジュールのみについて説明を行う．

図3.20にTBPアーキテクチャの全体図を示す．本アーキテクチャは三つのモジュールで構成されており，それぞれLoss算出，逆伝播演算，パラメータ更新を行う．以下にそれぞれの仔細について述べる．

Loss module

本モジュールは出力層の誤差(Loss)であるδ^Lの算出を行う．outputとlabelを各バッファから読み出し減算を行うが，8 bitの減算結果を符号部1 bitと小数部4 bitの計5 bitへ削減しその後の演算に用いる．本実装では活性化関数にLeaky ReLUとSigmoidを用いていることから活性化関数の微分値を求める演算を省略している．

Backpropagation module

本モジュールの前半部分は更新値ΔWと前層への誤差であるδ^{L-1}の算出を行い，後半部分は更新するパラメータの選択を行う．ΔWの計算には更新を行う層の入力であるinputとδ^Lが用いられ，δ^{L-1}の計算には更新を行う層のパラメータであるWとδ^Lが用いられる．これらinputとWは符号化モジュール(SIGN/0)で00，01，10に符号化され，順に+1，−1，0の意味を持つ．つまり，上位bitが0フラグbitであり，下位bitが符号bitである．本アーキテクチャは時分割演算方式であることから行列積が必要となるδ^{L-1}の計算は累算器(ACC)を用いて行った．ここで，更新されるパラメータの選択には使用リソース量の観点から乱数手法ではなく，累算器手法を用いた．乗算された結果は符号化モジュールにより符号化され，カウントモジュールに入力される(CNT+，CNT-)．カウントモジュールは入力された数値が+1，−1であった場合にカウントアップされる，カウントが事前に設定された内部の閾値を超えた場合に1を出力，カウンタは0になり，ΔWが00または01となる．カウントが内部の閾値を超えない場合にはΔWは10となる．

Update module

本モジュールは現在のパラメータW^tとΔWから新しいパラメータW^{t+1}の算出を行う．ΔWに応じてW^tの最下位bitと減算される値が選択される．この値は実際には8 bitとなる．つまり，+1の時は00000001，−1の時は11111111である．また，パラメー

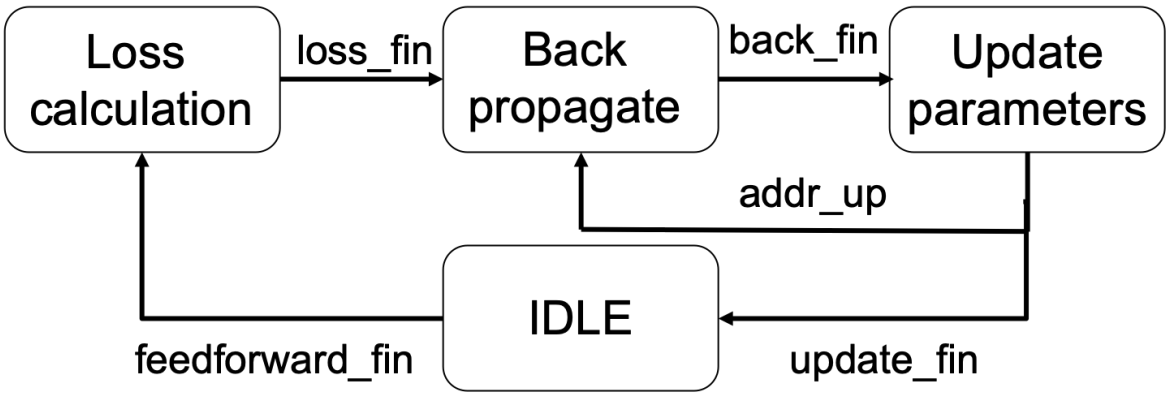

図 3.21 TBP モジュール全体のステートマシン．四つの状態からなり，休止状態，誤差計算，逆伝播，パラメータ更新の順に動作する．ただし，パラメータ更新モジュールから休止状態の遷移は全ての学習処理が終わった時になされる．

タの発散を防ぐ L2 正則化についても小規模な回路の追加で可能であり，L2 正則化を加えた誤差関数 E は式 (3.1) である．

$$E = \frac{1}{2}(output - label)^2 + \lambda |w|^2 \tag{3.1}$$

ここで，λ は L2 正則化に係るハイパーパラメータであり，w はニューラルネットワークのパラメータである．

ステートマシン

本項では提案アーキテクチャである TBP モジュールの動作をステートマシンを基に解説を行う．図 3.21 にモジュール全体のステートマシンを示す．TBP モジュールは先の回路図で触れた通り三つのモジュールから構成されている．そのためモジュール全体の状態としてはここに休止状態を加えた四つとなる．それぞれ，推論処理の終了まで待機を続ける状態，推論処理終了後の feedforward_fin 信号を受け取り次第，誤差の計算を行う状態，誤差計算の終了とともに立ち上がる loss_fin を受け取り誤差の逆伝播と更新値の計算を行う状態，更新値を受け取りパラメータの更新を行う状態となっている．ここで，本アーキテクチャは時分割シリアル演算であることから逆伝播演算を行う状態 (Backpropagate) とパラメータ更新を行う状態 (Update parameters) は，全てのパラメータ更新が終了するまで交互に動作を行う．全てのパラメータ更新が終了次第，

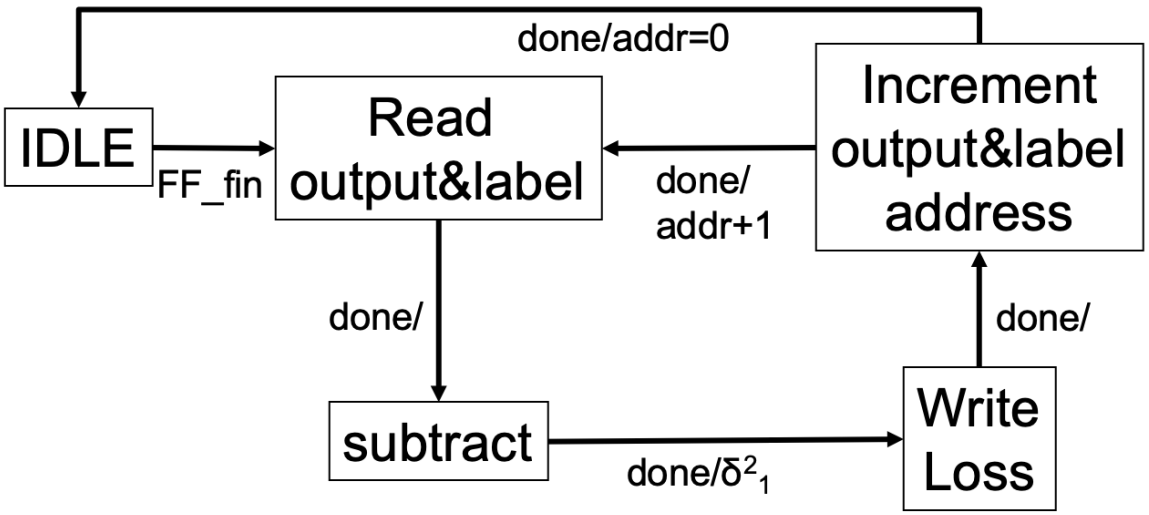

図 3.22　誤差計算状態のステートマシン．五つの内部状態から構成される．それぞれ，待機状態，出力・教師データ読み込み状態，演算状態，誤差の書き戻し状態，アドレス加算状態である．アドレス加算状態から待機状態への遷移は全てのニューロンに対する誤差の計算が終了した際に行われる．

update_fin 信号を立ち上げ休止状態へと遷移する．休止状態を除く三つの状態の詳細を以下に述べる．

Loss calculation

図 3.22 に誤差計算モジュールのステートマシンを示す．本モジュールも内部に五つの状態を持っている．それぞれ，待機状態，出力・教師データ読み込み状態，演算状態，誤差の書き戻し状態，アドレス加算状態である．待機状態からデータ読み込み状態への遷移は推論モジュールの処理が終了した際に立ち上げられる終了信号 (FF_fin) を受け取ったのちに行われる．データ読み込み状態ではバッファに保存されている出力データと教師データの読み込みを行う．演算状態では読み込んだ出力データと教師データを減算することで誤差 (δ_1^2) の計算を行う．誤差の計算が終了したらその誤差をバッファへと書き戻しを行う状態へと遷移する．誤差の書き込みを行ったら次のニューロンに対する誤差 (δ_2^2) の計算を行うために，バッファへアクセスする際のアドレス値の加算を行う．出力層のニューロン数が N_{out} であるとした場合，$\delta_{N_{out}}^2$ の計算が終了した時に loss_fin 信号を立ち上げ待機状態へと遷移する．

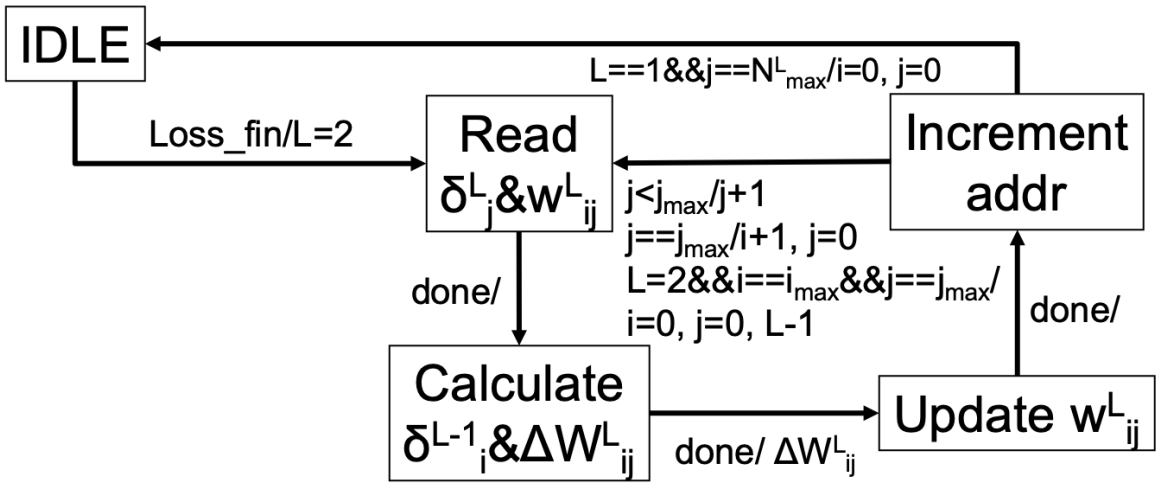

図 3.23　逆伝播状態とパラメータ更新状態のステートマシン．二つの状態が五つの内部状態から構成される．待機状態から逆伝播状態へと遷移し，誤差とパラメータの読み込み状態，前層への誤差と更新値計算状態へと移行する．その後，逆伝播状態から更新状態へと遷移し，パラメータの更新状態，アドレスの加算状態へと移行する．アドレス加算状態から待機状態への遷移は全てのパラメータに対して演算が終了すると行われる．

Backpropagate と Update parameters

図 3.23 に逆伝播モジュールとパラメータ更新モジュールのステートマシンを示す．この二つのモジュールが組み合わされ五つの内部状態を持つ．それぞれ，待機状態，誤差 (δ_j^2) とパラメータ (w_{ij}^L) の読み込み状態，前層への誤差 (δ_i^{L-1}) と更新値 ($\Delta \mathrm{W}_{ij}^L$) の計算状態，誤差とパラメータへのアドレスを加算する状態となっている．ここで L は現在計算対象としている層を，i は入力層側の計算対象としているニューロンのアドレスを，j は出力層側の計算対象としているニューロンのアドレスを示している．例えば，中間層と出力層間のパラメータ更新は $L = 2$，$i = 0$，$j = 0$ となる．中間層が 128 個のニューロン，出力層が 10 個のニューロンを持っている場合 $j = 0$ から $j = 9$ となり，同様に $i = 0$ から $i = 127$ まで計算が行われる．パラメータの更新は時分割シリアルで行われていることから，先にアドレス i を加算することで誤差バッファへのアクセス回数を削減している．アドレス $i = 127$ となった際に，アドレス j の値が加算される．アドレス加算状態から待機状態への遷移は入力層 (784 個のニューロン) と中間層 (128 個のニューロン) 間のパラメータ更新が終了した時 ($L = 1$，$i = 783$，$j = 127$) となる．

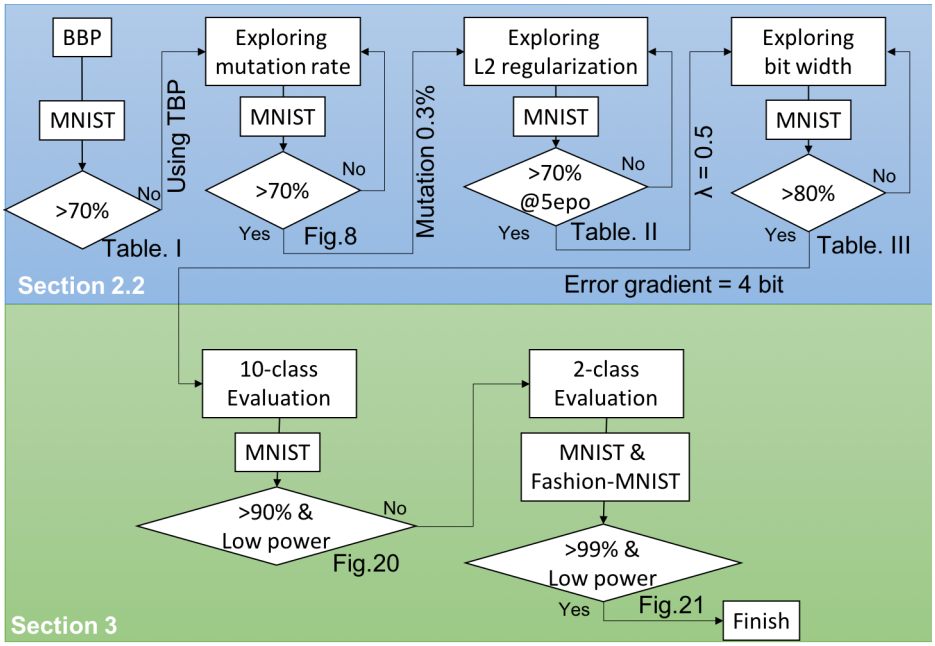

図 3.24　シミュレーションフローチャート．前節までには新規学習アルゴリズムの提案と Mutation の比率，L^2 正則化の導入，そして誤差のビット精度の探索を行った．本節では，MNIST 分類の結果と MNIST と Fashion-MNIST を用いた 2 クラス分類の結果を示す．

3.3　評価

本章では，前章で提案したエッジ AI に向けた省電力かつ省演算リソースである学習アルゴリズムの評価を，アルゴリズムとハードウェアアーキテクチャの観点から行った．以下に，まずは認識精度等のアルゴリズム評価について述べ，その後演算リソース等のアーキテクチャ評価について述べる．

3.3.1　アルゴリズム評価

図 3.24 に本節で提案学習アルゴリズムに対して行ったシミュレーションのフローチャートを示す．まず，提案した BBP アルゴリズムでは目標を達成できないため，Mutation を導入した TBP を用いることで改善を行った．次に，学習回数の増加につれ認識精度が低下することが確認できたため，L^2 正則化を実装することで精度の低下の抑制を図った．そして，認識精度のさらなる向上のために全ての値を三値化し学習処理を行うのではなく，誤差のみを多ビットとして扱うこととした．この時に，誤差の厳密な値ではなく大小関係を保持するということによりリソースの増加を微小なものとしている．

以上の結果に基づいて，本節ではより実用的なシミュレーションを実行した．第一に，MNIST データセットを使用した 10 クラス分類の検討を行った．ここで，ネットワークを 60,000 の学習データを用いて学習を行い．10,000 個のテストデータを用いて認識精度を測定した．この実験は学習回数が 30 エポックでありミニバッチサイズは 64 であるので，エポックごとに約 900 回の学習が行われた．MNIST は 10 クラスから成っているため．出力層のニューロンの個数は 10 個である．次に，MNIST データセットと Fashion-MNIST データセットの両方を使用した 2 クラス分類を行った．つまり，両方のデータセットからの任意に与えられた入力データがどちらのデータセットに属しているかを区別することを目的とした．この実験では，99%を超える識別精度を達成するために必要となる学習データサイズと学習回数の探索を行った．ここで，MNIST テストとは異なりミニバッチのサイズを 1 とするオンライン学習で行った．2 クラス分類であることから出力層のニューロンの個数は一つである．いずれのシミュレーションにおいても認識精度と消費電力の観点から，Numpy を使用したソフトウェアシミュレーションで評価を行った．比較対象としては 16 ビットに固定小数点化を行った学習手法である Fixed-BP を用いた．ネットワークは 3 層の MLP であり，入力層，128 個のニューロンから構成される中間層，および出力層で構成した．活性化関数として中間層に LeakyReLU（leak=0.25），出力層にシグモイド関数を用いた．また，損失関数は平均二乗誤差（MSE）である．実験で使用したハイパーパラメータは以下のとおりである．Fixed-BP では，学習率は 0.25 とした．推論処理のビット精度は，符号部 1 ビット，整数部 2 ビット，小数部 5 ビットであり，学習処理においては符号部 1 ビット，整数部 2 ビット，小数部 13 ビットである，一方で，TBP では Mutation は 0.003（= 0.3%）である．更新手法は乱数を用いる手法であり，正則化項（λ）は 0.5 とした．推論処理のビット精度は，符号部 2 ビット，整数部 2 ビット，小数部 4 ビットであり，学習処理においては，符号部 2 ビット，小数部 3 ビットである．

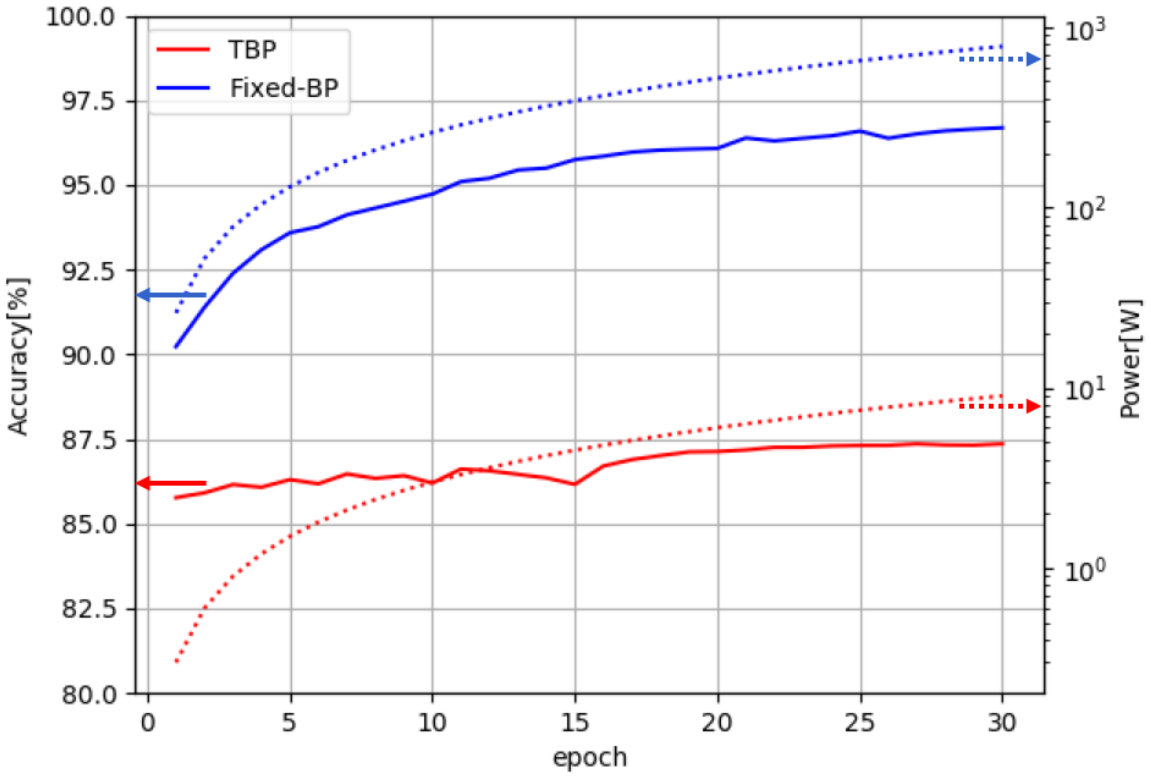

図 3.25　学習回数と認識精度/消費エネルギーの関係．実線は左の縦軸に対応し 50 回試行した平均認識精度を示し，点線は右の縦軸に対応し学習回数から見積もりを行った消費エネルギーを示す．

MNIST10 クラス分類テスト

図 3.25 に MNIST 分類の認識精度と消費電力を示す．Fixed-BP を使用した際の学習回数による認識精度の上昇とは対照的に，TBP は学習の全段階において認識精度の上昇は非常に緩やかなものとなっていることが確認できた．さらに，最高認識精度は Fixed-BP よりも低いことも確認できた．これは，更新手法のランダム性に起因するものである可能性が考えられる．前章で述べたようにパラメータは何らかのルールに従って更新されているものと推測されるが，TBP を用いた場合すべてのパラメータはいずれの更新手法 (乱数またはカウンタを用いる手法) においても等しく扱われるため，更新されるべきパラメータが更新されず，逆に更新されるべきではないパラメータが更新され

ている可能性がある．したがって，TBP は傾向としては認識精度を向上させているものの，15epoch 目のように時には認識精度の低下を招いてしまうと考えられる．この問題を解決するには，ランダム性に依存しない新しい更新手法が必要であると言えるだろう．また，シミュレーテッドアニーリング [71] のアイディアを基とし，学習初期では高い Mutation 率であるが学習が進むにつれ Mutation 率も下げていくといったスケジューリング手法も効果的であると考えられる．

本節での評価は，エッジデバイスに向けソフトウェアとハードウェアの協調設計による低電力・省リソースの学習アルゴリズム評価にとどまり，実デバイス上での評価には至っていない．そのため消費電力評価については，以下の通りに見積もった FPGA（0.15μm テクノロジを使用する Xilinx VirtexII ファミリの一部）の論理部分で消費される電力は，10^{-6}W/MHz のオーダーである [72]．一方，SRAM[73] での読み取りまたは書き込み操作は，0.18μm テクノロジー，2 ポート，O（10^3バイト）を使用した場合，10^{-3}W/MHz のオーダーである．以上の結果から，SRAM アクセスが算術演算よりも電力消費に関して支配的であると結論付けることが可能である．そのため，電力消費は SRAM[73] への読み取りおよび書き込みの回数によって見積もりを行った．認識精度と消費電力に基づいて，MNIST のような多クラス分類において Fixed-BP は TBP よりも適していると言える．ただし，エッジデバイスでは，O（10^3）の電力消費を必要とする学習アルゴリズムを適用するのは非常に難しいと考えられる．90%以上の識別精度が必要なく程々の識別精度で十分な場合は，TBP を使用して消費電力を削減することを検討できる．TBP では識別精度を上げるまでに多くの学習回数が必要となる弱点がある．逆に捉えるなら，TBP は Fixed-BP のような多くの学習回数を必要としないとも言える．>3 エポックの TBP を使用して，合理的な電力消費と識別精度の両方を達成可能である．TBP は高識別精度と低消費電力の両方を同時に達成することは現状不可能である．ただし，10 クラスでの高識別精度を必要としない場合，TBP はハードウェアコストとパフォーマンスのバランスを取ることが可能である．

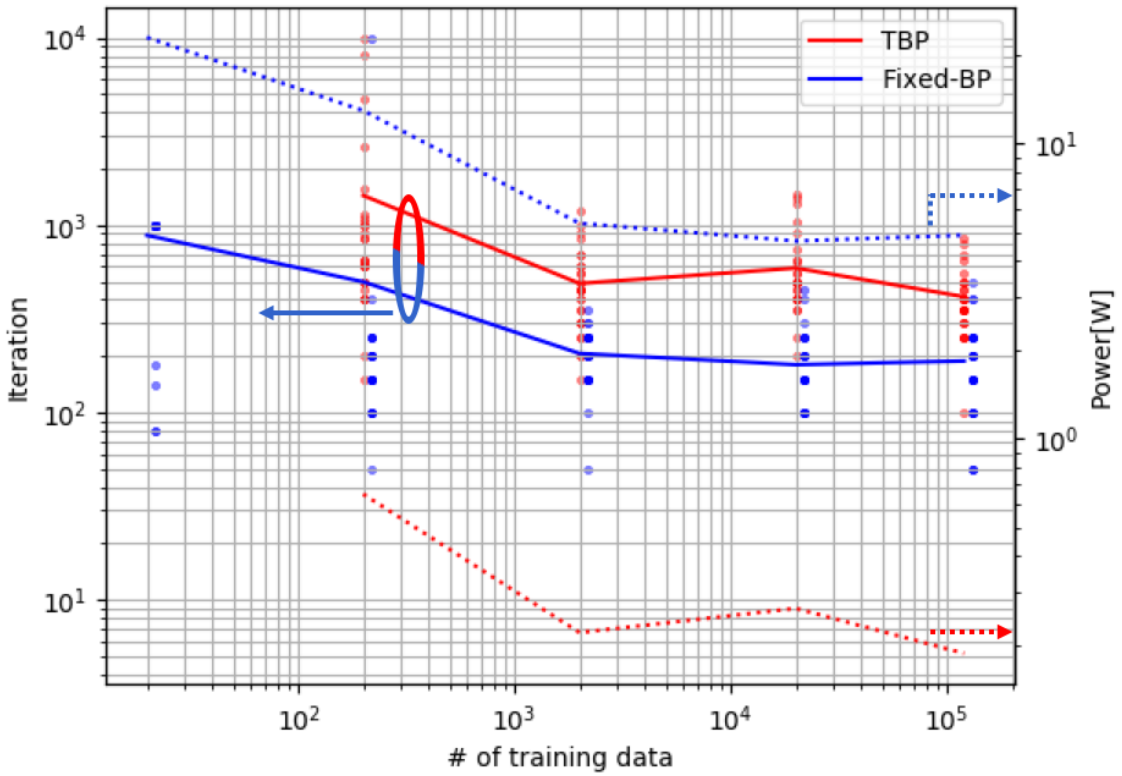

図 3.26　学習データサイズと識別精度が 99%に達するまでにかかった学習回数/総消費エネルギーの関係．各手法において総試行回数は 30 回である．実線は左の縦軸に対応し平均学習回数を示し，点線は右の縦軸に対応し学習回数から見積もりを行った消費エネルギーを示す．

