JAIST Repository
https://dspace.jaist.ac.jp/

Title	[課題研究報告書]グラフニューラルネットワークの実装技術に関する調査
Author(s)	玉川, 徹
Citation	
Issue Date	2024-09
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/19369
Rights	
Description	Supervisor: 田中 清史, 先端科学技術研究科, 修士(情報科学)

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

課題研究報告書

グラフニューラルネットワークの実装技術に関する調査

玉川徹

主指導教員　田中清史

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和 6 年 9 月

概要

ソーシャルネットワークを始めグラフ構造で表現可能な様々な関係に対して, クラス分類／回帰問題等を扱うグラフニューラルネットワーク (GNN) およびグラフ畳み込みニューラルネットワーク (GCN) が注目されている. ハードウェア実装して推論／トレーニングの速度を改善したり, 消費電力の削減をするためには, 入力として与えられるグラフを行列形式で表現した際に, 非ゼロの要素のデータ分布が非常にアンバランスである点, および, 行列サイズが非常に大きい点を考慮した設計をする必要がある. GNN ／ GCN の要素技術についてこれまで数々の研究がなされてきたが, GNN ／ GCN のハードウェア実装に関する知識の整理が不完全であるという課題がある. 本研究では, GNN ／ GCN のハードウェア実装に特に注目して, 関連するこれまでの研究について, 分類および有効な最適化手法を明らかにすることにより知識の整理を行った.

GNN ／ GCN のハードウェア実装は, 推論レイテンシの削減, 消費電力の削減, メモリアクセスの効率化等, 様々な最適化目標を持っている. また, 対象とする GNN の処理フェーズやデータセット, ネットワーク等, 前提とする条件も論文により大きく異なる. 既存研究におけるこれらの実装の傾向を明らかにするために, 各手法において対象とするフェーズ, 目的, データセット, ネットワーク, デバイス, および量子化精度に基づいて整理した. GNN ／ GCN のハードウェア実装の対象フェーズは, ①推論 (推論のみに対応) , ②トレーニング (トレーニングのみに対応), ③推論とトレーニング (推論とトレーニングの両方に対応) の 3 種類に分類される. ハードウェア実装の全体的な傾向として, フェーズについては, 推論フェーズのみを対象とする論文が 75 本, トレーニングフェーズのみを対象とする論文が 20 本, 推論／トレーニングの両方のフェーズを対象とする論文が 5 本であり, 推論フェーズを対象とする論文数が非常に多い. これは, 推論は訓練済みモデルを実際のアプリケーションで使用する段階であり, 多くの実際のアプリケーションで実行するにあたって低遅延の処理が求められていて, 推論の高速化に対する需要がトレーニングと比べて高いことが要因として考えられる. 目的については, 大半の研究がレイテンシと消費電力の削減の 2 つを挙げているが, 他にも, オフチップメモリアクセスの削減やチップ面積の削減, オフチップメモリ帯域幅使用量の削減を目的とする既存研究もある. データセットについて, ①推論の対象データセットでは, 上位 4 つの Cora, Reddit, Citeseer, Pubmed を対象としている論文は 40 本以上ありこれらが大半を占めていた. ②トレーニングのみを対象とするハードウェア実装について

は, ①と同様 Reddit と, Yelp, OGBN-products, Amazon が上位 4 つであった. ③推論／トレーニングの両方を対象とするハードウェア実装についても, Reddit, Cora, Citeseer, Pubmed 等, ①／②と同様のデータセットが採用されている. ネットワークについては GCN が極めて多く採用さていて, デバイスについては大半が ASIC あるいは FPGA である. 量子化精度については, 論文中で明示されていないものの中では特徴ベクトル／重