フィルタリングテスト

エッジデバイスは，前項の MNIST 分類のような多クラス分類タスクよりもフィルタリング等の 2 クラス分類タスクに適していると考えられる．すでに実現しつつある Internet of Things（IoT）時代において，最も重要なタスクの 1 つは，ネットワーク帯域幅の節約をするためにそのデータをサーバへと送る必要があるのかどうかを選択することである．多くの場合において必要となるデータと必要ではないデータとでは互いに独立した分布を持っているはずである．そのため，二つの独立したデータセットを低消費電力で正しく区別することは将来の社会において重要な役割を果たすであろう．そこで本項では Fixed-BP と TBP の両方を，各種数字の画像からなるデータセットと服

飾品画像からなるデータセットの二種類のデータセットを使用して評価を行う．このテストでは，2 クラス分類であることから出力層のニューロンの個数を 1 へと変更した．

図 3.26 にネットワークが 99%以上の識別精度を達成するまでにかかった学習回数とそこから見積もりを行った総消費エネルギーを示す．本実験においてミニバッチサイズは 1 としている．結果から TBP の総消費エネルギーは $O(10^{-1})$ であり，Fixed-BP よりも 2 桁小さいことが確認できた．したがって，TBP は高識別精度と低消費電力のバランスを取ることができる．TBP では，Fixed-BP よりも数百回多く学習が必要であった．しかし，これが重大な問題になるとは考えられない．実際に現実の環境で使用される事になるエッジデバイスでは高速動作をしたとしてもセンサ間での通信がボトルネックとなることから高速動作は求められず，また，ハードウェアの動作速度は人間が認識するには速すぎることから，学習回数の違いには気付かないであろうことも考えられる．ただし，学習データセットの分布が近い場合，TBP と Fixed-BP をいずれかを使用したとしてもネットワークは高識別精度で区別は不可能である．たとえば，MNIST データセットの偶数と奇数を高精度で区別することは不可能である．したがって，高精度で区別できる各分布の距離の調査も必要となるであろう．また，低消費電力を維持しながら，ネットワークが 3 つ以上の独立した分布を区別できるかどうかの検証も今後の課題として残っている．ネットワークは，学習に各クラスで 100 を超えるデータを必要とする．学習データセットが各クラス 10 個のデータしかない場合，TBP は 99%の識別精度を達成することが不可能となる．ただし，Fixed-BP も同様に不可能であることから，これは根本的な問題であると考えられる．エッジデバイスには大量のメモリは備えられていないことから，より小さなデータセットでトレーニングできるネットワークに多大な利点がある．本実験では，学習データセットサイズが 500 を超えるもので TBP を使用する場合，ネットワークが 99%を超える識別精度を達成することが確認できる．そのため，2,000 を超えるデータセットでも学習回数は減少しない．以上をまとめると TBP は，独立したデータ分布を使用したフィルタリングテストで高識別精度と低消費電力を同時に実現可能である．一方，TBP と Fixed-BP は，MNIST データセットの偶数と奇数を区別するなど，近い分布を使用したフィルタリングテストで高い識別精度を達成することは現状不可能であることが確認できている．

表 3.4　各種演算リソースの使用率．乗算器や SRAM 使用率の減少率に比べるとロジックエレメント (LEs) とレジスタの減少率は少ない．

	Total LEs	Total Registers	9 bit Multipliers	SRAM usage
FixedBP	18,073	12,246	11	204kByte
TBP	15,230	10,738	1	102kByte
Reduction	15.73%	12.31%	90.9%	49.8%

3.3.2　アーキテクチャ評価

本節では，提案アーキテクチャの評価について述べる．実験環境を以下に示す．使用したデータセットは手書き数字データの MNIST である．アーキテクチャの評価には Intel 社の FPGA (MAX10 10M25SCE144I7G) を使用し，Quartus Prime Standard Edition 15.1 を用いてコンフィグレーションを行った．NN のモデルは 3 層の多層パーセプトロンであり，入力層は 784 のニューロンを持ち，中間層には 128 のニューロンとし活性化関数として Leaky ReLU，出力層は 1 または 10 のニューロンとし活性化関数として Sigmoid を使用した．エッジ上での運用という観点からミニバッチサイズは 1 とした．比較対象としてアルゴリズム評価と同様に 16 bit に量子化した Fixed-BP を用いた．

前節の評価において，2 クラス分類には十分適用可能であることが確認できたことから，本節では機械学習のスタンダードな評価手法である MNIST10 クラス分類について評価を行った．表 3.4 に Fixed-BP から TBP へと変更した際の各リソースの減少率を示す．総ロジックエレメント (LEs) 数が 15.73%，総レジスタ数が 12.31%，乗算器数が 90.9%，SRAM 使用率が 49.8%減少していることが確認できた．以下にその理由について考察する．まず，LEs の減少についてであるが，TBP では乗算器の代わりにマルチプレクサを使用することで乗算を行う．このマルチプレクサの実装を LEs を用いて行ったため 10%台の減少率に収まったと考えられる，続いてレジスタ数についてであるが，内部メモリとして使用しているレジスタの内 TBP で削減に大きく寄与するものは前層への誤差である δ_hid レジスタである．これが 16 bit から 5 bit へと削減され，2208 から 690 への約 1500 の削減となっている．ここで，移植性の観点から IP カタログによる Block RAM の使用ではなくレジスタを用いた実装を行った．つまり，Intel 社の FPGA だけではなく他社の FPGA にも適用可能である．これら LEs とレジスタについては比

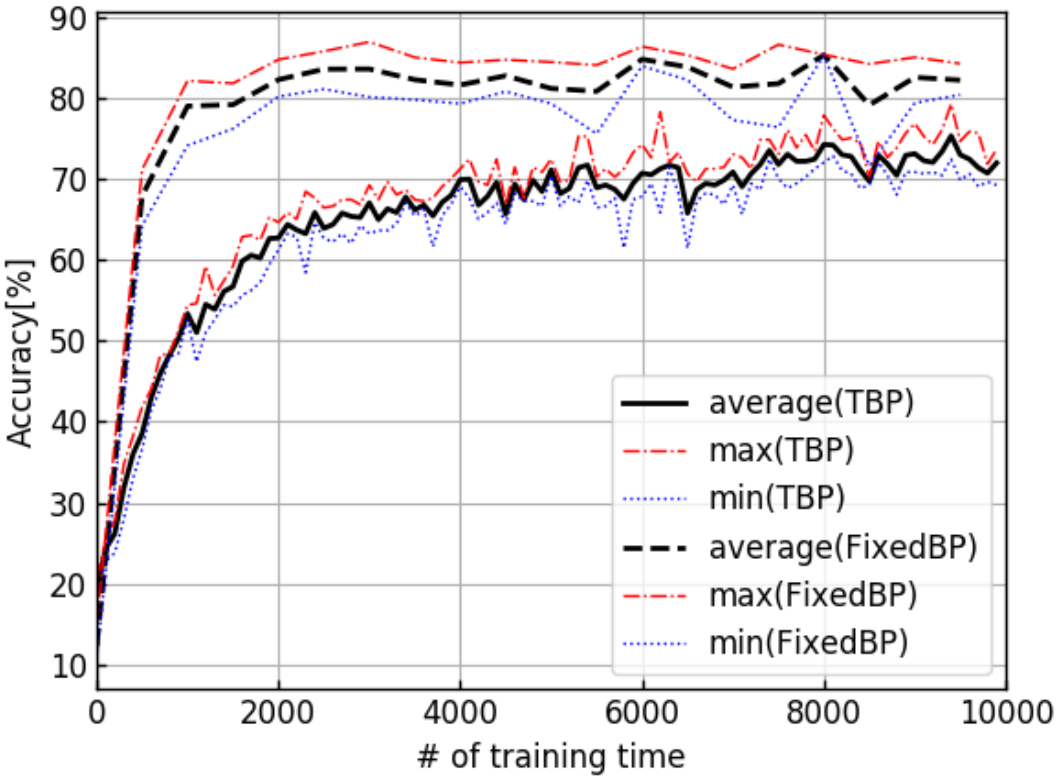

図 3.27 MNIST 分類テスト．点線が Fixed-BP，実線が TBP の平均精度を示す．各線の上下に存在する線は最大値と最小値を示している．

較対象である Fixed-BP アーキテクチャ自体がエッジ AI 向けに設計されていることから，大幅な削減率を達成できていないと考えられる．次に乗算器であるが，推論のみに使用されることから使用数は 1 となっている．最後に SRAM 使用率は，使用する NN のモデルが増大することでパラメータ数が支配的となる．このパラメータは Fixed-BP では 16 bit で保存する必要が生じていたが，TBP では 8 bit での保存を可能にしたため約半分となった．

図 3.27 に MAX10 上に実装した NN モデルを用いた MNIST10 クラス分類の結果を示す．Fixed-BP の平均認識精度を点線で示し，同様に TBP の平均認識精度を実線で示した．また，各線の上下に最大値と最小値も示した．実行時間の兼ね合いから総学習回数は 0.17epoch とした．Fixed-BP と比べた場合に TBP の精度は約 10%下回っていることが確認できた．現在のパラメータ更新規則は乱数に従って更新の有無を決定している．タスクの難易度が低い場合には更新されるべきパラメータが更新されなかったとしてもモデルの柔軟性により高精度を獲得できていたものが，タスクの難度上昇につ

れ精度を獲得できなくなったためであると考えられる．これを解決するためには，現在提案している乱数に従った手法ではなく，誤差の重み付け等により正しく更新されるべきパラメータを更新するという手法へと発展させる必要がある．また，Fixed-BPでも精度が80%台と低い理由の一つにミニバッチサイズが1であるということが考えられる．

3.4 結論

本章では誤差逆伝播法と確率的勾配降下法に基づいたシンプルで計算効率の高い学習アルゴリズムである三値バックプロパゲーション法 (TBP) の提案とそれを低電力かつ低演算リソースで実行可能なハードウェアアーキテクチャの提案を行った．提案アルゴリズムはパラメータ更新の際にLSBを+1，−1，0のいずれかの値で更新する手法である．これにより，低消費電力かつ低演算リソースを要求するエッジAIデバイスのオンライン学習のために低ビット精度の学習を可能としている．また，パラメータの更新を確率的に行うことによりメモリアクセスの回数を大幅に削減することで電力消費量の削減に繋げている．提案アーキテクチャは誤差計算モジュール，逆伝播・更新値計算モジュール，パラメータ更新モジュールの三つから構成されている，アーキテクチャ全体の動作はそれぞれのモジュールが順に動くように行われる．つまり，休止状態，誤差計算状態，逆伝播・更新値算出状態，パラメータ更新状態の順に遷移する．また，提案アーキテクチャは時分割シリアル演算方式であることから，誤差の算出や逆伝播演算，更新値を求めパラメータの更新に到るまで一つのニューロン，パラメータに対して演算を行う．実験により，2クラス分類タスクで99%の識別精度を達成するまでに消費したエネルギーが従来手法（Fixed-BP）に比べ2桁小さいことを示した．一方で手書き数字データセットであるMNISTを用いた10クラス分類では，要求リソース量をLE数15.73%，総レジスタ数12.31%，乗算器数90.9%，SRAM使用率49.8%まで削減したものの，認識精度の面においてはパラメータ更新方法の再検討の面などに課題を残すことになった．複雑なタスクで高認識精度を実現するために使用されるクラウドベースのAI処理とは対照的に，エッジAI処理は単純な分類タスクに使用されると考えられる．これらのタスクでは，さまざまな環境で動作するためにエッジAI上でのオンライン学習技術が必要となる．どの環境においてもそのデータが必要かどうかを判別するなど単純な知識を環境毎に適応し獲得するには，低消費電力の学習アルゴリズムが必要となる．クラウド–フォグ–エッジAI社会では，通信コストを削減する

ためデータの取捨選択がより重要となるだろう．提案手法はこのようなAI社会において効果的な解決策の一つになり得ることを実証した．

参考文献

[15] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[16] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). arXiv: 1412.6980.

[35] S. Zhou, Y. Wu, Z. Ni, et al. *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*. 2016. DOI: 10.48550/ARXIV.1606.06160.

[57] J. Hu, L. Shen, and G. Sun. “Squeeze-and-Excitation Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[58] J.-Y. Zhu, T. Park, P. Isola, et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[59] A. van den Oord, Y. Li, I. Babuschkin, et al. “Parallel WaveNet: Fast High-Fidelity Speech Synthesis”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. **80**. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 3918–3926.

[60] J. Dean. *Machine Learning for Systems and Systems for Machine Learning*. 2017.

[61] S. Pichai. *Google Keynote*. Google I/O. 2018.

[62] Fujitsu. *DLU^{TM}: Deep Learning Unit*. 2018.

[63] K. Guo, S. Zeng, J. Yu, et al. “[DL] A Survey of FPGA-Based Neural Network Inference Accelerators”. In: *ACM Transactions on Reconfigurable Technology and Systems* **12**.1 (Mar. 2019). DOI: 10.1145/3289185.

[64] W. Zhao, H. Fu, W. Luk, et al. “F-CNN: An FPGA-based Framework for Training Convolutional Neural Networks”. In: *2016 IEEE 27th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*. 2016, pp. 107–114. DOI: 10.1109/ASAP.2016.7760779.

[65] Y. Lecun, L. Bottou, Y. Bengio, et al. “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* **86**.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[66] F. Leens. “An Introduction to I2C and SPI Protocols”. In: *IEEE Instrumentation & Measurement Magazine* **12**.1 (2009), pp. 8–13. DOI: 10.1109/MIM.2009.4762946.

[67] M. D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. DOI: 10.48550/ARXIV.1212.5701.

[68] G. Hinton, N. Srivastava, and K. Swersky. “Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent”. In: *Cited on* **14**.8 (2012), p. 2.

[69] B. Xu, N. Wang, T. Chen, et al. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. DOI: 10.48550/ARXIV.1505.00853.

[70] R. Caruana, S. Lawrence, and C. Giles. “Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. **13**. MIT Press, 2000.

[71] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. In: *Science (New York, N.Y.)* **220**.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. eprint: https://www.science.org/doi/pdf/10.1126/science.220.4598.671.

[72] L. Shang, A. S. Kaviani, and K. Bathala. “Dynamic Power Consumption in Virtex™-II FPGA Family”. In: *Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays*. FPGA ’02. New York, NY, USA: Association for Computing Machinery, 2002, pp. 157–164. DOI: 10.1145/503048.503072.

[73] Visit. *http://www.csd.uoc.gr/%7Ehy534/03a/s31_ram_bl.htm*.

第4章　アナログCIMデバイスに向けたオンライン学習アルゴリズムとそのアーキテクチャ

4.1　導入

人工知能 (AI)，特に深層学習は，画像認識 [1]，自然言語処理 [5]，[6]，強化学習 [9] などの分野で優れた性能を発揮している．特に，[6] で，自然言語処理 AI モデルの性能は，パラメータ数，学習データセットサイズ，演算コストに対する予算の 3 つの変数によって支配されており，これらの変数を常に増加させることで，モデルの性能が無限に向上する可能性があることを示唆している．このようなアプリケーションは主に，Tensor Processing Units(TPU) または数百ワット規模の Graphics Processing Units(GPU) を含むクラウド上に構築される．一方で，スマートフォンやドローン，IoT スマート製品などの小さな製品を含むエッジベースの AI システムでは，Edge TPU，Jetson nano 等に代表される小規模な AI アクセラレータが推論処理エンジンとして使用される．従来のエッジベースの AI システムの多くは，クラウド上での事前学習を必要としている．つまり，低消費電力を謳うエッジデバイスではあるが，その背景に大量の電力消費とハードウェア資源を必要としている．さらに，上記エッジデバイスは推論処理のみを行い，学習処理もサポートされていたとしても限られた能力しか有していない．つまり，現在のエッジ AI システムはエッジデバイスを取り巻く固有の環境の学習には適していないことを意味している．従来のエッジデバイスは学習機能がないことから環境の変化に即座に対応できず，応用を考えた際に，AI による自動運転車が事故を起こすことや AI 制御された生産ラインが不良品を排出することは避けられないと考えられる．このような問題に対応するためには，環境の変化が生じた際には収集データをクラウドへ転送し，エッジデバイスに搭載されている AI モデルをアップデートするまでシステムを停止する必要が生じる．また，学習のためにエッジデバイス上に蓄えられた生データをクラウドに送信する必要があるため，プライバシーが保護されず，サイバー

攻撃に対して脆弱であるという危険性も内在する．従って，オンライン学習はエッジ AI システムにとって重要な技術である．

ディープニューラルネットワーク (DNN) モデルを学習するためには確率勾配降下法 (SGD)[15] および Adaptive Momentum Estimation(Adam)[16] などの手法が考えられる．しかし，このような手法は限られた電力および演算リソースを有するエッジデバイスでの実装には適していない．そこで，演算時に必要な情報量を削減することができる量子化方法の研究が行われている [74, 30, 35]．[75, 76, 77] において研究されている Ternerization は，推論および学習中の性能を維持しつつも，消費電力および演算リソースの削減に特に成功することを示した．[75] や [76] では，Ternerization を利用する手法が提案され，認識精度の低下が抑えつつも最先端の結果が得られたことを示した．また，[78, 79, 80] では，エッジデバイスを通じた学習に関する研究が行われている．ここでビット精度について取り上げると，[78] および [80] では，8 ビット以上の高い精度が使用されたが，[79] では，1–16 ビットとビット精度に幅を持たせることで柔軟なアーキテクチャが提案された．同様に学習処理機能を搭載したエッジ AI チップについては，[81, 82, 83, 84, 85, 45, 46] など多くの論文でその成果が発表されている．このうち，[81, 82, 83, 84] は，モデルのパラメータの保存先として外部 DRAM 使用している．そのため，重みのスパース性を利用して圧縮したとしても，消費電力は約 500 mW と非常に大きい．一方，不揮発性メモリ (NVM: Non Volatile Memory) の一つである ReRAM を搭載した [45] では，2 MB 以上のウェイトを格納することが可能であり，また，[45] では，Low Rank Training(LRT) アルゴリズムを用いてパラメータ更新回数を削減することにより，100mW オーダでの消費電力を実現している．しかし, パラメータを保存するメモリとして単に SRAM の代わりに ReRAM を使用する [45] のような方式ではフォンノイマンボトルネックが解消されず抜本的な消費電力の削減にはつながらない．ここで近年のエッジ AI 研究ではプロセッサとメモリとで生じるボトルネックを解消し，より低い消費電力での演算を可能とするために，メモリ上で NN の基本操作となる積和演算を行う Computing-in-Memory (CIM) アーキテクチャ研究が盛んになっている．[85] は SRAM を組み込んだ CIM デバイスであり，消費電力は 100mW オーダである．浮動小数点 bFloat16 まで扱えるが，SRAM であるため容量は 160 kB に制限される．この CIM 構成のアーキテクチャに不揮発性の CIM デバイスを利用することこそが，エッジ AI に適したアーキテクチャであると考えられる．[46] ではシミュレーションによる結果に留まっているものの，NVM を用いた CIM デバイスにより 100 mW 以下の低消費電力を実証した．本章における提案手法において，ReRAM ベースの CIM デバイスで

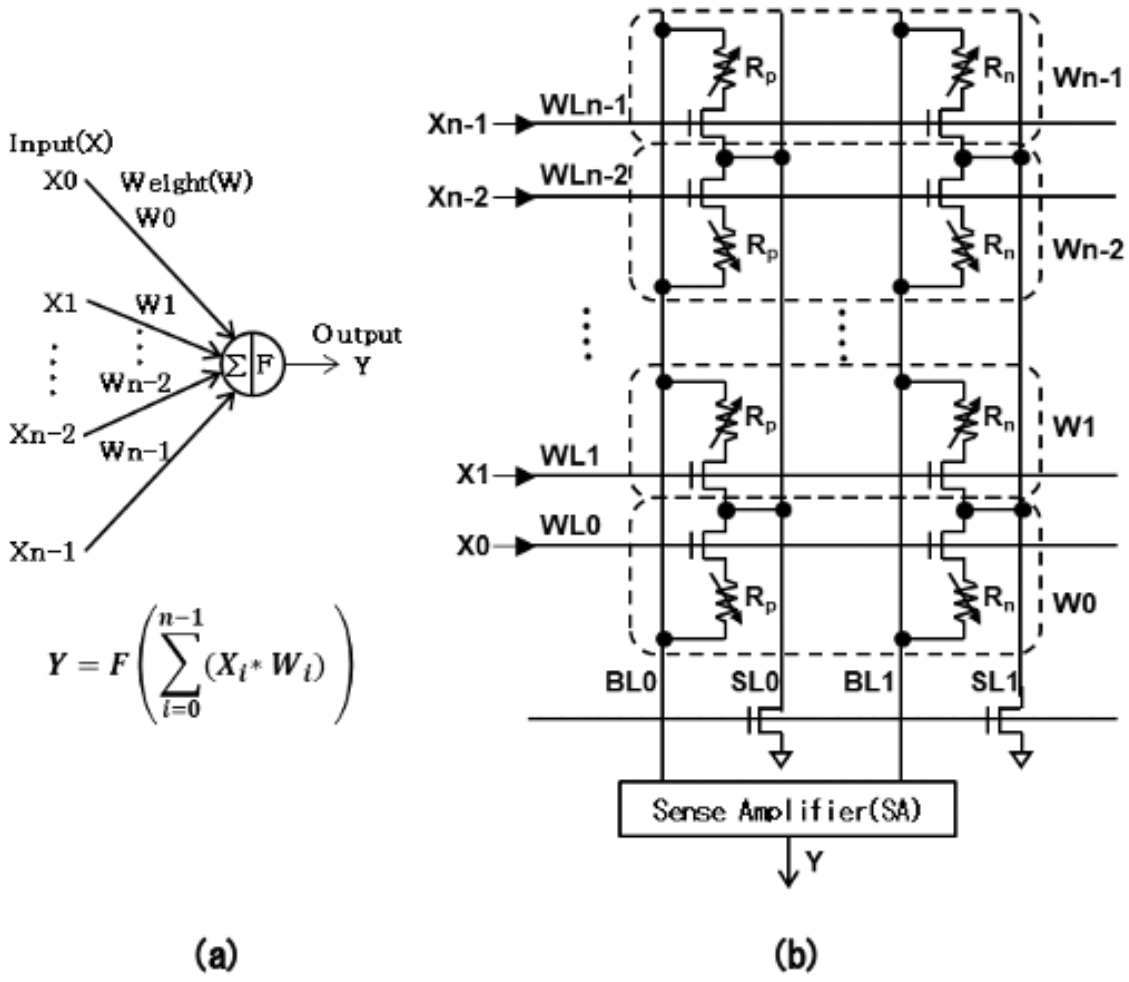

図 4.1　NN 内のあるニューロンとそれに対応する RAND チップの回路図．(a) ニューロンの模式図とその演算内容．(b) ニューロンに対応する RAND チップ内の演算回路．

ある RAND (Resistive Analog Neuromorphic Device) チップ [86, 42] を，エッジ AI 上のオンライン学習システム構築に向けて使用した．以下に，RAND の概要を示す．

4.1.1　RAND チップ概論

図 4.1 に RAND チップ内のあるニューロンにおける回路構成図を示す．図 4.1(a) は NN におけるニューロンの演算模式図であり，図 4.1(b) はこれと等価な演算を行う回路の構成図である．1 トランジスタ 1 レジスタ (1T1R) メモリセル構成の場合では，抵抗値は正の値しか取ることができない．そのために，正負の範囲で値を持つある 1 つの

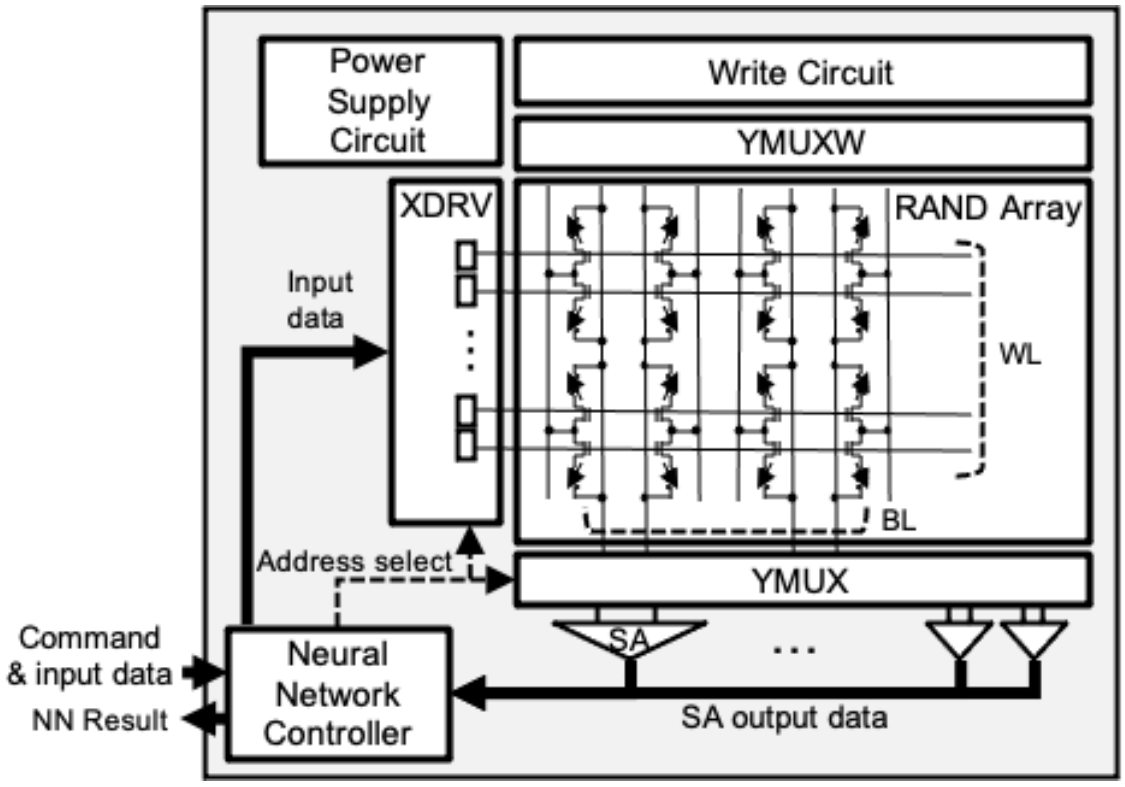

図 4.2　RAND チップアーキテクチャ

パラメータを表現するために 2 つの抵抗変化素子 (R_p,R_n) を用いる必要がある．具体的には，あるパラメータが正の値を持つ場合ではアナログ抵抗値は R_p に書き込まれ，残る R_n は高抵抗状態 (電流 0) に設定される．同様にパラメータが負の値を持つ場合ではアナログ抵抗値が R_n に書き込まれ，R_p は高抵抗状態 (電流 0) に設定される．入力である X は選択トランジスタのワード線に割り当てられ，$X = 0$ はワード線選択により制御され，$X = 1$ はワード線非選択により制御される．入力に応じて複数のワード線を選択することにより，ビット線 BL0 については正のパラメータの積和演算結果が得られ，ビット線 BL1 については負のパラメータの積和演算結果が得られる．センスアンプ (SA) は，ビット線 BL0，BL1 を流れる電流量をそれぞれ比較することにより，積和演算結果が正の値である場合には $Y = 1$ を出力し，負の値である場合には $Y = 0$ を出力する．すなわち，RAND ニューロン回路は入力 X と出力 Y が 2 値 (0/1) であり，入力 X とパラメータ W の積和が CIM 技術を用いたアナログ演算であり，活性化関数がステップ関数であるニューロン回路の一種である．

図 4.2 に RAND チップのアーキテクチャを示す．RAND アレイは，アレイ状に配置

されたメモリセルと複数のワード線 (WL) およびビット線 (BL) とを有する．行選択回路 XDRV はメモリセル選択トランジスタが接続された状態で，複数の WL を選択的に駆動する．列選択回路 YMUX は，複数の BL の中から所定のビット線対を選択して SA に接続する．SA は選択された BL ペアに流れる電流を比較し，“0” と “1” のデジタル値を出力する．コントローラは，外部入出力制御，XDRV・YMUX アドレス制御などのニューラルネットワークの動作制御を行う．列選択回路 YMUXW および書込回路は，RAND アレイへのパラメータ書込時に列選択および書込電圧印加を行う．ここで，RAND アーキテクチャにおける NN 推論処理のワークフローは，以下のように説明することができる．コントローラは，外部の NN 推論コマンド，NN 入力データ，および NN 構成情報を受信し，入力データを XDRV に設定し，NN 構成に基づいて YMUX ビット線選択状態を指定する．入力値が 1 のワード線は選択状態に駆動され，値が 0 のワード線は非選択状態に駆動される．その後，SA が駆動され，選択されたビット線対における電流の量を決定し，それをニューロン出力結果としてコントローラに出力する．SA 出力結果は，NN の構成に応じて，XDRV への入力データを次段の入力として設定するか，最終段の結果として出力するかが決定される．RAND アーキテクチャでは，XDRV の SA からの出力を次段の入力とし，SA で電流値を決定する本ワークフローを繰り返すことで，多層 NN の演算を実現している．

上述したように，RAND チップは NN パラメータを多値で保存することが可能であるものの，ニューロンの出力は，ステップ関数の出力となる 0 または 1 である．よって RAND チップを用いた学習可能なアーキテクチャを構築する際には，ステップ関数は微分不可能である点と，内部の活性値が読み出すことが不可能であるという点から通常の BP ではなく，Digital BP (DBP)[18, 87] が必要となる．以下に，DBP について簡潔にまとめる．

4.1.2　DBP アルゴリズム概論

図 4.3 に，DBP による中間層の教師信号 T_j の予測方法ついて概略図を示す．決定因子 S_j は，通常の BP でも用いる出力層と教師信号の誤差に基づいて，以下の方程式によって決定される．

$$S_j = \sum_i (O_i - T_i) W_{ij} = \sum_i S_{ji} \tag{4.1}$$

ここで，O_i は出力層の出力値，T_i は教師信号，W_{ij} は中間層と出力層間のパラメータ (重み)，j は隠れ層のノード，i は出力層のノードに対応する値である．この決定因子

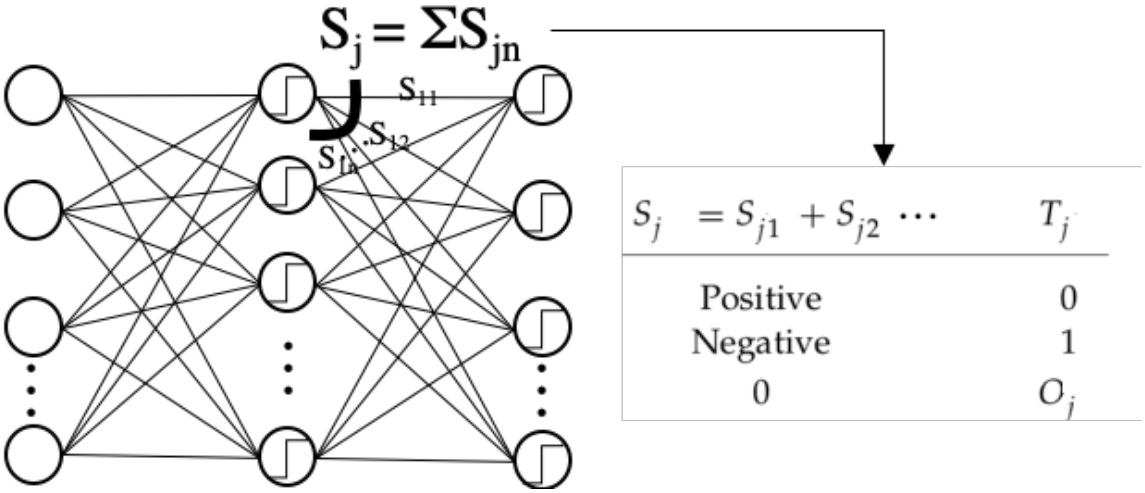

図 4.3　Digital BP を用いた教師信号予測方法．予測因子 S_{jn} の総和を用いて中間層の教師信号 T_j を予測する．

の符号に基づいて中間層の教師信号 T_j を予測・決定する．

本章では以下の内容について説明する．

- RAND チップのような CIM デバイスに向けた新規学習アルゴリズムと，CIM デバイスの推論と学習を制御する省リソースオンライン学習アーキテクチャを構築した．
- DBP を拡張した提案アルゴリズムにおいて，従来では性能が発揮できない出力層が多ノードである場合でも高い性能を発揮していることを示す．
- 学習処理である DBP については CIM 構成ではなく外部の FPGA で処理するが，nvCIM 構成でのオンライン学習をゼロから実装し，低消費電力の低ビット学習によりエッジ AI システムで 100 mW 以下での学習が可能となり，特に学習コアの演算効率は 7.77 GOPS/W であることを示す．
- 特に高速処理が要求されない，エッジ AI が投入先環境との相互作用で学習する場合 (日，時間，分の時間スケール)，において提案手法が最小電力消費アーキテクチャであることを示唆する．

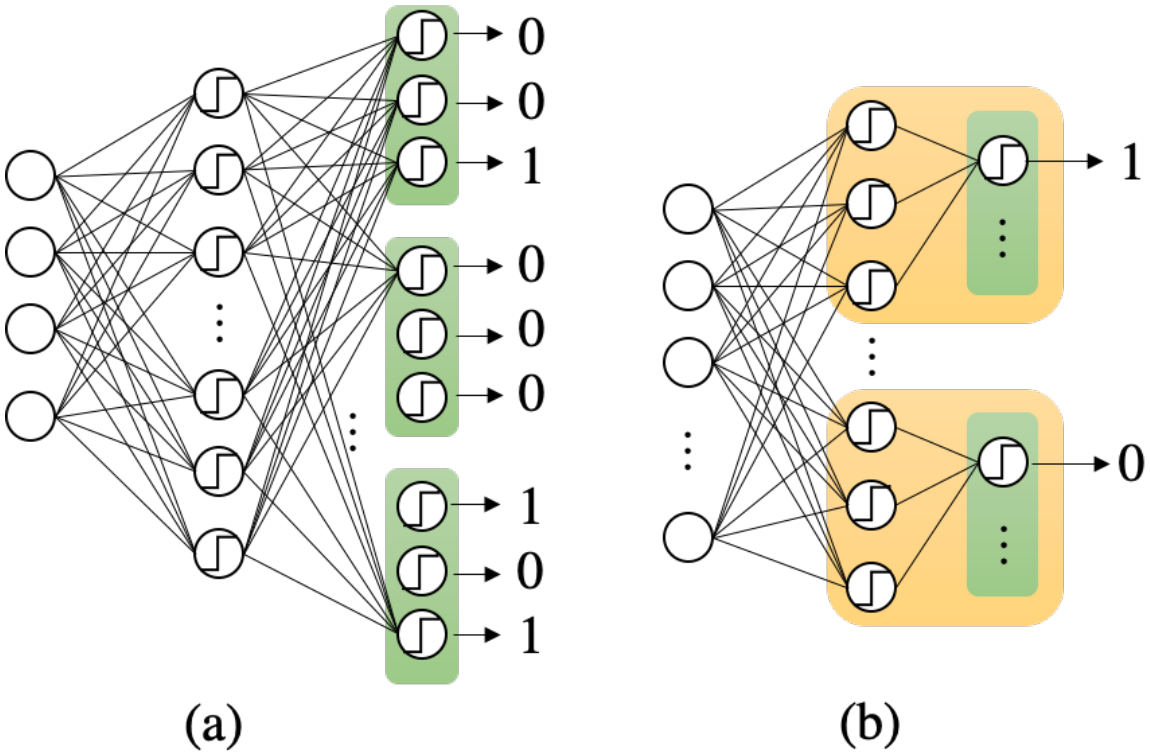

図 4.4　教師信号予測精度向上のためのネットワークモデル図．(a) 出力層のノード数を 1 から N 個に増やすクラスタ化を行なったネットワーク．(b) 中間–出力層間の重みを全結合からクラス毎に部分結合させたネットワーク．

4.2　提案手法

前述の通り活性化関数にステップ関数など微分不可能な関数を持つ場合にも誤差逆伝播法を適用できるように改良したアルゴリズムとして DBP が提案されている．この DBP アルゴリズムはニューラルネットワークの構造が小さなもので済むタスクである XOR の学習や XOR と XNOR の同時学習については 100%の識別精度で学習可能である．一方で，出力ノード数が複数になった際には十分な識別精度を確保できないという問題点がある．一般に NN は出力を複数個持つ場合が多いために，この解決は喫緊の課題となる．本章では，この解決に向けて DBP アルゴリズムの改良と，それを低電力で実行するアーキテクチャについて説明を行う．

4.2.1 DBP アルゴリズムの改良

DBP は大きなネットワーク構造が必要となる Iris データセットの識別や MNIST 分類問題に対しては十分な性能を発揮できないことが確認できた．本節では，この問題を解決するために多数決方式の導入と中間–出力層間部分結合の導入を提案する．

多数決方式

DBP はその使用用途から活性化関数にステップ関数を用いている．つまり，出力値は 0 or 1 という 2 値に限定されることから複数の出力ノードが 1 を出力するという識別ミスが発生しやすくなると考えられる．活性化関数として用いられることの多いシグモイド関数や tanh 関数の出力値は 0~1 や-1~1 といった多値であるために柔軟性を持っており，そこから最大値を NN の識別結果として用いることが可能である．ここで，出力がバイナリであるモデルに対しても出力値の柔軟性を確保するため，図 4.4 中 (a) に示すように出力ノードの 1 つが 1 つのクラスに対応するのではなく，複数個の出力ノードで 1 つのクラスに対応する多数決方式の導入を行った．図中の出力層において，緑色で囲われた出力ノード群がそれぞれのクラスに対応する．クラスタサイズ (多数決方式に用いるノード数) を奇数にし各クラスタ内で多数決を取ることで最終的な出力を得る．例えばこの例においては，出力結果は上から 3 番目のクラスとなる．多数決方式の導入により冗長性の確保が可能となり，より柔軟な出力値を取ることでより高い認識精度を得ることが可能となる．

中間–出力層間部分結合

DBP の原理的な問題として出力層のノード数が増えるに従い中間層の教師信号の予測精度は不確かなものになるという点が挙げられる．これは教師信号の予測を逆伝播された値の総和を符号化することで求めていることに起因する．この教師信号の予測能力低下による識別精度の低下は，出力層のノード数が 3 である Iris データセットの場合に多数決方式の導入による汎化性能向上で回避可能であった．しかし，10 個の出力ノードを持つ MNIST データセットの場合には回避不可能であることが確認された．出力クラスの増加はクラス間での誤差に関する情報量が相対的に低下する．つまり，識別タスクにおいては 1 を出力するべきクラスが一つだけであるのに対して，0 を出力するべきクラスは複数存在するために識別精度の低下を招くと考えた．この解決法として図 4.4 中 (b) に示すように中間層と出力層間を全結合にするのではなくクラス毎に独

立した結合をする部分結合方式を導入した．決定因子の正負は教師信号との誤差であるため，中間層から出力層をクラス毎に独立させることで前述の情報量の低下を抑制し教師信号の予測精度向上が可能となる．

ここに，戯れであるが一考察を遺しておく．出力ノードの増加が相対的に情報量の低下を招くのなら，そこにハイパーパラメータを導入することでもこの回避が可能であると考えられる．教師信号は決定因子の符号により決定され，決定因子は4.1が示すように逆伝播された誤差の総和を取ることにより決定される．ここの総和を重み付けして取ることにより回避が可能ではないだろうか．前述した部分結合方式は簡潔に記載するとクラス毎にMLPを構築するという方式である．そのために，必要となるパラメータ数は莫大なものとなる欠点を内包し，これはメモリ容量に制限のあるエッジデバイス運用を考えると致命的なものであると考えられる．本考察による重み付けによる教師信号の予測手法はパラメータ数の増大を抑えつつ性能を維持する手法となり得るために，真にエッジAIに向けた学習アルゴリズムなのではないだろうか．

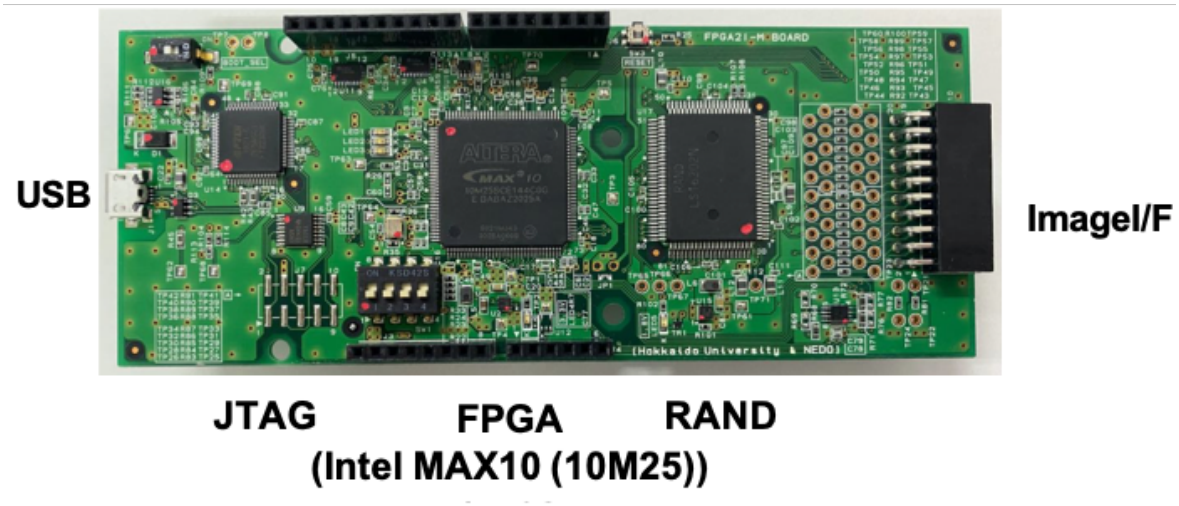

図 4.5 RAND チップと FPGA チップを同一ボード上に集積する提案デバイス．イメージセンサ用のインタフェースや Arduino 用の接続端子を備える．

4.2.2 DBP アーキテクチャ

本節では，前節で述べた DBP アルゴリズムと RAND チップの制御を実行するアーキテクチャについて説明する．図 4.5 に作成した FPGA ボードを示す．本提案ボードは FPGA に MAX10 10M25 シリーズを搭載しており，推論処理とパラメータの保存を行う RAND チップのほか，FPGA コンフィグレーション用の JTAG コネクタ，電力供給とシリアル通信を兼ねる USB 端子，イメージセンサ用のインタフェースを備える．また，本ボードは Arduino 用の AI 演算アクセラレータとして設計されていることから，マウント用の接続端子も備えている．図 4.6 に提案システムの概観ブロック図を示す．FPGA 内部には Arduino 等の外部デバイスとの通信モジュール (SPI Module)，DBP 演算の制御等を行う Controller，DBP 演算を行う DBP モジュールそして RAND チップを制御する RAND Controller が存在する．NN の構成情報や入力データといった初期設定は Arduino 等の外部デバイスを通じて行い，また，これらのデバイスを用いて推論や学習の開始信号を入力する．さらに，DBP モジュールは 3 つの子モジュールから構成されている．それぞれ，DBP 演算を行いパラメータの更新値を算出と更新を行う PE (Processing Element) と，パラメータの一時保存を行う Internal buffer と，パラメータの解像度を維持するための正規化モジュールである．本アーキテクチャは CIM デバイスである RAND チップを推論時と学習時において次のように用いる．推論時には RAND チップをパラメータを保存するメモリとしての役割に加え，順伝播演算の行列演算を

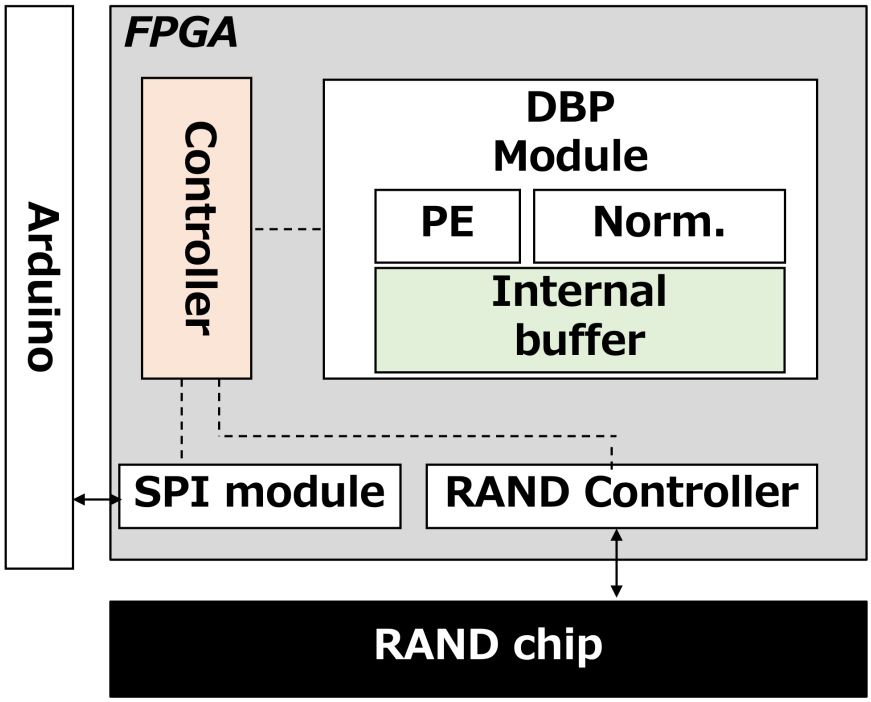

図 4.6　提案FPGAアーキテクチャの概観図．提案アーキテクチャはSPI通信用モジュールとDBPモジュール，全体制御モジュールとRANDチップ用の制御モジュールから構成される．

行う並列積和演算器として用いる．そのため，FPGAはRANDチップを制御するコントローラとして動作する．一方，学習時にはRANDチップを外部メモリとしてのみ用いている．そのため，FPGAはRANDチップ制御動作のほか，逆伝播演算やパラメータ更新という学習処理も行う．この時の本アーキテクチャはフォンノイマン型のアーキテクチャとなるためにフォンノイマンボトルネックが問題になることが予想される．そこで，DBPモジュール内部にInternal bufferとController内部に0スキップ処理を設けることで，メモリアクセス回数の削減を図った．

DBP core

本項では，DBPアルゴリズム演算を実行するコアの詳細を説明する．DBPコアは，DBPコントローラからの開始信号を検出すると動作を開始し，その内部はステートマシンによって制御される．図4.7(a)に誤差(δ_n)を算出するモジュールを示す．本モジュールは推論処理により得られた出力(OUT)と教師信号(SV)とを入力に持つ．つまり，本誤差演算回路は上記2種の1ビット入力から2ビットの出力を一つだけ返す．誤差(δ_n)

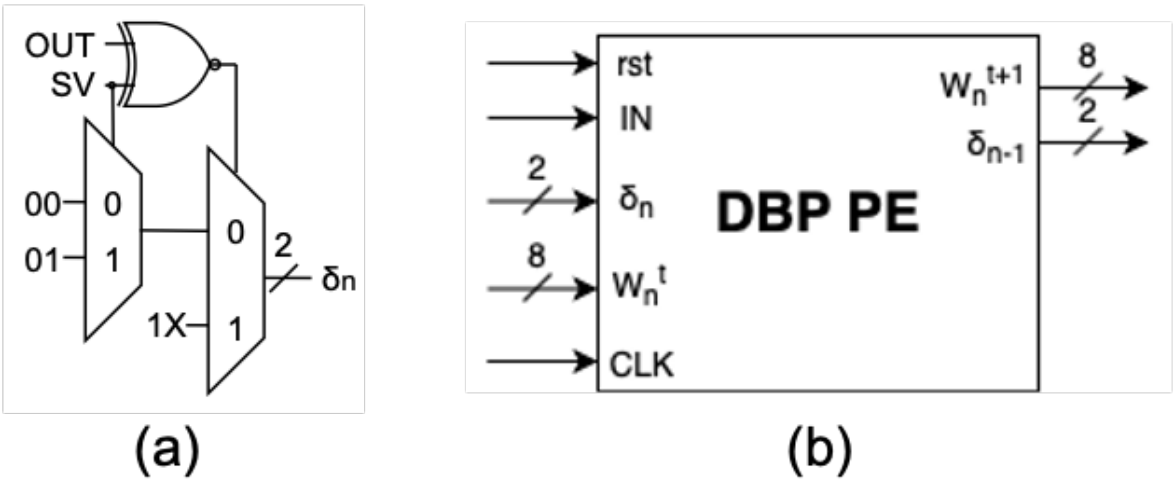

図 4.7 (a) 誤差算出回路 “1X” は “10” または “11” の任意の値を表す．(b) DBP 演算を行う論理素子 (Processing element)

は多値であるために通常減算器を用いる必要があるが，RAND チップを用いる場合には 2 値であることから XNOR によって演算可能である．ここで，図中の “1X” は “10” または “11” の内の任意の値を表す．この 2 ビットの出力値は DBP コア内ではフラグとして扱われる．2 ビットの内の上位ビットはゼロフラグビットであり，0 の時は非ゼロを，1 の時はゼロを示す．同様に，下位ビットは符号ビットであり，0 の時は生の値を，1 の時は負の値を示す．

図 4.7(b) に DBP モジュール内部にある PE を示す．PE の入力は，アキュムレータへの内部リセット信号 (rst)，各層への入力データ (IN)，各層の誤差 (δ_n)，各層の重み (W_n^t)，およびクロック (CLK) である．そして PE の出力は，パラメータの更新値 (W_n^{t+1}) と前層へと逆伝搬された誤差 (δ_{n-1}) の二つである．ここで，W_n^t と W_n^{t+1} は，それぞれ学習前と学習後のパラメータであり，n は演算対象となっている層を示す．たとえば，ネットワークが 3 つの層 (入力，中間，出力層) で構成されている場合において，中間層と出力層間のパラメータを対象として演算を行っている際には $n = 2$ となる．対象となっている層のパラメータ全てに対して演算が完了すると，n を $n - 1$ として演算対象は前の層へと移行する．図 4.8 に PE 内部の概要図を示す．PE 内部は次の二つの回路から構成されている．教師予測を行う回路 (左部) では出力層の誤差 δ^{out} とパラメータ w^t から中間層の教師信号 δ^{hid} を予測する．教師信号を予測する際に必要となる行列演算の実装は，RAND チップからのパラメータ読み出しは逐次的に行われるために時間方向に

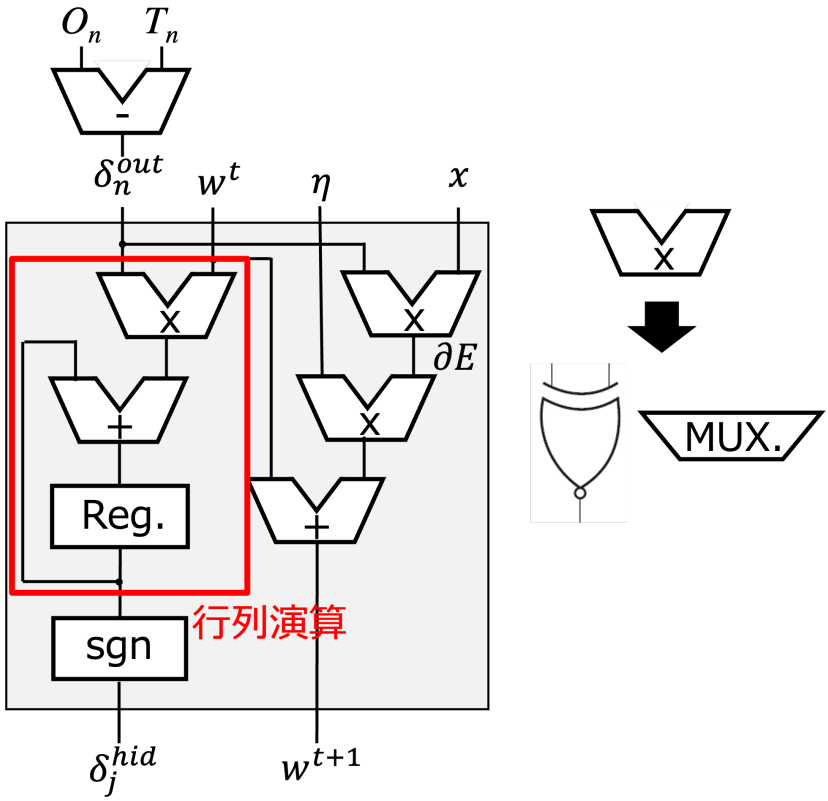

図 4.8　PE 内部の概要図．PE は出力層の誤差 δ^{out}，重み w^t，学習率 η，入力 x を受け取り，中間層の予測教師信号 δ^{hid}，重みの更新値 w^{t+1} を出力する．入力と誤差は 2 値であるために乗算器は XNOR 等の論理素子とマルチプレクサにより実装可能となる．

展開することで行っている．そのため，図中赤色で囲ったような乗算器とアキュームレータによる実装となっている．行列演算が終了するとその値は符号化回路 (sgn) により 3 値化され，これが予測教師信号 δ^{hid} となる．パラメータの更新を行う回路 (右部) では出力層の誤差と入力 x から更新されたパラメータ w^{t+1} を算出する．上記二つの回路において図中では乗算器を用いているが，パラメータと学習率を除く値は 2 値化されているために，論理素子とマルチプレクサ (MUX.) によって実装が可能である．以下に PE 内部で行われる演算回路の仔細について説明する．

PE: 教師予測と誤差伝播

図 4.9(a) は教師信号の予測を行う演算モジュールである．本アーキテクチャでは，教師信号予測 (SV_{n-1}) に伴う内積演算を時分割で実装している．つまり，複数の演算器

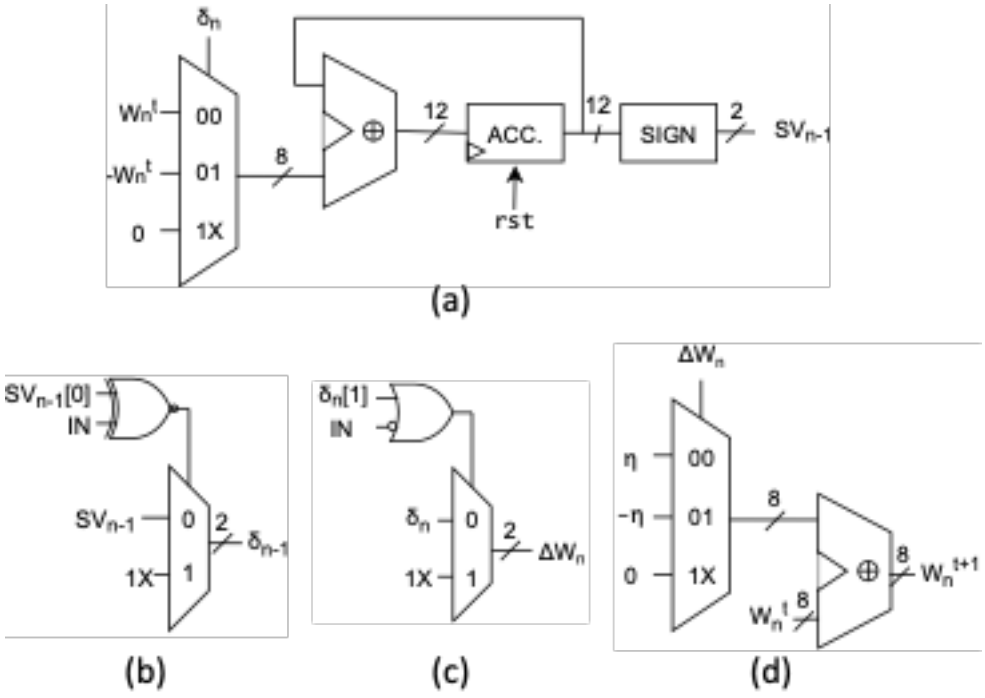

図 4.9 PE 内部の演算回路図．(a) 中間層の教師信号予測回路 (b) 中間層の誤差算出回路 (c) 更新値 ΔW 算出回路 (d) パラメータ更新回路．η は学習率を表す．

を用いて並列的に行うのではなく，加算器とアキュームレータを用意し共有することでよりエッジでの処理に向いた実装を行っている．ここで，逆伝播での誤差 (δ_n) とパラメータ (W_n^t) の乗算は，パラメータをそのまま出力する，パラメータの補数 ($-W_n^t$) を出力する，または 0 を出力するという 3 つを選択するセレクタを使用して実装可能である．SV_{n-1} の上位ビットはゼロまたは非ゼロを示し，下位ビットは 0 または 1 を示す．図 4.9(b) は予測された教師信号から誤差の算出を行うモジュールである．誤差伝播演算 (δ_{n-1}) は，演算対象としているノードへの入力 (IN) と SV_{n-1} の下位ビットとの XNOR によって実装可能である．前述の通り，この演算はノード毎に時分割に行われ，δ_{n-1} の下位ビットは符号ビットである．対象ノードの内積演算が完了次第リセット信号 (rst) がアキュムレータへと入力され，アキュームレータ内部の状態はリセットされ，次ノードの演算へと対象が移り，上記演算サイクルを再度実行する．

PE: ΔW 算出とパラメータ更新

図 4.9(c) はパラメータの更新値 (ΔW) を求める演算モジュールであり，(d) は求めた更新値からパラメータの更新を行う演算モジュールである．ΔW は対象ノードの誤差 (δ_n)

と入力 (IN) の反転との OR 演算によって算出可能である．ここで，ΔW の上位ビットはゼロフラグビットであり，下位ビットは符号ビットとなる．パラメータ更新 (W_n^{t+1}) は ΔW の値によって実行され，“00” の時にはパラメータ W_n^t に対して学習率 (η) を加算し，“01” の時には *eta* を減算する．そして，“1X” の時にはパラメータの更新は行わない．

正規化モジュール

本正規化モジュールは認識精度の減少を抑制するために実装されている．RAND チップ内において NN のパラメータは 8 ビットで保持されており，この表現力を維持するために用いられる．正規化演算はパラメータの絶対値の最大値を用いて，各パラメータを除算することにより実装される．ただし，本アーキテクチャにおいてはこの除算器は Quartus の IP カタログを用いて実装されている．エッジ AI であることを考慮に入れると，演算リソースの観点からは除算器ではなくビットシフト演算を用いて実装することが望ましいものの，除数であるパラメータの絶対値の最大値は可変であることから除算器での実装に留まっている．

Internal buffer

本アーキテクチャは学習時において，RAND チップを外部メモリとして用いるフォンノイマン型のアーキテクチャとなる．ここで，メモリとしての RAND チップは逐次読み出し，並列書き込みのデバイスである．そのため，FPGA 内部では演算素子 (PE) を一つ用意し共有することで低リソースの実装を行っている．ただし，パラメータの更新が完了する毎に逐次的に書き戻すとメモリアクセス回数が増大してしまう．従って，DBP コア内に一時保存のためのバッファを用意することで RAND チップへの書き込み回数を削減を図った．加えて，書き込み回数の削減は ReRAM デバイスの非理想的な特性の影響を削減することにも繋がる．

0 スキップ

出力層の誤差 δ_n が 0 である場合，つまり，推論結果が教師信号と一致し正しい出力が得られている場合には学習処理の全てをスキップする．図 4.10 に RAND チップを外部メモリとして使用した際に生じる読出しと書込み時の非正確性を示す．これは任意

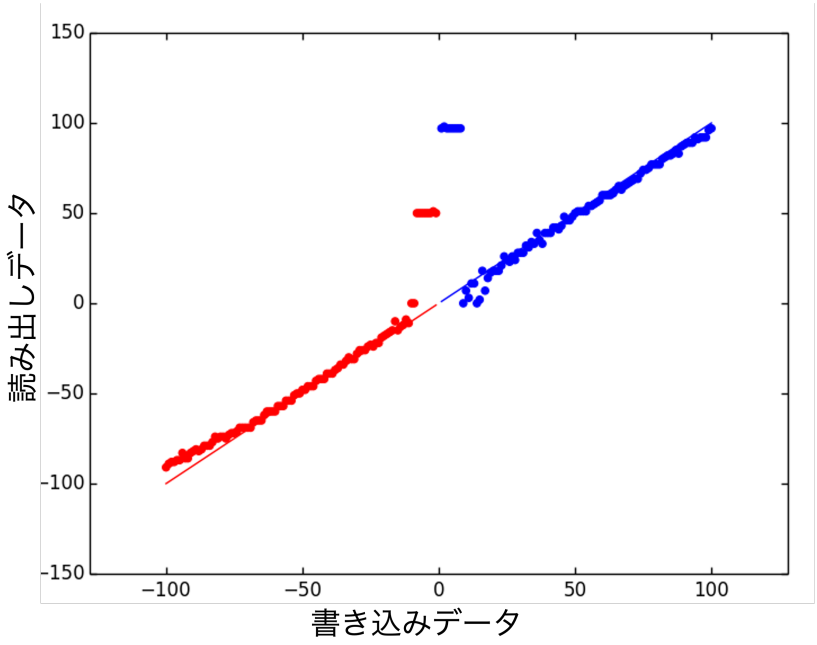

図 4.10 同一アドレス上へのデータ読出し/書込みの精度．任意のアドレスに対して −100 ~ +100 までの値を書込みその後読出したデータをプロットした．赤色が負の値，青色が正の値を示し，理想的なメモリとしての挙動を実線で示す．

のアドレスに対し，−100 ~ +100 の値を書込み，直後に読出したデータをプロットしたものである．理想的なメモリ動作の場合にはデータ点が実線上へプロットされることが期待される．一方で，メモリとして RAND チップを用いた場合には，特に-100 付近と 0 付近において実線からのズレが確認できる．また，極端に外れている値については，規定のリトライ回数を経ても書込みに失敗した結果，残存データがプロットされたものである．特に誤差が 0 の時，パラメータ更新時には読出した値をそのまま書込むという動作を行う．つまり 0 スキップがない場合には，学習の完了により理想的なパラメータが得られたとしても，0 を減算するという処理によりパラメータの値が非理想的な値への変化を誘発する．0 スキップの導入により，ReRAM デバイスの非理想的な特性の影響を削減し収束性を高めることが可能である．他にも副次的な効果として，スキップ処理によるメモリアクセス回数と演算回数の削減が期待できる．

4.3 評価

本章では，前章にて提案した DBP の改良アルゴリズムに関する評価と，FPGA へと実装した DBP アーキテクチャに関する評価を行う．以下に，まずは多クラス問題におけ

表 4.1　Iris 認識精度．試行回数は各 30 回である．

cluster size	mean	median	max	min
DBP (cluster:1)	79.7	81.7	93.3	50.0
DBP (cluster:3)	92.4	93.3	100	66.7
DBP (cluster:7)	85.28	85.60	100	83.3
DBP (cluster:9)	92.8	93.3	100	76.7
DBP (cluster:31)	93.9	96.7	100	66.7

る認識精度や回帰問題における性能というアルゴリズムの観点からの評価を行う．その後演算リソースや消費電力といったアーキテクチャの観点からの評価を行う．

4.3.1　アルゴリズム評価

アルゴリズム評価は Iris データセット，MNIST データセットを用いた識別問題に対してと，線形と非線形の関数を用いた回帰問題に対して行った．上記問題に対して，提案手法である多数決方式と部分結合方式の効果を検証した．以下にその詳細を示す．

Iris データセットを用いた評価

Iris データセットは三種類のあやめの花に関するデータセットである．入力データとして花びらとがく片についてそれぞれ長さと幅という 4 つの特徴量を持ち，出力クラス数は 3 である．この識別問題における実験条件を以下に示す．NN モデルの構造は入力層のノード数が 4，中間層のノード数が 16，出力層のノード数が 3*クラスタサイズである．学習データ数は 120 個，テストデータ数は 30 個であり，学習率は 0.01 とした．活性化関数は中間層，出力層ともにステップ関数である．ミニバッチサイズはオンライン学習を前提としていることから 1 である，表 4.1 に各クラスタサイズにおける認識精度を示す．試行回数を各 30 回として評価を行った．平均値で見ると，オリジナルであるクラスタ化を行わない DBP では認識精度が 79.7%であるが，提案手法であるクラスタ化を導入することで 90%を超える認識精度を達成することが確認できた．図 4.11 に DBP を用いた決定境界の様子を示す．ここで，入力データとして花びらとがく片の長さの 2 つの要素のみを用いている．そのため入力ノード数が前述の 4 つから 2 つへと変更されており，識別問題としての難度が上昇している．ここで，モデルを丸で示すデータのみを用いて学習しその後バツで示すデータのみを用いて決定境界を導出し

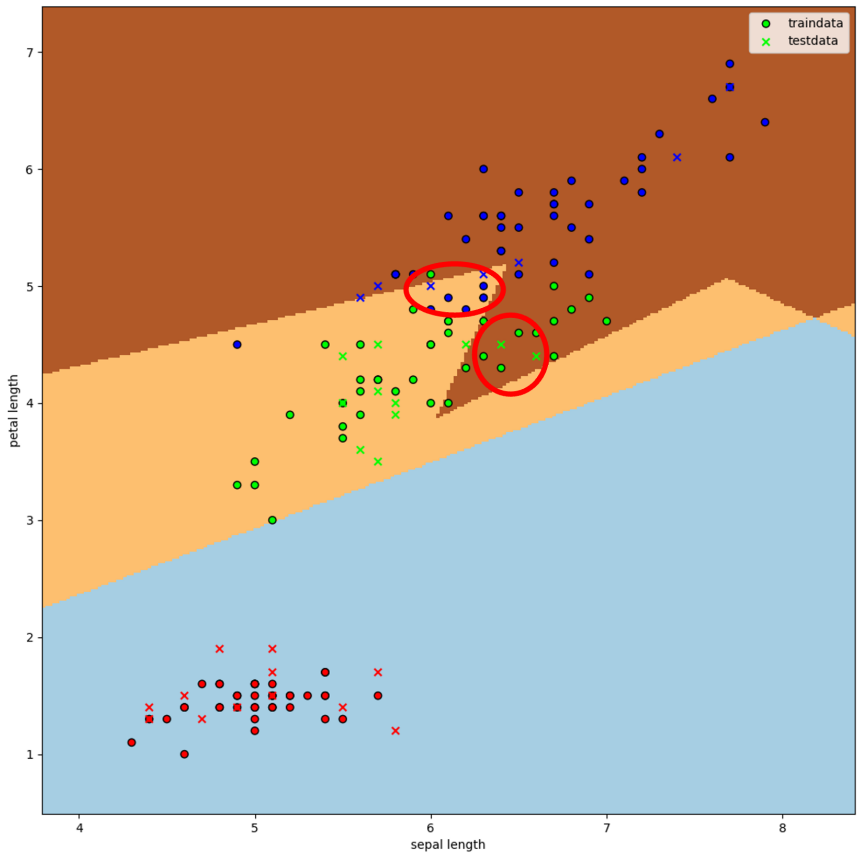

図 4.11 Iris データセットを用いた際の DBP による識別結果．グラフ化のために入力次元数を 2 としている．丸が訓練データ，バツがテストデータであり，赤色で囲った部分において誤識別となっている．

た．データ分布が離れている赤色で示すデータに対しては正しく学習できているものの，データ分布が混在している青色と緑色のデータに対しては正しく学習ができていないことが確認できた．これは DBP が潜在的に抱えている問題によるものだと考えられる．つまり，パラメータの更新が 3 値 $(0, +\eta, -\eta)$ で行われているために学習毎に決定境界が大きく変化することで収束性の低さや学習の不確かさを招いていると予想される．

回帰問題を用いた評価

次に図 4.12 に示すように回帰問題に対して提案手法の性能評価を行った．本評価においてはクラスタ化と部分結合方式を用いて行った．RAND チップは入力を 2 値で取

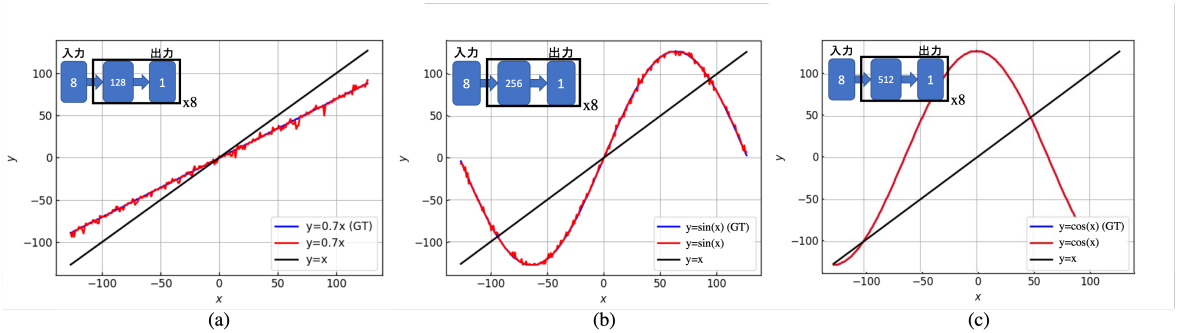

図 4.12　回帰問題に対する性能評価．各図の左上に用いたモデル構造を示している．(a) 線形回帰問題 ($y = 0.7x$)．(b) 非線形回帰問題 ($y = sin(x)$). (c) 非線形回帰問題 ($y = cos(x)$).

表 4.2　MNIST 認識精度．試行回数は各 5 回である．

	mean	median	max	min
BP (float)	93.56	93.10	94.90	92.80
BP (fixed)	92.90	91.58	92.90	90.10
DBP (original)	46.96	46.80	50.00	44.80
DBP (cluster:1)	75.00	75.50	77.30	71.70
DBP (cluster:3)	84.32	84.90	86.20	82.50
DBP (cluster:5)	85.60	85.40	87.60	83.20
DBP (cluster:31)	78.74	79.40	80.10	76.10

るため，入力層のノードを 8 個とすることで 8 ビットの値を擬似的に表現している．中間層のノード数は図中左上に示すモデル構造に記載している通りである．出力層のノード数は 1 個であるが，部分結合となるサブネットワークは 8 個存在するためにモデル全体として見た際の出力ノード数は 8 個となる．こちらも入力層と同様に 8 ビットを擬似的に表現している．図中赤色で示している回帰の性能が高くないのは中間層の数による表現力の低さと，この擬似的なビット表現に起因すると考えられる．

MNIST データセットを用いた評価

MNIST データサイズは 14x14 に縮小したものを用いた．ネットワークの各ノード数は入力層 196，中間層 64×10，出力層 10× クラスタサイズである．活性化関数は中間層，出力層ともにステップ関数である．ミニバッチサイズはオンライン学習を前提と

していることから 1 である，比較対象として RAND チップは重みをアナログ値で持っていることから浮動小数点方式での BP 実装に加え，エッジベースのシステムであることから 16 bit に量子化した固定小数点方式での BP 実装を用いた．比較対象のネットワークのノード数は入力層 196，中間層 64，出力層 10 であり，活性化関数はシグモイド関数である．表 4.2 に認識精度を示す．この認識精度は学習データ数が 6000，テストデータ数は 1000 である時に BP は 30epoch，DBP は 6epoch 学習をしたものである．クラスタと部分結合の導入によって識別精度が約 47%から約 86%にまで向上できたことが確認できた．クラスタサイズの増加は冗長性の確保につながり，汎化性能向上に結びつくことから識別精度の向上に寄与していると考えられる．一方であまりに大きなクラスタサイズでは教師信号の予測難度の増加を引き起こすために識別精度の低下を招いていると予測される．また，DBP の epoch 数を 30 にした場合に 6epoch の時より識別精度が低下していることが確認されたが，これは更新値の大きさによるものだと考えられる．DBP により得られる更新値は学習率の値そのものとなるために学習後期になると更新値が大きくなってしまうと予想できる．この問題の解決のためには学習率に減衰項を導入することが挙げられる．

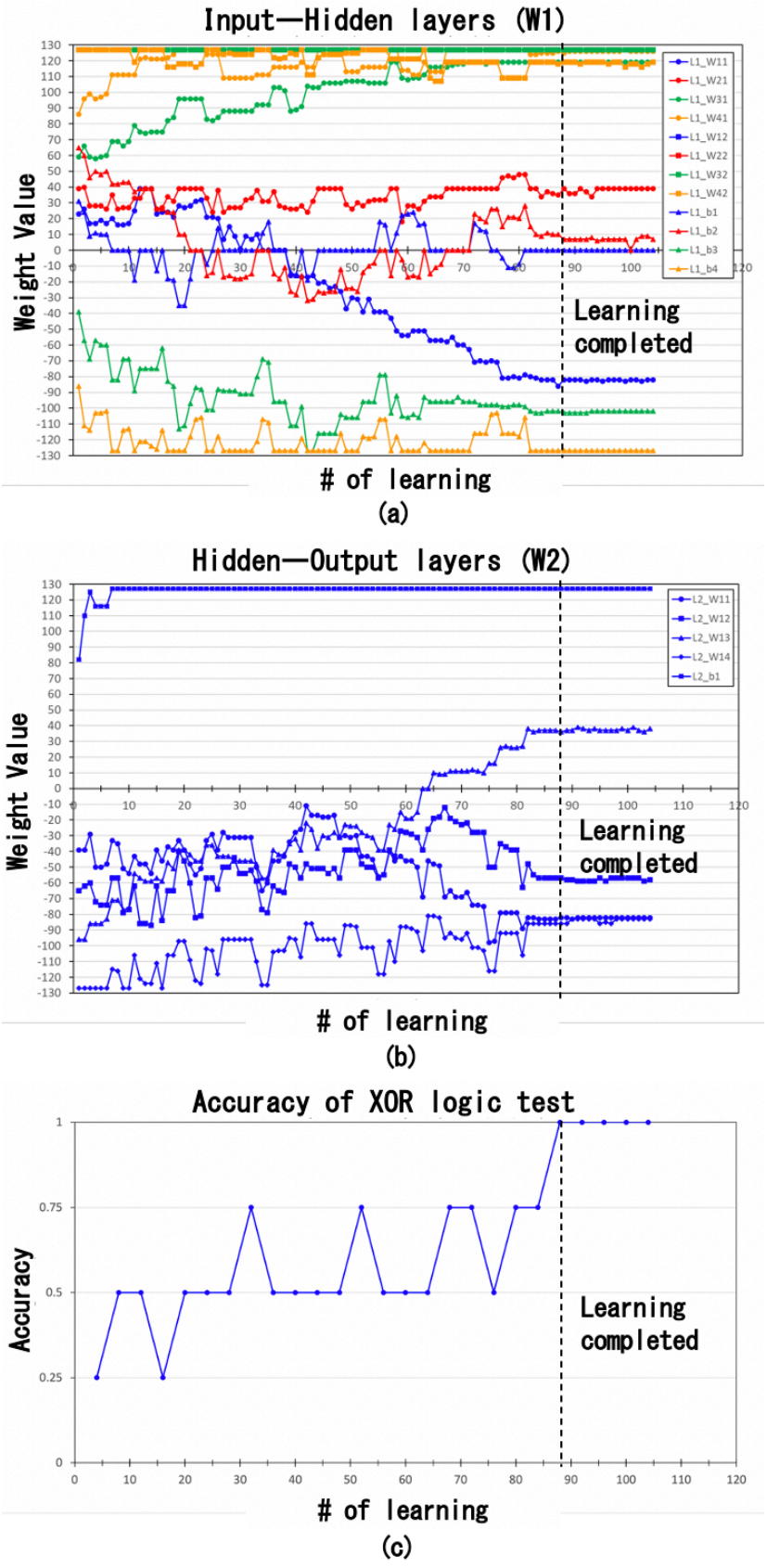

図 4.13　XOR 学習時のパラメータ変化の推移．(a) 入力層と中間層間のパラメータ (W1) 変化の様子．(b) 中間層と出力層間のパラメータ (W2) 変化の推移．(c) XOR 学習の認識精度の推移の様子

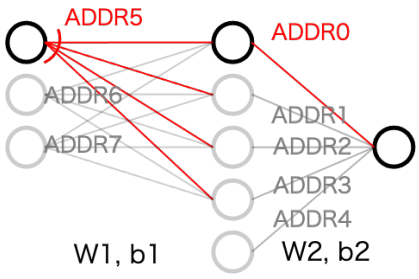

図 4.14　NN モデルと RAND チップに対するアドレスマップ．各アドレスは最大 16 個の重みを格納可能であり，赤線で示すように各ノードに対して各アドレスが割り振られる．

4.3.2　アーキテクチャ評価

図 4.13 に事前実験の結果を示す．この事前評価では推論は RAND チップを用いて行い，学習は Python により実装された DBP を用いて行った．図 4.13(a) に学習により変化する入力層と中間層間の重み (W1) の変化の推移を示す．同様に，図 4.13(b) には中間層と出力層間の重み (W2) の変化の推移を示す．図 4.13(c) には認識精度の変化の推移を示す．およそ 90 回の学習で収束することが確認でき，そのタイミングにおいて各重みの変化も安定することも確認できた．この事前実験の結果，RAND チップを実際に用いた場合でも DBP アルゴリズムがオンライン学習に効果的であることが確認できた．

本提案アーキテクチャの評価の大きな目的の一つとして，前節で行ったシミュレーションによる事前実験結果との比較がある．そのために本評価においては多クラス分類や回帰問題での評価ではなく，XOR の識別問題を用いた．図 4.14 に本評価において用いた NN モデルとそのパラメータを RAND チップへマッピングした際のアドレスを示す．W1 のアドレスは ADDR5 と ADDR6 であり，W2 のアドレスは ADDR0 から ADDR5 である．加えて入力層と中間層はそれぞれバイアスノード (b1, b2) を備えており，その重みのアドレスは ADDR7 と ADDR4 で示されている．各アドレスは最大 16 個の重みを保持することが可能である．例えば，W1 の ADDR5 では 16 個のうち 4 つが使用されている．他の条件として，ミニバッチサイズは 1 であり，各エポックにおいてデータの並びはシャッフルされている．重みとバイアスは 8 ビットの符号付き整数である．最適化手法は慣性項のない SGD であり，学習率は 0.1 である．ただし RAND チップ内部の階調は正負それぞれ 128 であることから，実際の更新値としての値は 13

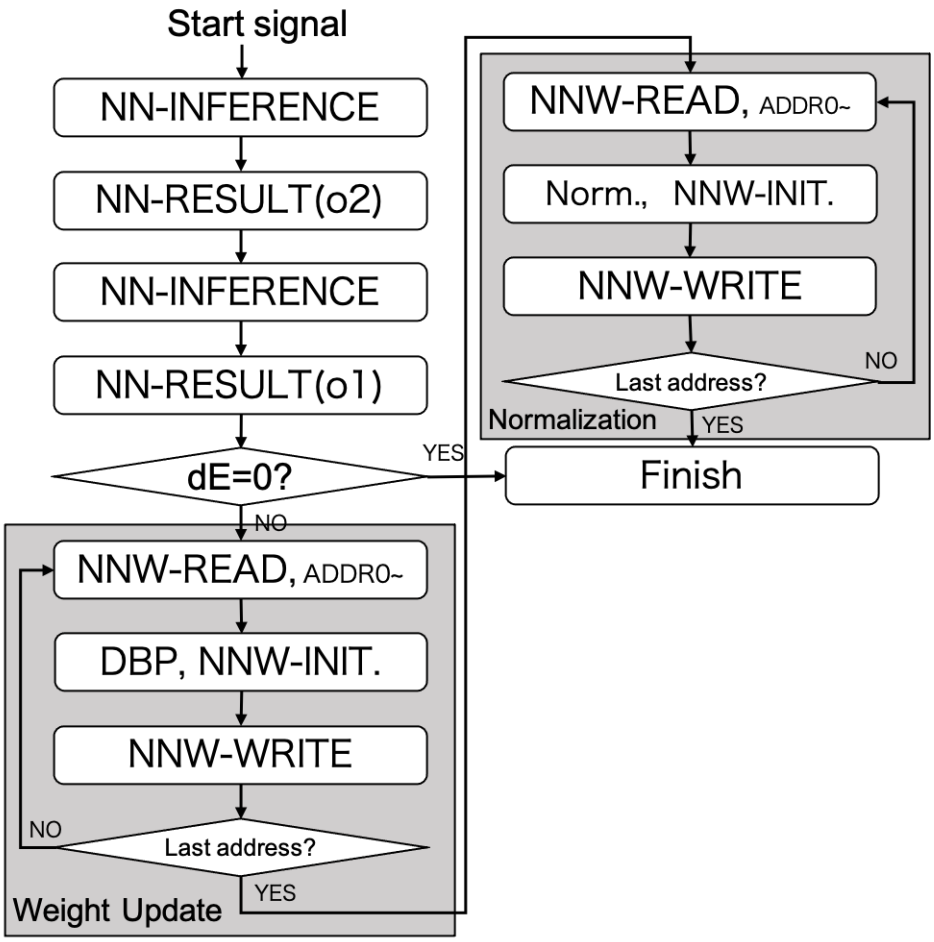

図 4.15　1回の学習あたりの提案アーキテクチャの実行フローチャート．推論結果が正しい場合には0 skipにより終了状態へと遷移する．学習に関する各演算は重み毎に行われ，RANDチップへの書き込みはアドレス毎に行われる．

である．

実行フロー

図4.15に1回あたりの学習の開始から終了までのフローチャートを示す．PythonやArduino等の外部機器から与えられる学習開始信号をFPGA内部で検知し次第，推論処理コマンド(NN-INFERENCE)をRANDへと発行する．ここで，RANDチップを動作させるためにはいくつかのコマンドをRANDへと発行する必要がある．各NN-XXXXコマンドは，推論処理の実行のようなNN関連のコマンドであることを示す．同様に，

NNW-XXXX コマンドは RAND チップへのパラメータの書き込みや読み出しに関連したコマンドであることを示す．例えば NNW-WRITE コマンドは，パラメータを格納しているアドレスと 8 ビット符号付き整数値のデータを設定して実行し，最大 16 個までのデータをアナログ抵抗値として対象アドレスのメモリセルに書き込む．

学習時には各層の出力値 (o1, o2) が必要となるものの，出力値を得るコマンドである NN-INFERENCE では任意の層の出力値のみが得られる．つまり，複数層の出力値を得るためには，NN-INFERENCE コマンドを層の数だけ実行する必要がある．そのため，フローチャート上では NN-INFERENCE が 2 回に渡り実行されている．ここで，推論処理の結果が正しい (dE=0，o2 と教師信号が一致する) 場合には，フローチャート上の後続の処理である DBP 演算や正規化処理，パラメータの更新と RAND チップへの書き戻しといった処理はスキップされ終了状態へと移行する．反対に，推論処理の結果が間違っている場合ではフローチャート通りの処理が行われ，パラメータの更新が行われる．パラメータの更新は，中間層と出力層間のパラメータ (W2、b2) から開始される．この層間の更新が終了すると，入力層と中間層間とのパラメータ (W1、b1) 更新へと処理が移行する．パラメータの読み出しは各層のノード毎に個別に実行される．DBP 演算の処理は読み出したパラメータを用いて行い，更新されたパラメータは RAND チップへと書き戻される．1 つのアドレスに対する一連の処理ステップが終了すると，アドレスカウンタが加算され次のアドレスに対して同様に処理ステップが繰り返される．最終アドレスである b1 (ADDR7) のパラメータが更新されると，正規化処理の実行へと移行する．正規化処理も同様に ADDR0 から処理が始まる．読み出し，正規化，および書き込みプロセスは各アドレス毎に実行される．DBP 演算の時と同様に指定されたアドレスの処理が完了すると，アドレスカウンタが加算され，最終アドレスまで正規化処理が繰り返される．最終アドレスに達すると，終了状態へと移行する．

ここで，各 RAND チップに対して発行する NNW-READ, NNW-INIT 等の各実行コマンドは同時に実行することはできない．また，学習時には RAND チップを外部メモリとして扱う都合上，FPGA と RAND チップ間の通信が実行時間上のボトルネックとなる．そのため学習処理全体の実行時間は RAND チップの処理速度に制限される．従って，DBP モジュールを並列化しただけでは実行時間の短縮は望めなく，並列化による高速化を得るためには RAND チップを複数個用いるか，RAND チップ内に DBP モジュールを実装する必要があると考えられる．

表 4.3　[86] の Table 1 に示されている RAND チップのキーパラメータ.

Process technology	Synapse array size	Device material
180 nm	2M weights	Tantalum Oxide (TaOx)
Chip area	**TOPS/W**	**Power**
12.6 mm^2	20.7	15.8 mW

表 4.4　Quartus により導出されたロジック使用率.

	PE in DBP core	DBP core	Whole architecture
# of LEs	99	330	3027
# of registers	23	104	1141
# of memory bits	0	0	128
# of DSPs	0	0	0

ハードウェアリソース

表 4.3 にプロセスサイズ等の RAND チップ [86] のキーパラメータを，表 4.4 に本提案アーキテクチャにおけるロジック使用率を示す．使用率は Quartus を用いたコンパイル結果から得られた．本提案アーキテクチャは，前述した並列性に関する考察から DBP コア内部には PE が一つのみ有しており，BRAM (Block RAM) や乗算器の使用数は最小限な構成となっている．使用 FPGA チップの論理素子数は 25,000 であることから，アーキテクチャ全体でも使用率が約 10%であり，学習部に関しては制御部を含めても約 1%，演算回路のみでは約 0.4%の使用率となっている．Quartus power analyzer による消費電力評価では DBP コアの静的消費電力は 5.58 mW，動的消費電力は 0.96 mW であった．ここで，消費電力算出にあたり使用されたトグル率は実際に学習処理を行った結果から得られたものを利用した．一回の学習に要したクロック数は 228 clk であり，50 MHz 駆動であることから 4.56 us となる．よって，動的消費電力から DBP コアが一回の学習で消費するエネルギーは 4.38 nJ である．演算は乗算と加算とがパラメータの数だけ行われ 34 OP (3*4*2+5*1*2) となることから，演算効率は 7.77 GOPS/W であり，MAC 演算あたりのエネルギーは 0.258 nJ/MAC である．ここで，学習時には非ノイマン型アーキテクチャとなる本提案において RAND チップは逐次読出し並列書込みの外部メモリとして扱われる．また，処理速度は RAND チップとの通信により律速されていることから，仮に PE の並列性を高めたとしてもその恩恵は得られない．そのため，モデルサイズが増加した場合には実行時間のみが増大し，使用リソース量については

表 4.5　提案アーキテクチャ（CIM 推論，ノイマン型学習）と同じ条件を持つデバイスとの比較．

	This work	B. Crafton[46] ISLPED2019	M. Giordano[45] VLSI’21
Task (Model size)	MLP (2-4-1)	MLP (sim.) (784-800-10)	ResNet-18
Training	DBP from scratch	BP from scratch	LRT for incremental
Freq.	50 MHz	NA	200 MHz
Weight capacity	2 MB	NA	2 MB
Precision (weight)	INT8	INT8	INT8
Precision (Activation)	binary	INT8	FP16
Energy	4.38 nJ/ sample	750 nJ/NA	4.5 mJ/sample
Time	4.56 us/ sample	2.6 us/NA	44 ms/sample
Power	0.96 mW/ sample	288 mW/ NA	102 mW/sample
TOPS/W	7.77×10^{-3}	1.96	2.2 (all included)

スケーラビリティが確保できていると考えられる．

表 4.5 に本提案アーキテクチャと他の手法との比較結果を示す．比較対象としては，パラメータの保存に ReRAM の NVM を使用しており，学習は CIM デバイス内に実装されていないノイマン型のアーキテクチャ構成を持つ．なお，B.Crafton[46] の評価は，シミュレーション結果に基づくものである．本提案手法と B.Crafton[46] の手法はエッジデバイス上のみでのゼロからの学習に焦点を当てている．一方で，M. Giordano[45] の手法はインクリメンタル学習にのみ焦点を当てており，事前に学習されたパラメータを用いることが前提となっている．つまり，エッジデバイスのみでのゼロからの学習には対応していない．本提案システムにおいて 1 回の学習にかかる消費電力は 100 mW 以下であると見積もることが可能であり，これは [85] と [45] の 100 mW 以上の消費電力より少ないものである．提案学習コアは学習処理において 7.77 GOPS/W の演算効率を達成した．これは，FPGA と RAND チップ間の通信ボトルネックに依るところが大きく，将来的に DBP コアを RAND チップ内に実装することにより解決可能であると考えている．

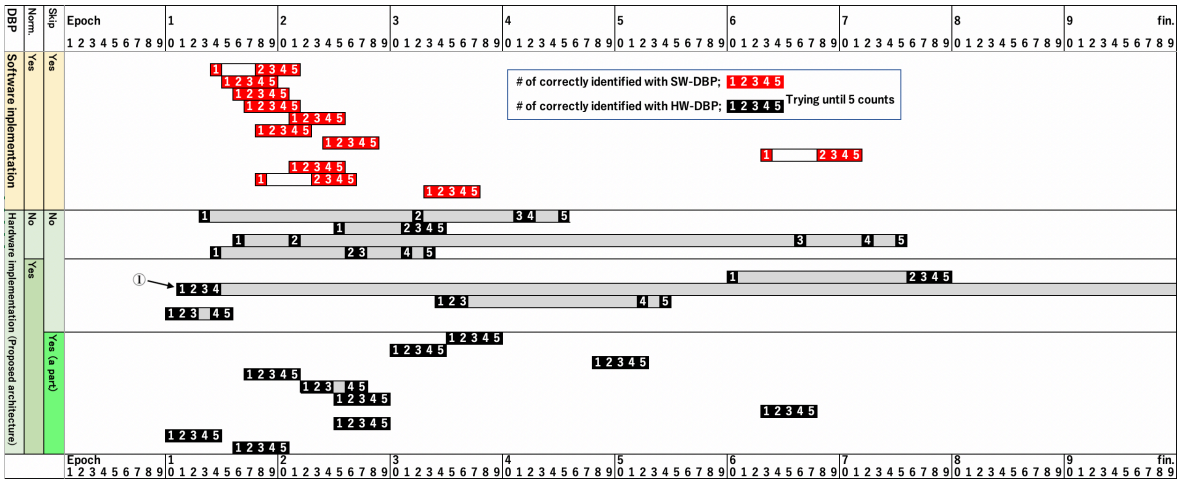

図 4.16　XOR 論理の学習経過と結果．赤色で示すデータはソフトウェアによる学習結果を，黒色で示す結果はハードウェアによる学習結果をそれぞれ示す．正しい推論結果が 5 回得られるまでの様子を示し，推論結果と教師データが一致した際のエポックと累積回数がプロットされている．①に最悪のケースを示す．

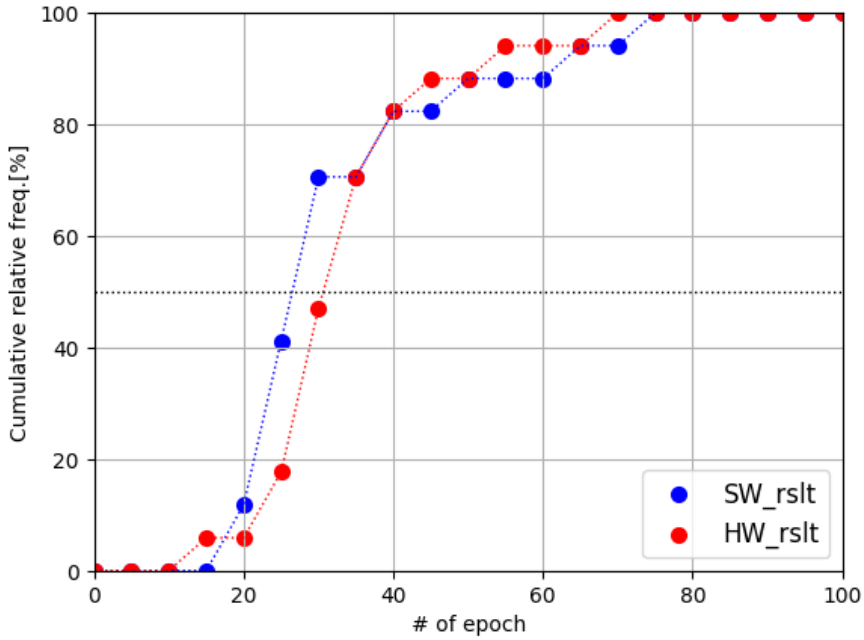

図 4.17　図 4.16 のソフトウェアとハードウェアによる結果の累積度数分布．青色がソフトウェアの赤色がハードウェアの分布を示す．両者ともに試行回数の内 8 割が 40epoch 内に収束しており，ソフトウェアとハードウェアの動作一致が確認できた．

シミュレーションとの比較

図 4.16 に各条件の XOR 学習の結果を示す．各条件は図中左部に示しているように FPGA と RAND の非一体型アーキテクチャと提案手法である一体型アーキテクチャ，ソフトウェア実装 DBP とハードウェア実装 DBP，正規化処理の有無，スキップ処理の有無となっている．各条件で複数回試行しており，それぞれの結果をプロットした．実験では 1 エポックあたり推論処理が 4 回行われる．4 回全てに対して正しく識別できた回数をカウントし，そのカウント値が累積 5 回となるまでに要したエポック数を求めた．赤色で示された結果は予備実験で行ったシミュレーションであるソフトウェア実装 DBP を表す．これらの結果は約 30 エポックまでに収束し，また累積 5 回まで連続している傾向を確認できた．ここで，識別成功判定の 1 回から 2 回までに間があるものについては判定を python 側で行っているために UART 通信による影響が出ていると考えられる．通信エラーにより本来の出力値と異なるものを検知した場合，望ましくない重みへの更新が発生する可能性が存在する．

一方，黒色で示された結果は提案手法であるハードウェア実装 DBP を表す．上から順に DBP のみの実装，DBP と正規化の実装，DBP と正規化そしてスキップ処理の実装となっている．DBP のみの場合では各成功判定から次の判定まで連続していないことが確認できた．原因として正規化処理の未実装によるものが大きいと考え，続いて正規化処理の実装を行い実験を行った．正規化処理の実装により成功判定の連続した結果が得られる傾向が高くなったことが確認できた．しかし，①に示すように動作の不安定な面を残しており，これは本来値を書き換えてはいけないパラメータ ($\Delta W = 0$) まで，読み出し/書き込みの際に書き換わってしまっているためであると考えられる．従って，推論結果が正しい時には重み更新と正規化をスキップする処理を追加することで解決を図った．また，スキップ処理は動作速度の向上や消費電力の削減という観点からも重要なものである．最終的な結果として累積 5 回まで連続して得られる頻度が多くなり，ソフトウェア実装による DBP と同等の精度が得られたことが確認できた．依然として成功判定間が連続していないものについては現在のスキップ処理はアドレス毎に行っており，DBP 演算において ΔW が 0 になった重みに対して，個別に更新のスキップ処理が行えていないためだと考えられる．現在の RAND チップは同時にパラメータを書き込む方式であるために，書き込み時の ReRAM デバイスの非理想的な特性により，学習の不安定性が引き起こされていると予想される．これを解決するための個別スキップの実装には RAND チップ自体の更なる改良が必要となる．図 4.17 に図 4.16 の最上部のソフトウェアの結果と最下部のハードウェアによる結果についてそれぞれの

累積度数分布を示す．DBPをソフトウェアで実装し学習した結果を青色で示し，提案手法となるDBPをハードウェアで実装し学習した結果を赤色で示す．両者ともに試行回数の内8割が40epoch内に収束しており，ソフトウェアとハードウェアの動作一致が確認できた．よって，アルゴリズム評価で用いたIrisデータセットやMNISTデータセット，回帰問題についても，本提案アーキテクチャが効果的であることが期待できる．ここで，予備実験，提案手法の両方の結果において成功判定の1回目までに要するエポック数が多いものについてはミニバッチサイズが1であることや更新値が三値(今回の場合+13，-13，0)に限定されていること等が考えられる．これらの解決にあたり，ミニバッチ処理を時間方向に展開することで対応することや慣性項の導入などのアルゴリズムの改良が挙げられる．

4.4　結論

本章では，アナログCIMデバイスに向けたオンライン学習アルゴリズムとしてDBPアルゴリズムの改良と，DBP演算アーキテクチャの構築を行いFPGAへと実装した内容について述べた．不揮発性のCIMデバイスはその特性から低電力にAI演算を可能とするデバイスとして注目されており，これに学習機能を搭載することはエッジAIの可用性を大幅に広げ，次世代の情報社会実現に寄与すると考えられる．従来のDBPアルゴリズムでは出力ノード数の増加に伴う教師信号の予測精度低下という問題点があった．これを解決するための手法として，一つのクラスに対して多数決により出力を得ることで汎化性能の向上を狙うクラスタ化の導入と，他クラスの誤差情報による予測精度の低下を避ける部分結合の導入を行いその説明を行った．本提案により従来のDBP実装では約47%だったMNISTの識別精度が約86%に向上することが確認できた．

提案FPGAアーキテクチャは，RANDチップの制御と学習機能であるDBPを実行するモジュールとで構成されている．事前に行っていたソフトウェアによるシミュレーションと同等の性能が得られていることを確認した他，パラメータ更新時のRANDチップへの書き戻しに改良の余地が残されていることも確認できた．提案アーキテクチャは100 mW以下での学習を可能にしており，特に学習コアにおいては動作周波数が50 MHzの場合で，0.96 mWと7.77 GOPS/Wの演算効率を達成したことが確認できた．

参考文献

[1] M. Tan and Q. Le. “EfficientNetV2: Smaller Models and Faster Training”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. **139**. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 10096–10106.

[5] A. Vaswani, N. Shazeer, N. Parmar, et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. **30**. Curran Associates, Inc., 2017.

[6] J. Kaplan, S. McCandlish, T. Henighan, et al. “Scaling Laws for Neural Language Models”. In: *arXiv preprint arXiv:2001.08361* (2020). arXiv: 2001.08361.

[9] D. Silver, J. Schrittwieser, K. Simonyan, et al. “Mastering the Game of Go without Human Knowledge”. In: *nature* **550**.7676 (2017), pp. 354–359.

[15] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[16] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). arXiv: 1412.6980.

[18] T. Oohori, H. Naganuma, and K. Watanabe. “A New Backpropagation Learning Algorithm for Layered Neural Networks with Nondifferentiable Units”. In: *Neural Computation* **19**.5 (2007), pp. 1422–1435. DOI: 10.1162/neco.2007.19.5.1422.

[30] M. Courbariaux, I. Hubara, D. Soudry, et al. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. 2016. DOI: 10.48550/ARXIV.1602.02830.

[35] S. Zhou, Y. Wu, Z. Ni, et al. *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*. 2016. DOI: 10.48550/ARXIV.1606.06160.

[42] 河野 和幸. “不揮発性メモリを用いた AI チップの実装技術”. In: 電子情報通信学会誌*, The journal of the Institute of Electronics, Information and Communication Engineers* **103**.5 (May 2020), pp. 543–548.

[45] M. Giordano, K. Prabhu, K. Koul, et al. “CHIMERA: A 0.92 TOPS, 2.2 TOPS/W Edge AI Accelerator with 2 MByte on-Chip Foundry Resistive RAM for Efficient Training and Inference”. In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492347.

[46] B. Crafton, M. West, P. Basnet, et al. “Local Learning in RRAM Neural Networks with Sparse Direct Feedback Alignment”. In: *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE. 2019, pp. 1–6.

[74] S. Han, H. Mao, and W. J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. 2015. DOI: 10.48550/ARXIV.1510.00149.

[75] J. Choi, P. I.-J. Chuang, Z. Wang, et al. *Bridging the Accuracy Gap for 2-Bit Quantized Neural Networks (QNN)*. 2018. DOI: 10.48550/ARXIV.1807.06964.

[76] D. Zhang, J. Yang, D. Ye, et al. *LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks*. 2018. DOI: 10.48550/ARXIV.1807.10029.

[77] T. Kaneko, K. Orimo, I. Hida, et al. “A Study on a Low Power Optimization Algorithm for an Edge-AI Device”. In: *Nonlinear Theory and Its Applications, IEICE* **10**.4 (2019), pp. 373–389. DOI: 10.1587/nolta.10.373.

[78] J. Lee, J. Lee, D. Han, et al. “7.7 LNPU: A 25.3TFLOPS/W Sparse Deep-Neural-Network Learning Processor with Fine-Grained Mixed Precision of FP8-FP16”. In: *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2019, pp. 142–144. DOI: 10.1109/ISSCC.2019.8662302.

[79] B. Fleischer, S. Shukla, M. Ziegler, et al. “A Scalable Multi- TeraOPS Deep Learning Processor Core for AI Trainina and Inference”. In: *2018 IEEE Symposium on VLSI Circuits*. 2018, pp. 35–36. DOI: 10.1109/VLSIC.2018.8502276.

[80] D. Han, J. Lee, J. Lee, et al. “A 1.32 TOPS/W Energy Efficient Deep Neural Network Learning Processor with Direct Feedback Alignment Based Heterogeneous Core Architecture”. In: *2019 Symposium on VLSI Circuits*. 2019, pp. C304–C305. DOI: 10.23919/VLSIC.2019.8778006.

[81] C. Kim, S. Kang, D. Shin, et al. “A 2.1TFLOPS/W Mobile Deep RL Accelerator with Transposable PE Array and Experience Compression”. In: *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2019, pp. 136–138. DOI: 10.1109/ISSCC.2019.8662447.

[82] S. Kang, D. Han, J. Lee, et al. “7.4 GANPU: A 135TFLOPS/W Multi-DNN Training Processor for GANs with Speculative Dual-Sparsity Exploitation”. In: *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2020, pp. 140–142. DOI: 10.1109/ISSCC19947.2020.9062989.

[83] Y. Wang, Y. Qin, D. Deng, et al. “A 28nm 276.55TFLOPS/W Sparse Deep-Neural-Network Training Processor with Implicit Redundancy Speculation and Batch Normalization Reformulation”. In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492420.

[84] J. Lee, S. Kim, S. Kim, et al. “OmniDRL: A 29.3 TFLOPS/W Deep Reinforcement Learning Processor with Dualmode Weight Compression and on-Chip Sparse Weight Transposer”. In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492504.

[85] J. Lee, J. Kim, W. Jo, et al. “A 13.7 TFLOPS/W Floating-Point DNN Processor Using Heterogeneous Computing Architecture with Exponent-Computing-in-Memory”. In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492476.

[86] R. Mochida, K. Kouno, Y. Hayata, et al. “A 4M Synapses Integrated Analog ReRAM Based 66.5 TOPS/W Neural-Network Processor with Cell Current Controlled Writing and Flexible Network Architecture”. In: *2018 IEEE Symposium on VLSI Technology*. 2018, pp. 175–176. DOI: 10.1109/VLSIT.2018.8510676.

[87] 金子 竜也, 山岸 善治, 百瀬 啓, et al. “アナログ AI チップのオンライン学習に向けた改良型デジタルバックプロパゲーション法の提案”. In: 第 *30* 回日本神経回路学会全国大会. 2020.

第5章　ハードウェア指向対数量子化最適化手法

5.1　導入

人工知能 (AI)，特に機械学習のニューラルネットワークの分野において深層学習を代表に種々のタスクにおいてその性能の高さを示している [1, 5, 6, 9]．本分野における発展は GPU や TPU[25] といった大規模なクラウドベースの AI によるところが大きい．そのため，AI の恩恵はネットワーク越しのサイバー空間上に半ば限定されており，我々の日常生活である実空間へと還元されていないのが現状である．これを解決するために近年では我々の身の回りの端末であるエッジデバイス上で AI 処理を行う研究が活発になっている [88, 89, 90]．ただし，これらのエッジ AI 研究は推論機能を有するのみであり学習機能についてはサポートされていない．将来的な社会システムにおいて，セキュリティや環境の変化に対応するためには学習によるモデル更新の必要性が高まると考えられ，エッジ AI に向けた学習手法の研究についても取り組まれてきている [82, 45].

ここで，ニューラルネットワークの学習には最適化手法と誤差逆伝播法 (BP) とに関する研究が存在する．前者は得られた誤差を基にパラメータ更新を行うための手法であり，確率的勾配降下法 (SGD)[15] やこの発展である Adam[16] 等に代表される，後者はあらかじめ設定された誤差関数から計算された誤差をモデル全体へと伝播させる手法であり，BP の他に Direct Feedback Alignment (DFA)[91] や Difference Target Propagation[92] 等が存在する．特にエッジ AI では [35] を始めとした BP や SGD といった従来提案されている手法にビット制限を行い固定小数点化する手法が主流である．他にも，DFA をパイプライン化しエッジ上でも高速な学習を可能とすることを目的とした研究 [93] や，DFA をスパースにすることで不揮発性のコンピューティングインメモリ上で低電力に学習を行うことを目的とした研究 [46] が存在するものの，最適化手法は SGD である．両研究ともにエッジ AI の学習に関する研究であるが，対象としているのは最適化手法ではなく誤差の伝播に関するものである．SGD より高度な最適化手法を用いるこ

Algorithm 2 Holmes Optimizer

Initialize θ_0, $m_0 \leftarrow 0$, $t \leftarrow 0$

while θ_t not converged

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta L(\theta_{t-1})$

$m_t \leftarrow 2^{\lfloor \log_2(\mathrm{m_{t-1}}) \rfloor} + \eta g_t$

Update

$\theta_t \leftarrow \theta_{t-1} + m_t$

とは学習回数の削減に繋がり，高速な学習を可能とする．また，パラメータ更新に伴うメモリアクセス回数の削減も可能とし，メモリアクセスが消費電力の約 50%を占める [28] ことからも低消費電力化に大きく寄与する．しかし，演算リソースの観点から Adam 等の高度な最適化手法の実装について課題が残っている．そこで，本論文はこの解決に向けて以下の提案を行う．

- MomentumSGD[94] のモーメンタム項の計算を対数量子化によって実現する手法である Holmes について説明する．Holmes は (1) 対数量子化によるモーメンタム項の情報量削減と (2) メモリアクセス回数削減を可能とする高速な学習とそれによる消費電力削減という 2 つの利点を持つ．
- ベンチマーク関数と MNIST を用いて Holmes の性能評価を行う．比較対象は固定小数点方式を採用した SGD と MomentumSGD である．この二つに比べ，Holmes は高速な収束性と高い認識精度を達成することを示す．
- Holmes を用いた学習で発生するパラメータの発散とその解決法について説明する．Holmes は固定小数点化された値に対数量子化を施すことで発生する量子化誤差を活用している．量子化による誤差が発生しない場合では悪影響を与える可能性が生ずるがその回避方法を示す．

5.2 提案手法

本章では提案最適化手法である Holmes (Hardware-oriented Logarithmic Momentum Estimation) についての説明を行う．また，本章では簡単のため整数部を 3bit，小数部を 2bit とした固定小数点方式を基に説明を行う．

Algorithm 3 MomentumSGD Optimizer

Initialize θ_0, $m_0 \leftarrow 0$, $t \leftarrow 0$

while θ_t not converged

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta L(\theta_{t-1})$

$m_t \leftarrow \beta m_{t-1} + \eta g_t$

Update

$\theta_t \leftarrow \theta_{t-1} + m_t$

5.2.1 Holmes: Hardware-oriented Logarithmic Momentum Estimation

本提案手法である Holmes はニューラルネットワークの学習に用いられる手法の一つである MomentumSGD を基にした最適化アルゴリズムである．Holmes ではモーメンタム項の計算に必要となる小数点の乗算を，固定小数点と対数量子化の組み合わせによる量子化誤差で代替した．これにより，乗算器の削減とモーメンタム項の情報量削減によるメモリ容量の削減を可能とした．本提案手法と MomentumSGD とを用いた場合のパラメータ更新方法を Algo. 2 と Algo. 3 に示す．ここで，θ はニューラルネットワークのパラメータ，t は時刻，m はモーメンタム項，g_t は損失関数の勾配，$L(\theta)$ は損失関数であり，η と β はそれぞれハイパーパラメータである．一般に β の値は 0.9 とすることが多いが，本論文では実装の対象を演算リソースの乏しいエッジデバイスとしていることからこの値を 0.75 として実験と評価を行った．これはビットシフトと減算により計算可能であり，乗算器を用いる必要がないためである．また，Holmes においてはパラメータとモーメンタム項，損失関数の勾配の各値は固定小数点化を前提としている．上述の通り，MomentumSGD と Holmes との違いは βm_{t-1} の部分が $2^{\lfloor \log_2(m_{t-1}) \rfloor}$ と変化した点のみであり，この部分を対数量子化と定義した．対数量子化による乗算の代替についての仔細を以下に記述する．

固定小数点方式に対する対数量子化

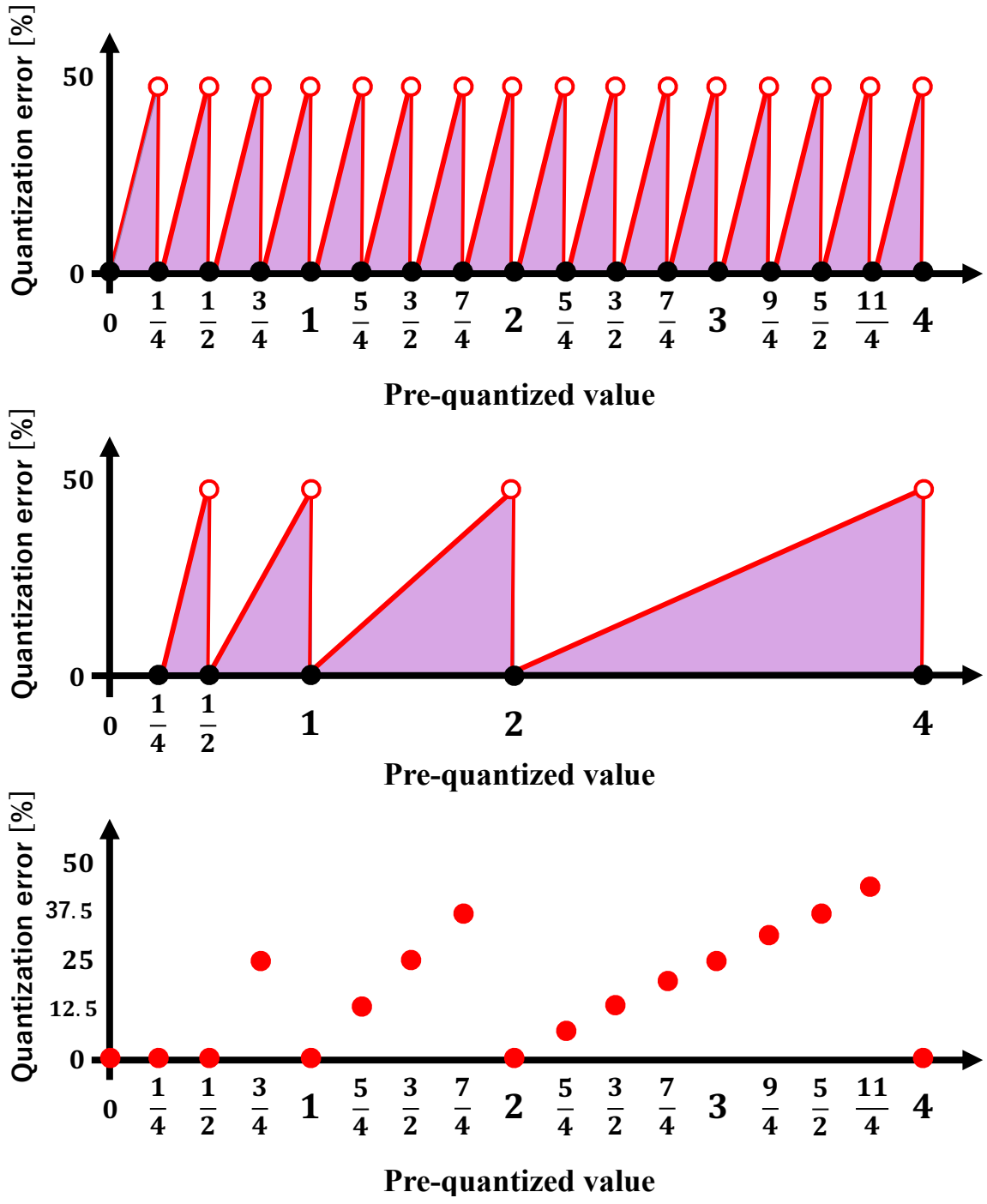

図 5.1　小数点以下 2bit までの量子化に伴う誤差の割合．上から順に線形量子化 (固定小数点化)，対数量子化，固定小数点化されたものに対数量子化を施した際の量子化誤差の割合を示す．(上) 量子化後の取り得る値は等間隔で位置しており，誤差の割合は 0% 以上 50% 未満である．(中) 量子化後の取り得る値は等間隔ではなく，絶対値が大きいほど間隔が広くなる．誤差の割合は 0% 以上 50% 未満である．(下) 絶対値が小さい時は誤差が 0 になる割合が高く，絶対値が大きくなるにつれて誤差が 0 になる割合が低くなっている．

小数を表す代表的な方法には浮動小数点方式と固定小数点方式の 2 つが存在する．特にエッジ AI デバイスにおいては演算リソースの観点から固定小数点方式が採用されてきた．ここで固定小数点方式は符号部・整数部・小数部から構成されており，整数部と小数部に用いるビット数により数値の表現力が決定される．このビット数が多い (＝ビット精度が高い) ほど情報量の削減による誤差が生じにくくなるものの，必要メモリが多くなる．固定小数点化の量子化誤差によるニューラルネットワークの性能低下とパラメータ保存に関するメモリ容量はトレードオフの関係にある．ビット精度はニューラルネットワークのハードウェアアーキテクチャを考案する際のハイパーパラメータとなり，これまで経験則的に決められてきた．

図 5.1 上部に示している通り，浮動小数点方式から固定小数点方式へと量子化を行った場合，量子化前の値から見た量子化誤差の割合の平均値が 25% となる．同様に，図. 5.1 中部に示している通り，浮動小数方式から対数量子化を行った場合も，量子化前の値から見た量子化誤差の割合の平均値が 25% となる．例えば量子化前の値が 6 であれば 4 (= 2^2) に変換され，量子化前の値が 0.1 であれば 0.0625 (= 2^{-4}) に変換される．つまり，前節で定義した対数量子化とは量子化前の値以下で最も近い 2 の累乗の値に近似することを示す．いずれの場合も量子化前の値の大きさと量子化誤差の割合との間には相関がなく，量子化誤差により元の値の 75% となることは前述のモーメンタム項のハイパーパラメータ β (= 0.75) を乗ずることに等しい．ただし本提案手法は，既に固定小数点化されたモーメンタム項に対数量子化を施すというものである．この際の量子化誤差の割合は図 5.1 下部に示すようになる．各インターバルについて個別に分析すると，表 5.1 のようになり，量子化前の値が大きいほど誤差の割合が大きくなる．つまり，モーメンタム項の値が小さな場合には減衰を小さく，大きな場合には減衰も大きくする．また，従来のハイパーパラメータ β はスカラ値による一律の乗算であったが，本提案では個々のモーメンタム項に応じた演算結果が得られる．そのため，定性的には高度な最適化手法である RMSProp に近い効果が得られることが期待できる．

5.3　評価

本章では提案手法である Holmes について，最適化アルゴリズムを評価するベンチマーク関数である Rosenbrock 関数 (5.1) と Three–hump camel 関数 (5.2) を用いた評価と，ニューラルネットワークの標準的なデータセットである MNIST を用いた評価を

表 5.1　量子化前の値を区間ごとに分け，それぞれの量子化割合の平均値を示す．絶対値が大きくなるにつれて誤差の割合の平均値が大きくなっていることが分かる．また，この図では量子化ビット数を整数 3bit 小数 2bit で考えている．

Pre-quantized value (P)	Average value of quantization error [%]
$P < 0$	0
$0 \leq P < 2^{-2}$	0
$2^{-2} \leq P < 2^{-1}$	0
$2^{-1} \leq P < 2^{0}$	12.5
$2^{0} \leq P < 2^{1}$	18.8
$2^{1} \leq P < 2^{2}$	21.9
$2^{2} \leq P < 2^{3}$	24.2
$2^{3} \leq P$	–

行った結果について示す．それぞれのベンチマーク関数の定義式は以下の通りである．

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \tag{5.1}$$

ここで，n は次元数であり本評価では 2 とした．

$$f(x, y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2 \tag{5.2}$$

ニューラルネットワークを用いた実験での条件を表. 5.2 に示す．ここで，使用したモデルは入力層，中間層，出力層の 3 層からなる多層パーセプトロンである．比較対象として固定小数点方式へと量子化を行った SGD と MomentumSGD を用いた．Holmes においてはモーメンタム項は 16bit から 5bi へとさらに量子化されていることに注意されたい．

表 5.2　ニューラルネットワークを用いた実験での各条件.

データセット	MNIST
モデル構造	input 784, hidden 128, output 10
ミニバッチサイズ	32
活性化関数	Sigmoid
学習率	0.25
ビット精度	16bit (符号部 1, 整数部 2, 小数部 13)
ビット精度 (Holmes)	5bit (符号部 1, 対数部 4)

5.3.1　ベンチマーク関数を用いた評価

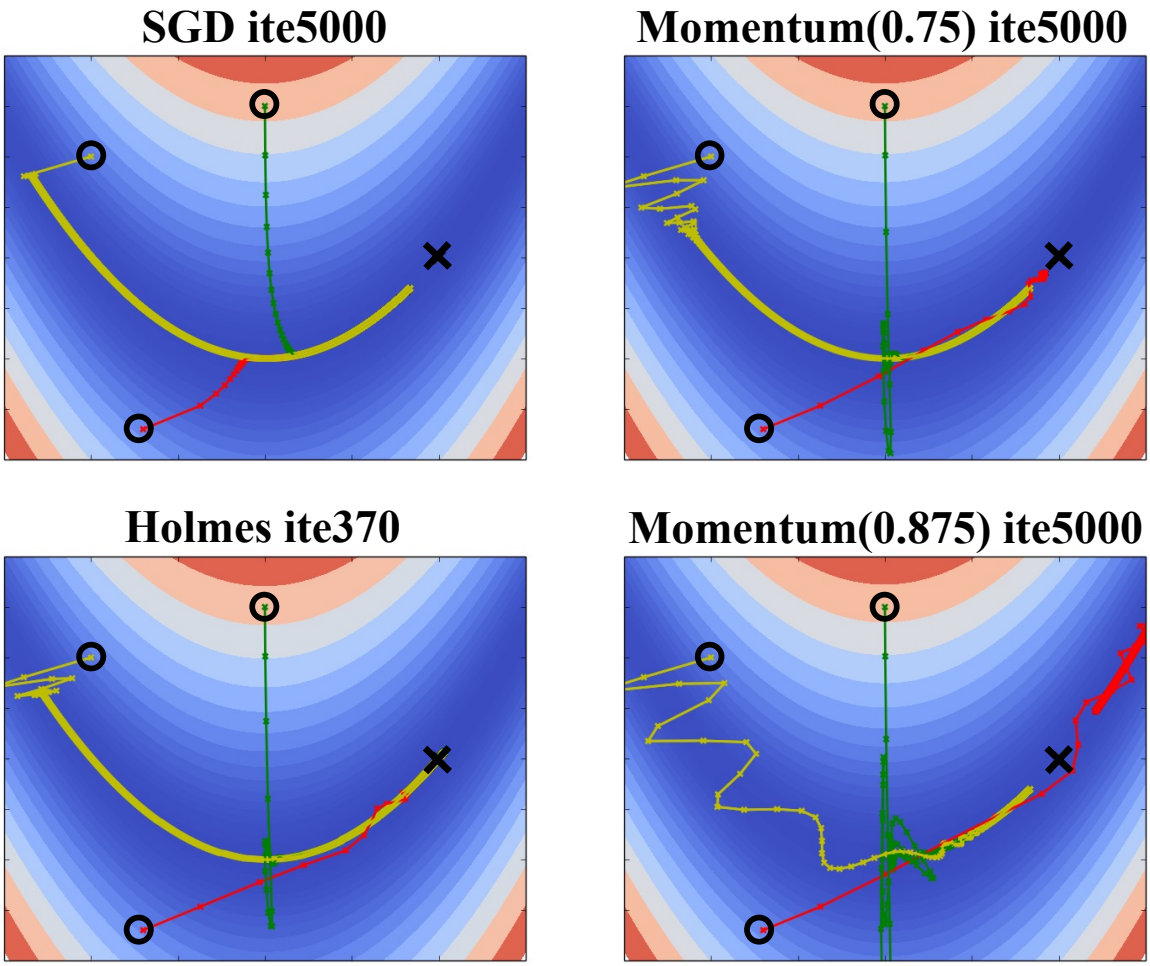

図 5.2　Rosenbrock 関数を用いた評価結果．O からスタートし最適解 X へと収束していく様子を示す．初期値は 3 種類でありそれぞれ赤，緑，黄で示す．本条件下で Holmes は 370 回の更新で最適解へと到達した．SGD と MomentumSGD は 5000 回の更新でも最適解へと到達できていない．また，MomentumSGD において β が 0.875 の時は安定していない様子が見られた．

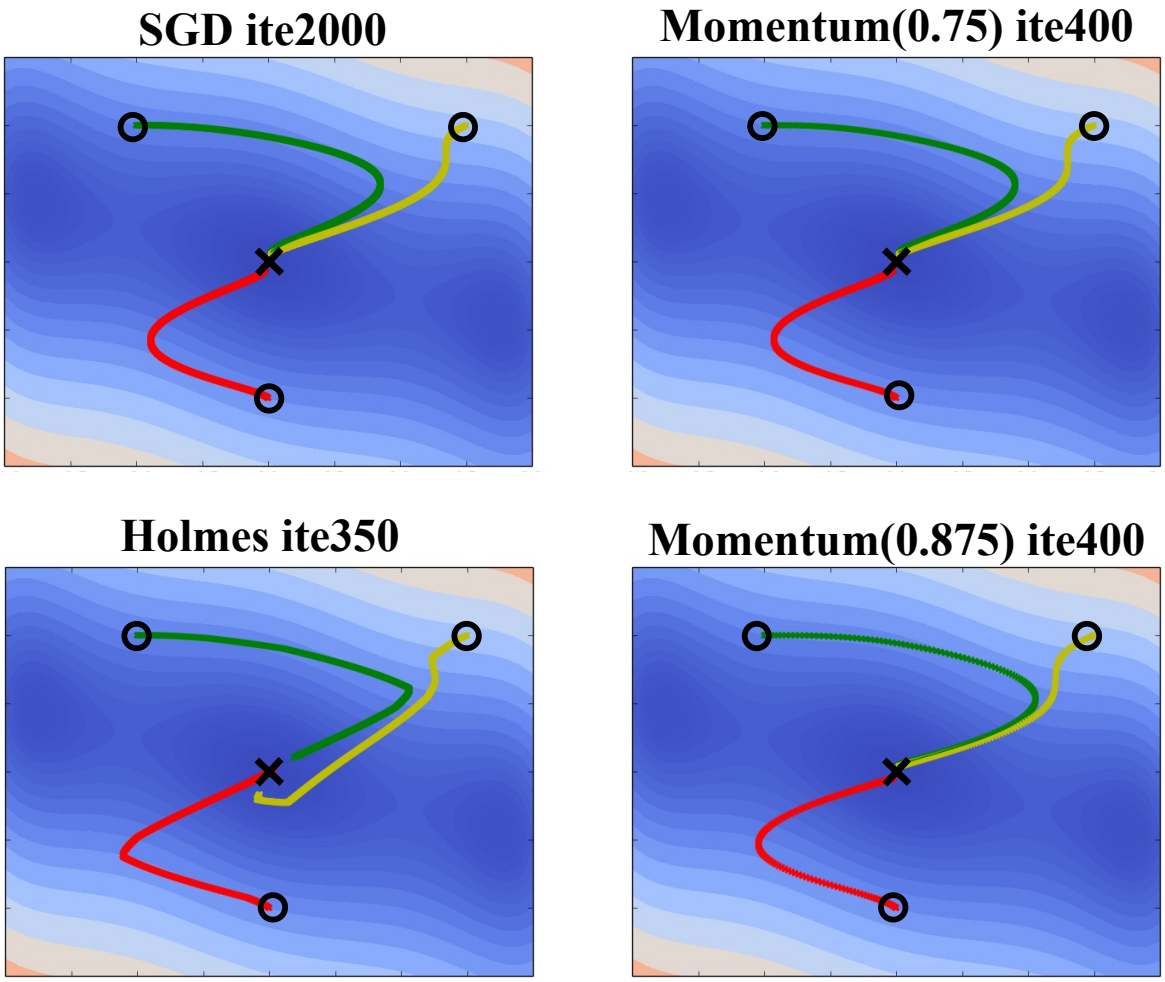

図 5.3 Three–hump camel 関数を用いた評価結果．O からスタートし最適解 X へと収束していく様子を示す．初期値は 3 種類でありそれぞれ赤，緑，黄で示す．本条件下では Holmes は 350 回の更新で最適解へと到達した．SGD は 2000 回の更新で最適解へと到達し，MomentumSGD は β がいずれの場合にも 400 回の更新で最適解へと到達した．

図 5.2 に Rosenbrock 関数を，図 5.3 に Three–hump camel 関数を用いた場合の比較結果を示す．それぞれの関数で最適化を行い，最適解にたどり着くまでの経路と更新回数 (最大 5000 まで) を示している．用いた初期値は 3 種類であり，それぞれを赤，緑，黄で探索を行った経路を示している．MomentumSGD においてはハイパーパラメータ β が 0.75 の時と 0.875 の時の 2 種類で行った．図 5.2 のローゼン関数においては，Holmes 用いた場合の最適解への収束速度が SGD や MomentumSGD に比べて 10 倍以上速いことが確認できた．ただし，本実験においては SGD と MomentumSGD 共に固定小数点化を

施したものを用いたために，これら二つでは量子化誤差の影響により最適解へと到達できなかったものと考えられる．移動の様子を見ると Holmes は SGD や MomentumSGD よりも振動から抜け出す速度が速いことが確認できた．また，MomentumSGD において β の設定が収束までの安定に大きな影響を与えることが確認できる．β はハイパーパラメータ経験則的に決める必要があるが，Holmes ではその調整の必要がなく適切な減衰項を自動で得られている．図 5.3 の Three hump camel 関数上においては，最適解への収束まで SGD では 2000 回要したが，Holmes と MomentumSGD は更新回数に大きな差は見られなかった．Holmes において，特に黄の初期値で顕著であるが，最適解への収束に向かう際に遠回りしている様子が見受けられる．これは勾配よりもモーメンタム項の影響が強く出ていることに起因すると考えられる．β の乗算を対数量子化誤差により代替を行っているが，勾配 (ηg_t) の値が小さい時には代替できない可能性が存在する．

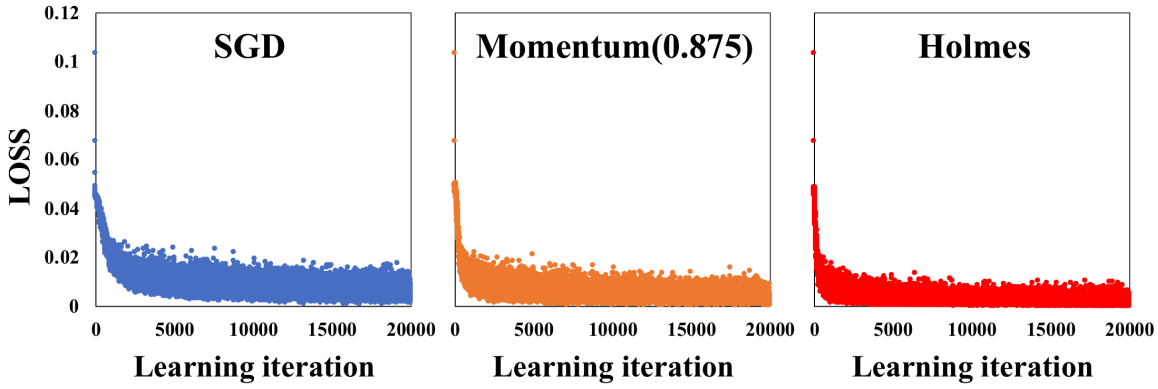

図 5.4　SGD，MomentumSGD と Holmes の損失の比較結果．Holmes，MomentumSGD，SGD の順で高速に損失が収束へ向かうことが確認できる．

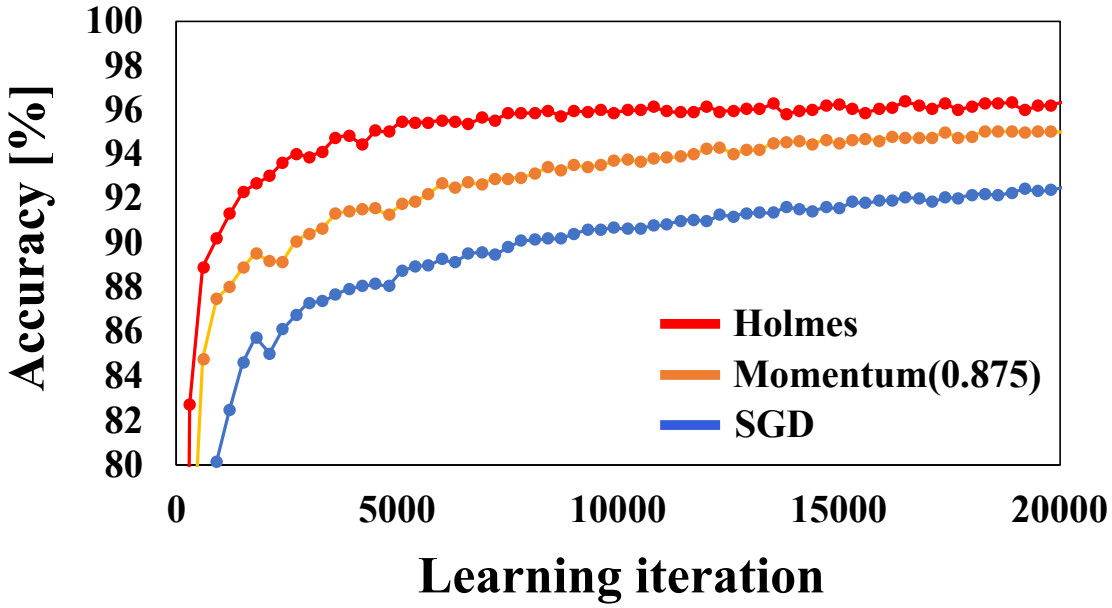

図 5.5 Holmes，MomentumSGD，SGD での精度比較結果．Holmes は MomentumSGD や SGD と比較すると高速に，かつ高い精度で収束していることが確認できる．データ点は 300 回毎の学習回数でプロットした．

5.3.2 全結合型 NN への適用

次に，MNIST データセットを用いて SGD，MomentumSGD，Holmes での収束速度について 1 回の学習毎の損失を用いて比較した．ここで，ミニバッチサイズが 32 であるから 1epoch の学習回数は 1875 回である．図 5.4 に示すように，学習開始時はほぼ同じ損失である．1epoch 後の損失の大きさを比較すると Holmes，MomentumSGD，SGD の順になっていることが確認できた．ただし，MomentumSGD の減衰項 β については事前実験から最も良い精度が得られた 0.875 とした．

最後に Holmes と SGD，MomentumSGD の汎化性能の違いをテストデータの精度を用いて比較した．図 5.5 では 300 回の学習毎にその時点でのモデルによる認識精度を算出したものを示す．テストデータセットでの評価においても Holmes は SGD や Momentum に比べて高い精度を達成できていることが確認できた．特に学習回数が少ない時の性能の差が顕著である．例えば学習回数が 5000 回の時には SGD のスコアが 88.06%，Momentum のスコアが 91.28% であるのに対し，Holmes は 95.03% の認識精

度を達成している．MomentumSGD では 95% の精度を達成するには 20000 回以上の学習を要することから，Holmes では 4 倍以上の学習速度向上を果たしている．ニューラルネットワークアーキテクチャにおいて消費される電力の約 50% であることを考えると，高速な学習を可能とする本手法を用いることで低電力化に大きく寄与することが予見される．特に制約の厳しいエッジコンピューティングにおいては，高い学習精度と高速な学習による低電力化，対数量子化を用いることでの省メモリ化という三点からも，エッジデバイスでのオンライン学習を可能にする非常に有用な手法であると考えられる．

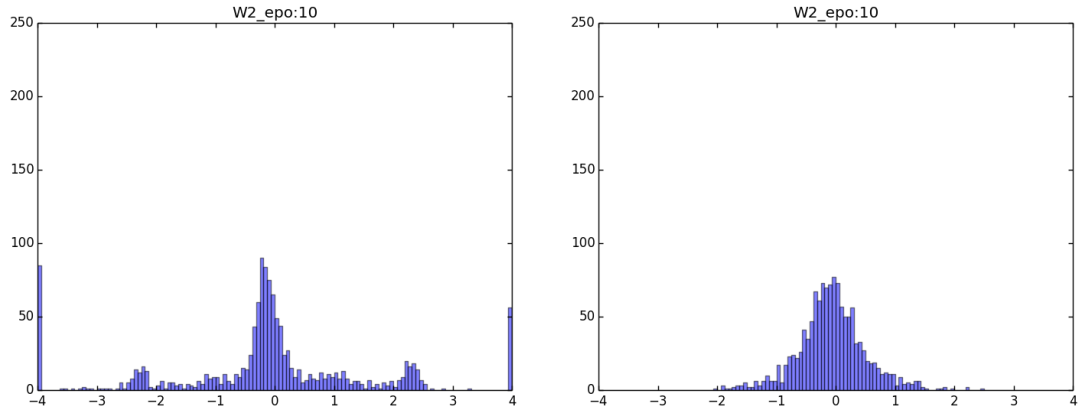

図 5.6　中間層と出力層間の重み分布についてのヒストグラム．それぞれ 10epoch 学習した後の様子を示している． (左)Holmes の学習では重みが ±4 へと発散している． (右)Holmes へ忘却機構を追加することで発散を防ぐことが可能である．

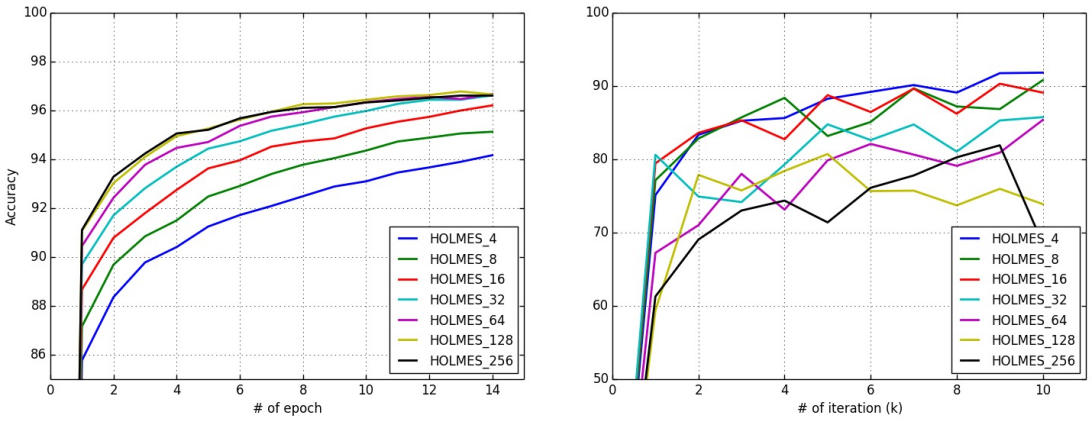

図 5.7　リセット機構を追加した場合の MNIST データセットにおける精度比較．リセットタイミングは図中の HOLMES_XX に示しており，16 の場合は 16 回の学習毎にモーメンタム項を 0 としている．(左) ミニバッチサイズが 32．(右) ミニバッチサイズが 1

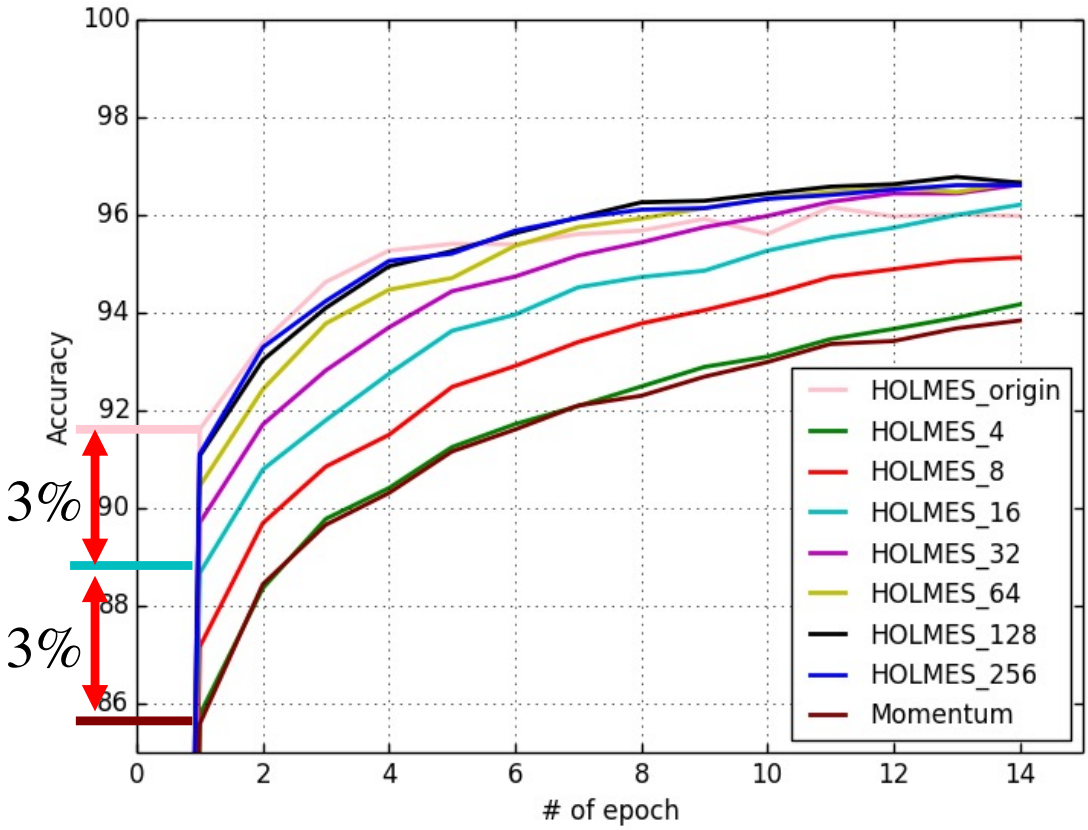

図 5.8　図 5.7 にリセット機構の無い Holmes (Holmes_origin) と MomentumSGD を追加した評価結果．各リセットタイミングを示す色が図 5.7 とは変わっている点に注意されたい．

5.3.3　パラメータ分布

図 5.6 左部に Holmes を用いて 10epoch 学習を行った場合の中間層と出力層間の重みのヒストグラムを示す．学習により重みが発散し，–4 や +4 となる重みが増えていくことが確認できた．これは 5.3.1 節で触れた通り，特に勾配の値が小さな時には対数量子化による乗算の代替が機能しないことから．モーメンタム項が減衰せずに残存し続けることに起因すると考えられる．そのため，一定期間毎にモーメンタム項をリセット機構を追加することでこの解決を図った．これによる学習結果のヒストグラムを図 5.6 右部に示す．結果より，発散が防げていることが確認できた．

図 5.7 にリセットタイミングを 4~256 回毎とした場合での MNIST の精度評価結果について示す．左側はミニバッチサイズが 32 の時を，右側はミニバッチサイズが 1 (＝オンライン学習) の結果をそれぞれ示す．ミニバッチサイズが小さくなるにつれ，サン

表 5.3　Holmes，MomentumSGD，RMSProp のリソース要件を SGD と比較したもの．比較は，必要なメモリ，必要な演算，ハイパーパラメータの数の 3 つのカテゴリで行った．Operations の欄は SGD と比較して増加した操作を示している．

	Memory	Operations	Hyperparameter
SGD	–	–	–
Holmes	$\log_2(\mathrm{w} \times \mathrm{bit})$	addition	±0
Momentum	$w \times \mathrm{bit}$	addition multiplication	+1
RMSProp	$2 \times w \times \mathrm{bit}$	addition multiplication square root division	+1

プルデータの影響が強くなり汎化性能が下がることから認識精度は低下する．両条件を比較した結果リセットタイミングを 16 回毎 (赤色) とするのが適当であると判断した．次に図 5.8 にリセット無しをピンク色とし，MomentumSGD を茶色として追加したものを示す．なお，本図においては 16 回毎のリセットを示す色が水色となっている．Holmes の利点の一つは高精度を達成するまでに要する学習回数が少ない点である．リセット機構を追加しその間隔を 16 回毎とした場合において，1epoch 時点での精度は 3% 低下，収束までの学習回数が 2 倍になってしまうが，依然として MomentumSGD よりも速く高精度を達成できることが確認できた．

5.4　ハードウェア化に向けた一考察

第 5.3 節では，ノイマン型の AI アーキテクチャにおいて電力的なボトルネックとなるメモリアクセス回数に係る評価を行った．本節ではその他ハードウェア化するにあたってメモリコストや必要な演算などについて，提案手法である Holmes と MomentumSGD，RMSProp との比較・分析を行う．

5.4.1　必要メモリ容量

Holmes においてモーメンタムベクトルの保存に必要となるメモリ容量は，重みとバイアスそれぞれのモーメンタムに対して “符号” と “対数量子化後の値”，つまり $w\times(1+\lfloor\log_2(\text{bit}-1)\rfloor)$ となる．ここで，w はパラメータの総数であり，bit は量子化ビット数である．例えばモデル構造が 2–4–1 のバイアス項を含む MLP を 16bit 固定小数点方式で表現している場合を考えると，$17\times(1+4)$ となる．一方で，MomentumSGD ではモーメンタムベクトルを記憶するのに必要なメモリ量が $w\times$bit (bit = 量子化ビット数) であり，RMSProp ではモーメンタムベクトルは無いが，過去の勾配の 2 乗和 v を保存する必要がある．勾配を 2 乗した値を保持するのに必要なメモリは $2\times w\times$bit である．よって，モーメンタムベクトルの保存に要求されるメモリ容量は Holmes $(w\times(1+\lfloor\log_2(\text{bit}-1)\rfloor))$，MomentumSGD $(w\times\text{bit})$，RMSProp $(2\times w\times\text{bit})$ の順に小さくなる．

5.4.2　必要な演算リソース

次に各最適化手法の実装において必要となる演算リソースを考える．表 5.3 は，SGD を基準としてそれぞれの最適化手法を用いる際に必要となる演算やハイパーパラメータの増加量を示している．Holmes はモーメンタムベクトルを重みに加算する処理が増えている．MomentumSGD ではモーメンタムベクトルを重みに加算する処理とモーメンタムベクトルにハイパーパラメータ β を乗算する処理が増えている．ここで，加算と乗算は SGD 内でも使っているので，Holmes と MomentumSGD では演算器を共有することで演算リソースの増加量を最小限にすることが可能である．

一方で，RMSProp では加算と乗算に加えて平方根演算と除算が増えている．平方根演算と除算は SGD 内では使われていない．それに，最も致命的なのは平方根演算と除算をハードウェアで実装するためには多量の演算リソースが必要であり，特にエッジコンピューティングには向かないことは自明である．

5.4.3　ハイパーパラメータ

ハイパーパラメータとは，学習率に代表されるようにユーザが手動で適宜設定しなければならないパラメータである．解く問題に応じて適切な値が変化するので，ハイパーパラメータの増加は探査空間の増加に繋がる．AI の実応用を考えた場合にはこの問題は無視できないものとなる．ここで，MomentumSGD のハイパーパラメータであ

る βm_{t-1} の部分が Holmes では $2^{\lfloor \log_2(m_{t-1}) \rfloor}$ となっており，β が無くなっている．Holmes は MomentumSGD のハイパーパラメータの削減を可能としている．RMSProp にはモーメンタムベクトルは無いが，別のハイパーパラメータを用いて学習率が小さくなりすぎないようにしている．よってハイパーパラメータは MomentumSGD と RMSProp が同数で Holmes はそれらより 1 個少ない．

RMSProp との類似性

MomentumSGD と Holmes ではどちらもモーメンタムベクトルを使っている．MomentumSGD では，モーメンタムベクトルの調整を一定の値 β を乗じることで行っているが，Holmes においては二重の量子化によってこれを行っている．二重の量子化によるモーメンタムベクトルの調整は，単純にモーメンタムベクトル全体を一律に削減するのではなく，削減割合はモーメンタムベクトルの各要素の大きさに応じて変化する．この “各要素の大きさに応じて調整する” という副次的に得られた効果は，RMSProp において行われている学習率の自動調整と類似している．そのために，MomentumSGD より高い収束性を獲得していると考えられる．

5.4.4 アーキテクチャ設計

第 3 章において提案したアーキテクチャをベースにして，MomentumSGD と Holmes の誤差逆伝播とパラメータ更新部のハードウェアアーキテクチャブロック図を構築した．図 5.9 に示した SGD のブロック図と比較すると，図 5.10 に示す MomentumSGD では，モーメンタムベクトル，ハイパーパラメータ β を格納するためのメモリと，乗算・加算の演算器が追加される．ここで，モーメンタムベクトルを格納するメモリのサイズは前述の通り $w \times \text{bit}$ となる．同様に，図 5.11 に示す Holmes も同様にモーメンタムベクトルを格納するためのメモリが追加されているが，対数量子化により圧縮されているため，格納に必要なメモリ量は $w \times (1 + \lfloor \log_2(\text{bit} - 1) \rfloor)$ となる．また，Holmes ではハイパーパラメータの追加はないものの代わりに対数量子化を行う圧縮器と展開器が追加されている．圧縮器は，上位ビットから順に 0 と 1 が反転する場所を検索し，“最上位ビット (符号ビット)” と “0 と 1 が反転する場所の下位ビット側の位置 (反転位置)” の情報を出力する．一方で展開器は，符号ビットが 0 の場合は反転位置のみを 1 とした数値を出力し，符号ビットが 1 の場合は反転位置より上が 1，反転位置と反転位置より下が 0 とした数値を出力する．

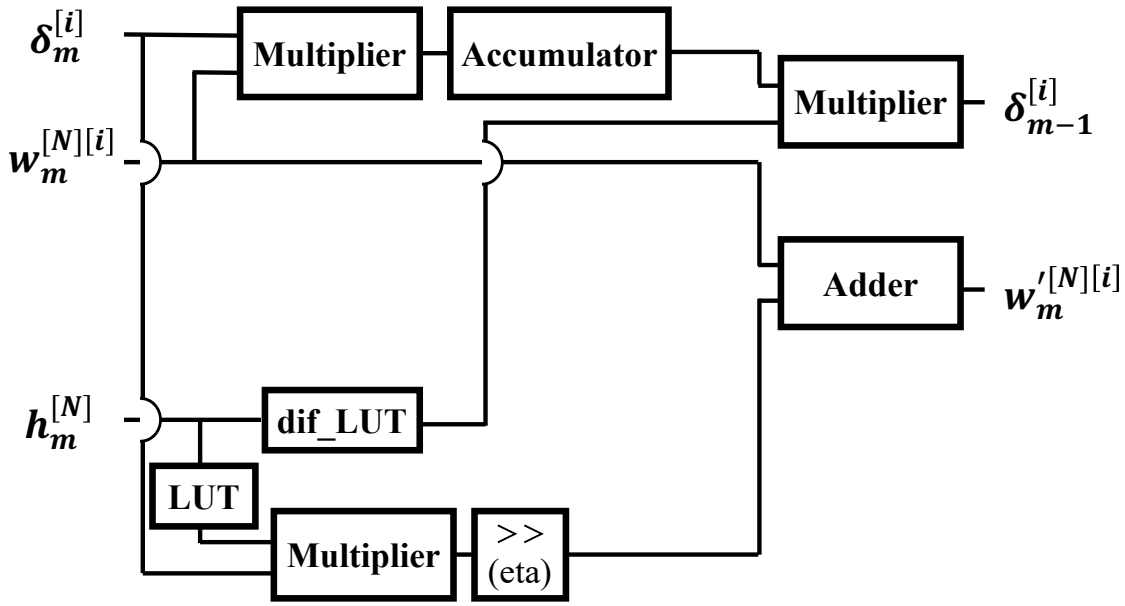

図 5.9 SGD の誤差逆伝播部とパラメータ更新部のブロック図抜粋．h は 1 つ前の層の出力値であり，δ は 1 つ前の層の損失であり，>> は学習率をビットシフトで代用したものである．LUT は活性化関数をルックアップテーブル形式で格納されており，dif_LUT は活性化関数の微分をルックアップテーブル形式で格納されている．

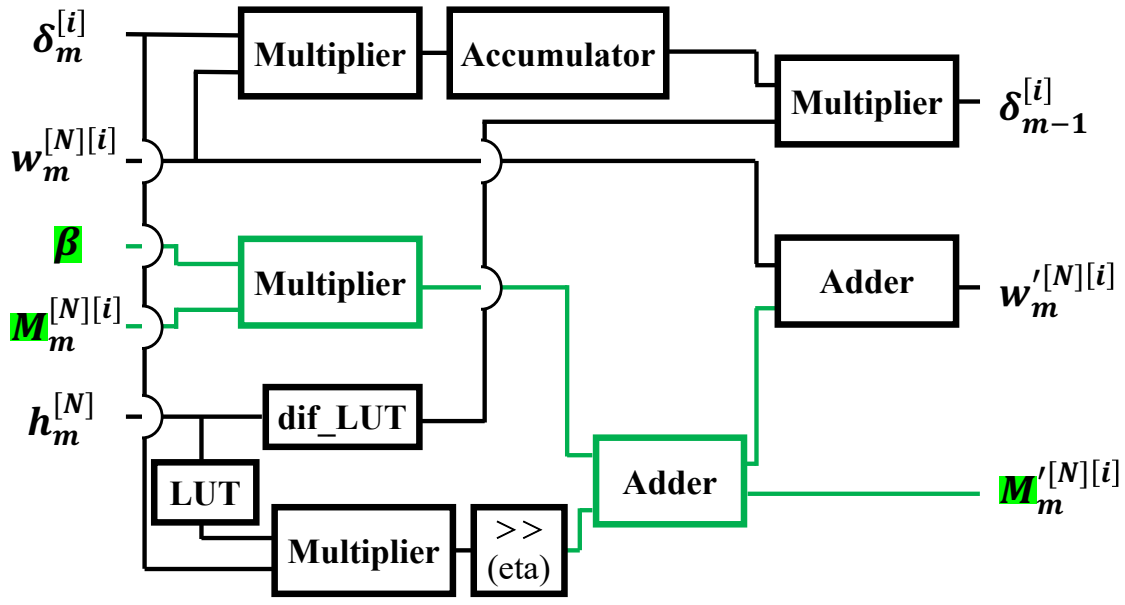

図 5.10 MomentumSGD の誤差逆伝播部とパラメータ更新部のブロック図．緑色の部分は，図 5.9 と比較して追加されたモジュール．

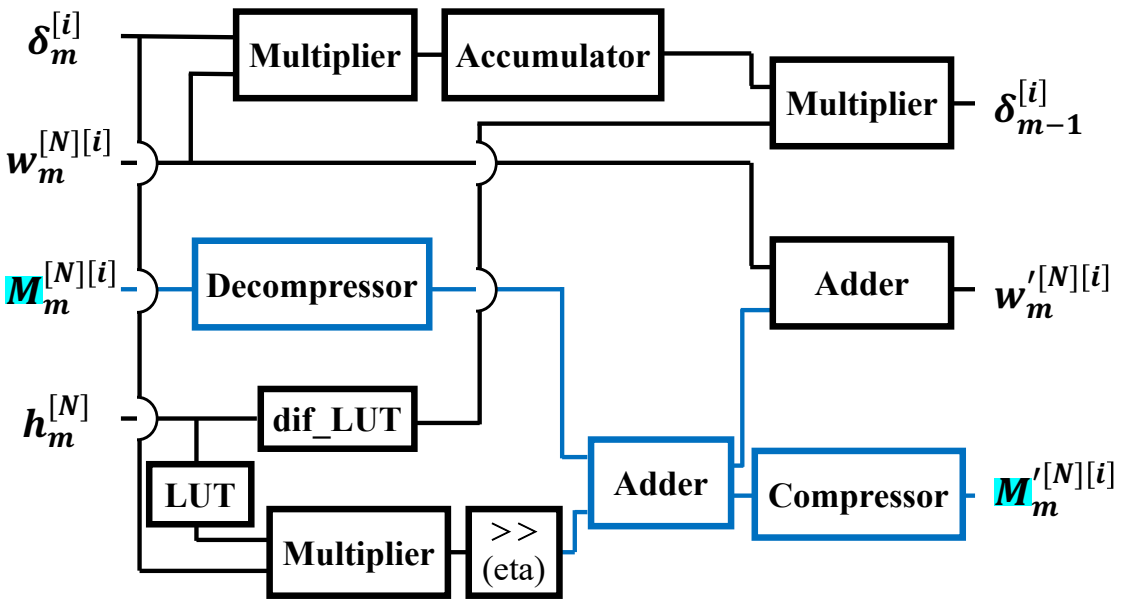

図 5.11 Holmes の誤差逆伝播部とパラメータ更新部のブロック図．青色の部分は，図 5.9 と比較して追加されたモジュール．圧縮機は，入力された値を対数量子化した値を出力し，さらにその出力値は量子化後の値を表現できる必要最低限の情報量となっている．展開器は，その必要最低限の情報量を数値に戻して出力する．

5.5 結論

本論文では，エッジ AI のオンライン学習に向けた最適化手法の提案を行った．本提案手法は MomentumSGD を基に考案しており，モーメンタム項のハイパーパラメータ乗算に関する部分を対数量子化に置き換えるものである．これにより，従来 16 ビットで保存されていた値を 5 ビットに量子化することでメモリ容量の削減を可能とした．また，従来スカラ値で一律に減衰率が計算されてたものを，個々のパラメータに応じた減衰を行うことにより高速なパラメータ探索を可能とした．MNIST データセットにおける精度評価では同条件の MomentumSGD よりも 4 倍以上の学習高速化を果たした．制約の厳しいエッジデバイス上でも，メモリ容量の削減ならびにメモリアクセス回数削減による低消費電力化とを両立することで高精度なオンライン学習を可能にすることを示した．

参考文献

[1] M. Tan and Q. Le. “EfficientNetV2: Smaller Models and Faster Training”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. **139**. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 10096–10106.

[5] A. Vaswani, N. Shazeer, N. Parmar, et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. **30**. Curran Associates, Inc., 2017.

[6] J. Kaplan, S. McCandlish, T. Henighan, et al. “Scaling Laws for Neural Language Models”. In: *arXiv preprint arXiv:2001.08361* (2020). arXiv: 2001.08361.

[9] D. Silver, J. Schrittwieser, K. Simonyan, et al. “Mastering the Game of Go without Human Knowledge”. In: *nature* **550**.7676 (2017), pp. 354–359.

[15] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[16] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). arXiv: 1412.6980.

[25] N. P. Jouppi, C. Young, N. Patil, et al. “In-Datacenter Performance Analysis of a Tensor Processing Unit”. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. 2017, pp. 1–12.

[28] S. K. Gonugondla, M. Kang, and N. Shanbhag. “A 42pJ/Decision 3.12TOPS/W Robust in-Memory Machine Learning Classifier with on-Chip Training”. In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, pp. 490–492. DOI: 10.1109/ISSCC.2018.8310398.

[35] S. Zhou, Y. Wu, Z. Ni, et al. *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*. 2016. DOI: 10.48550/ARXIV.1606.06160.

[45] M. Giordano, K. Prabhu, K. Koul, et al. “CHIMERA: A 0.92 TOPS, 2.2 TOPS/W Edge AI Accelerator with 2 MByte on-Chip Foundry Resistive RAM for Efficient Training and Inference”. In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492347.

[46] B. Crafton, M. West, P. Basnet, et al. “Local Learning in RRAM Neural Networks with Sparse Direct Feedback Alignment”. In: *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE. 2019, pp. 1–6.

[82] S. Kang, D. Han, J. Lee, et al. “7.4 GANPU: A 135TFLOPS/W Multi-DNN Training Processor for GANs with Speculative Dual-Sparsity Exploitation”. In: *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2020, pp. 140–142. DOI: 10.1109/ISSCC19947.2020.9062989.

[88] B. Moons, R. Uytterhoeven, W. Dehaene, et al. “14.5 Envision: A 0.26-to-10TOPS/W Subword-Parallel Dynamic-Voltage-Accuracy-Frequency-Scalable Convolutional Neural Network Processor in 28nm FDSOI”. In: *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, pp. 246–247. DOI: 10.1109/ISSCC.2017.7870353.

[89] K. Ueyoshi, K. Ando, K. Hirose, et al. “QUEST: A 7.49TOPS Multi-Purpose Log-Quantized DNN Inference Engine Stacked on 96MB 3D SRAM Using Inductive-Coupling Technology in 40nm CMOS”. In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, pp. 216–218. DOI: 10.1109/ISSCC.2018.8310261.

[90] Z. Wang, Z. Li, L. Xu, et al. “An All-Weights-on-Chip DNN Accelerator in 22nm ULL Featuring 24×1 Mb eRRAM”. In: *2020 IEEE Symposium on VLSI Circuits*. 2020, pp. 1–2. DOI: 10.1109/VLSICircuits18222.2020.9162811.

[91] A. Nøkland. *Direct Feedback Alignment Provides Learning in Deep Neural Networks*. 2016. DOI: 10.48550/ARXIV.1609.01596.

[92] D.-H. Lee, S. Zhang, A. Fischer, et al. *Difference Target Propagation*. 2014. DOI: 10.48550/ARXIV.1412.7525.

[93] D. Han, J. Lee, and H.-J. Yoo. “DF-LNPU: A Pipelined Direct Feedback Alignment-Based Deep Neural Network Learning Processor for Fast Online Learning”. In: *IEEE Journal of Solid-State Circuits* **56**.5 (2021), pp. 1630–1640. DOI: 10.1109/JSSC.2020.3042978.

[94] N. Qian. “On the Momentum Term in Gradient Descent Learning Algorithms”. In: *Neural Networks* **12**.1 (1999), pp. 145–151. DOI: 10.1016/S0893-6080(98)00116-6.

第6章　総括

現在，AIを用いた情報処理はコンピューティング基盤の性能向上とビッグデータそしてAIアルゴリズムの発展を背景として隆盛期にある．エッジデバイスから収集されるデータをデータセンタであるクラウドへと集約して処理するAIシステムに内在する社会的・技術的な課題から，データを集約せずにその場で処理を行うエッジコンピューティングへの期待が高まっている．特に，従来クラウドで行われていたAIモデルの学習処理をエッジ上で完結させることで，機密性・可用性を更に高めようという動きが盛り上がりを見せている．本研究では，電力や演算性能が制限されるエッジデバイスにおいて演算負荷の高い学習処理を実現するための新規アルゴリズムとそれらを実装するアーキテクチャの構築を目的とした．

現在のAI技術の主流であるニューラルネットワークは推論処理と学習処理の二つに大別することができる．ここで，学習処理は更に誤差逆伝播法と最適化手法という2つの手法から構成されており，これらの演算は計算負荷が高くエッジデバイス上でリアルタイムかつ小さな電力で処理を行うためにはハードウェア化を見据えたアルゴリズムの改良や，アーキテクチャの効率化を行う必要がある．予てより推論処理において，AIモデルのパラメータや出力値等の数値情報を削減することで演算性能を維持したまま低消費電力化が可能であると知られている．そのため，まずは学習処理においても演算性能を維持したままで情報量の削減を行うアルゴリズム・アーキテクチャの開発に取り組んだ．

第2章では，ニューラルネットワークの学習に使われる最適化手法の中で最も基本的な確立的勾配降下法のハードウェア指向アルゴリズムについて述べた．本アルゴリズムでは数値表現に係るビット精度を32ビットの浮動小数点方式から固定小数点方式へと変更しビット精度を制限することで演算の軽量化を行った．生成モデルの一つであるGAN (Generative Adversarial Networks) を対象に，推論・学習処理のビット精度を探査し，それぞれ最大7ビット・25ビット要することが判明した．また，上記知見を基に仮想のアーキテクチャを考案した．本アーキテクチャは推論処理を行うPE (Processing Elements) と学習処理を行うPEで構成されており，いずれも内部に並列処理に関する

ハイパーパラメータを有している．並列性を可変にすることで性能と電力間のトレードオフの選択が可能となり，広範な用途に対応できることを示した．

第 3 章では，誤差逆伝播法を軽量化する手法の提案とアーキテクチャの構築，FPGA への実装について述べた．本提案手法では，固定小数点方式のオンデバイス学習という条件下において推論と学習間とで生じる要求ビット精度の差に着目した．学習前後のパラメータの変化の様子を観察した結果，推論時に使用される値が変化する割合は極小数であることを確認した．つまり，パラメータに関わるメモリの容量や書き込み電力等の大部分は無為に消費されていることを意味する．そこで，性能を維持したまま誤差逆伝播法のビット精度・書き込み回数を削減するアルゴリズムを開発した．また，演算リソース量が最低となるエッジ AI 向けアーキテクチャを考案し FPGA へと実装した．既存手法との比較を行い，使用メモリ量を 49.8%削減可能であることを示したほか，見積もり上であるがメモリアクセスに係る消費電力を 0.0017 倍にまで削減可能であることを示した．

第 4 章では，アナログ回路を導入した CIM (Computing-in-Memory) デバイスに向けた誤差逆伝播法及び CIM デバイスを外部メモリとして用いることで学習アーキテクチャの構築について述べた．特に低電力が求められるエッジ AI においては，従来のノイマン型アーキテクチャで生じるプロセッサとメモリ間のボトルネックが大きな問題となる．この解消に向けて，メモリ上で AI 処理を行う非ノイマン型アーキテクチャである CIM が注目されている．提案学習アルゴリズムは，ReRAM を用いてアナログ回路的に推論処理を行う CIM デバイスに向けたものである．本 CIM デバイスの特性から通常の誤差逆伝播法ではなく Digital BP 法を用いた．より複雑なモデルに対しては性能が低くなるという問題点があり，この解決に向けてアルゴリズムの改良を行った．改良アルゴリズムを用いることで線形回帰や多クラスの識別タスクの学習が可能となることを示した．また，アーキテクチャの構築と FPGA への実装を行い，ソフトウェアシミュレーションとの比較により同等の性能が得ていることと，演算コア部の消費電力を 10mW 以下にできることを示した．

第 5 章では，軽量性と収束性を両立する最適化手法について述べた．従来のエッジ AI 向け学習アルゴリズム・アーキテクチャ研究では，最適化手法として確立的勾配降下法あるいはこれに少々の改良を加えたものを用いている．高度な最適化手法は慣性項を用いるためにメモリ容量の増大を招くことや，平方根演算等の演算処理が必要となりエッジ AI 運用を考えた際の問題点となっている．提案最適化手法は，より高度な最適化手法で用いられている慣性項へ二重の量子化を施すことでメモリ容量と演算処理の

削減を可能とした．慣性項を用いる従来の最適化手法と比べて，メモリ領域を約 70% 削減と収束までにかかる学習回数を 1/4 にまで削減とを両立していることを示した．

これらの研究は，電力や演算性能に厳しい制約を持つエッジデバイスにおいて完全なオフライン環境下での AI 処理を可能とするためのものである．本研究が，Internet of Things だけではなく Intelligence on Things，更にはその先の次世代情報社会の実現に向けた一歩目となることを期待する．

謝辞

本研究は北海道大学大学院情報科学院，集積ナノシステム研究室において 2017 年 4 月から現在に至るまでに行った研究をまとめたものである．本研究の遂行と論文の執筆に際し，ご多忙のなか多大なるご指導ご協力を賜りました，北海道大学大学院情報科学研究院 集積ナノシステム研究室 浅井哲也教授に深く感謝申し上げます．同様に，百瀬啓氏には主に論文執筆や研究面に関して多大なるご協力をいただきました．深く感謝いたします．特に第 4 章において，研究遂行と論文執筆にあたり河野和幸様をはじめとするヌヴォトンテクノロジージャパン株式会社に多大なるご協力を頂きましたこと深謝いたします．本論文の作成にあたり，有益な御討論をして頂いた北海道大学大学院情報科学院情報科学専攻 池辺将之教授，本久順一教授，葛西誠也教授，富田章久教授に厚く御礼申し上げます．本研究を進めるにあたり，有益な御討論をして頂いた北海道大学大学院情報科学研究院 集積ナノシステム研究室 安藤洸太助教授，東京工業大学 科学技術創成研究院 本村真人教授，東京大学大学院情報理工学系研究科 高前田伸也准教授に厚く御礼申し上げます．同様に萩原成基氏，阿部佑紀氏をはじめとする集積ナノシステム研究室の皆様方には有意義な議論をして頂きました．以上の方々に心より感謝いたします．この他にも多くの先生方，先輩方にも御助言や御支援を頂きましたことに謹んで感謝の意を表します．

最後に，本研究の遂行にあたり精神的，経済的にも支援頂いた両親と家族へ心よりの感謝を．

本研究に関する発表・業績

学術論文

1. Yamagishi Y., Kaneko T., Akai-Kasaya M., and Asai T., "Holmes: A hardware-oriented optimizer using logarithms," IEICE Transactions on Information and Systems, vol. E105-D, no. 12, pp. 2040-2047 (2022).

2. Yamagishi Y., Kaneko T., Akai-Kasaya M., and Asai T., "Hardware-oriented deep reinforcement learning for edge computing," Nonlinear Theory and Its Applications, vol. E12-N, no. 3, pp. 526-544 (2021).

3. （研究紹介）百瀬 啓, 金子 竜也, 浅井 哲也, "脳型ハードウェア作りを実機で体験する AI Circuit Lab," トランジスタ技術, vol. 2021, (2021), in press.

4. （解説記事）百瀬 啓, 金子 竜也, 浅井 哲也, "マイコンに AI を実装するための基礎知識," トランジスタ技術, vol. 2020, no. 11, pp. 169-179 (2020).

5. （解説記事）金子 竜也, "ハードウェア AI 作りの第一歩！ 人工知能の基本アルゴリズム," トランジスタ技術, vol. 2020, no. 10, pp. 30-34 (2020).

6. （解説記事）金子 竜也, "ハードウェア AI 製作に使うアーキテクチャの基本," トランジスタ技術, vol. 2020, no. 10, pp. 53-56 (2020).

7. Momose H., Kaneko T., and Asai T., "Systems and circuits for AI chips and their trends," Japanese Journal of Applied Physics, vol. 59, no. 5, pp. 050502(1)-(15) (2020).

8. Kaneko T., Orimo K., Hida I., Takamaeda-Y S., Ikebe M., Motomura M., and Asai T., “A study on a low power optimization algorithm for an edge-AI Device,” Nonlinear Theory and Its Applications, vol. E10-N, no. 4, pp. 373-389, 2019.

9. Kaneko T., Ikebe M., Takamaeda-Y S., Motomura M., and Asai T., “Hardware-oriented algorithm and architecture for generative adversarial networks,” Journal of Signal Processing, vol. 23, no. 4, pp. 151-154, 2019.

国際会議

1. Kaneko T., Momose H., and Asai T., "On-Device Training Architecture for Analog ReRAM Neural Networks with Digital BP," MEMRISYS 2022, Boston Marriott Cambridge Cambridge, Cambridge, USA (Nov. 30-Dec. 2, 2022).

2. Yamagishi Y., Kaneko T., Akai-Kasaya M., and Asai T., "Hardware design of the target Q-network for edge-oriented deep reinforcement learning," RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing 2021, pp. 30-34, Online (Mar. 1-3, 2021).

3. Kaneko T., Momose H., and Asai T., “An FPGA accelerator for embedded microcontrollers implementing a ternarized backpropagation algorithm,” 2019 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2019), Grand Park Royal Cancun Caribe, Cancun, Mexico, Dec. 2019.

4. Momose H., Kaneko T., and Asai T., “An FPGA accelerator for Arduino implementing a ternarized backpropagation llgorithm,” 2019 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2019) demo session, Grand Park Royal Cancun Caribe, Cancun, Mexico, Dec. 2019.

5. Kaneko T., Ikebe M., Takamaeda-Yamazaki S., Motomura M., and Asai T., “Hardware-oriented algorithm and architecture for generative adversarial networks,” The 2019 RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing, Hilton Waikiki Beach Hotel, Honolulu, USA, Mar. 2019.

6. Kaneko T., Ikebe M., Takamaeda-Yamazaki S., Motomura M., and Asai T., “Ternarized backpropagation: a hardware-oriented optimization algorithm for edge-oriented AI devices,” The 7th RIEC International Symposium on Brain Functions and Brain Computer, Research Institute of Electrical Communication, Tohoku University, Sendai, Japan, Feb. 2019.

7. Kaneko T., Ikebe M., Takamaeda-Yamazaki S., Motomura M., and Asai T., “A study on ternary back propagation algorithm for embedded egde-AI processing,” Joint workshop of UCL-ICN, NTT, UCL-Gatsby and AIBS: Analysis and Synthesis for Human/Artificial Cognition and Behaviour, Seaside House, Okinawa Institute of Science and Technology, Okinawa, Japan, Oct. 2018.

国内学会

1. 金子 竜也, 山岸 善治, 百瀬 啓, 浅井 哲也, "エッジ AI のオンライン学習に向けたハードウェア指向対数量子化オプティマイザの提案," 電子情報通信学会複雑コミュニケーションサイエンス研究会, 北海道 ルスツリゾートホテル&コンベンション, (ハイブリッド開催), 2022 年 3 月 27 日.

2. 山岸 善治, 金子 竜也, 赤井 恵, 浅井 哲也, "エッジ学習に向けたモーメンタム最適化法のハードウェア設計," 2021 年電子情報通信学会ソサイエティ大会, (オンライン開催), 2021 年 9 月 14-17 日.

3. 金子 竜也, 山岸 善治, 百瀬 啓, 浅井 哲也, "アナログ AI デバイスのオンライン学習に向けた学習アルゴリズムとその FPGA アーキテクチャ," 第 34 回 回路とシステムワークショップ, 北九州国際会議場, (小倉), 2021 年 8 月 26-27 日.

4. 金子 竜也, 百瀬 啓, 浅井 哲也, "不揮発アナログ AI デバイス（RAND）のオンライン学習制御システム実装とその評価," LSI とシステムのワークショップ 2021, (オンライン), 2021 年 5 月 10-11 日.

5. 山岸 善治, 金子 竜也, 百瀬 啓, 赤井 恵, 浅井 哲也, "強化学習を用いたマイコン制御ロボットアーム間の物体移動評価," 2021 年電子情報通信学会総合大会, (オンライン開催), 2021 年 3 月 9-12 日.

6. 金子 竜也, 山岸 善治, 百瀬 啓, 浅井 哲也, "アナログ AI チップのオンライン学習に向けた改良型デジタルバックプロパゲーション法の提案," 第 30 回日本神経回路学会全国大会, (オンライン開催), 2020 年 12 月 2-5 日.

7. 金子 竜也, 浅井 哲也, "エッジ AI コンピューティングに向けた低電力・低リソース化学習アルゴリズムとその FPGA 実装," 東北大学電気通信研究所共同プロジェクト研究会, 東北大学電気通信研究所, (仙台), 2020 年 2 月 5 日.

8. 金子 竜也, 山岸 善治, 百瀬 啓, 浅井 哲也, “エッジ AI に向けた三値バックプロパゲーション法とその FPGA 実装,” 電子情報通信学会 非線形問題研究会, 宮古島マリンターミナル, 沖縄, 2020 年 1 月.

9. 金子 竜也, 高前田 伸也, 本村 真人, 浅井 哲也, “オンライン学習を行う階層型ニューラルネットワークハードウェアの低電力化に向けた三値バックプロパゲーション法の提案,” LSI とシステムのワークショップ 2019, 東京大学生産技術研究所, 東京, 2019 年 5 月.

10. 金子 竜也, 折茂 健太郎, 池辺 将之, 高前田 伸也, 本村 真人, 浅井 哲也, “敵対的生成ネットワークのハードウェア指向アルゴリズムとそのアーキテクチャの検討,” 2018 年電子情報通信学会 NOLTA ソサイエティ大会, 京都テルサ, 京都, 2018 年 6 月.

特許

1. 浅井 哲也, 山岸 善治, 金子 竜也, "最適化装置及び最適化方法並びに最適化用プログラム," 特願 2022-011651（2022 年 1 月 28 日）.

受賞

1. 金子 竜也, "エッジ AI に向けた三値バックプロパゲーション法とその FPGA 実装," 電子情報通信学会 非線形問題研究会 - 2019 年度非線形問題研究会発表奨励賞, 2020 年 5 月 15 日.

2. 金子 竜也, 北海道大学大学院情報科学研究院 - 2019 年度研究院長賞（修士）, 2020 年 3 月 25 日.

3. Kaneko T., Asai T., and Momose H., “Reconfigurable AI shield for embedded microcontrollers,” Maker Faire Roma 2019 – MAKER OF MERIT 2019, Oct. 2019.

4. Kaneko T., Ikebe M., Takamaeda-Yamazaki S., Motomura M., and Asai T., “Hardware-oriented algorithm and architecture for generative adversarial networks,” The Research Institute of Signal Processing – NSCP’19 Student Paper Award, Mar. 2019.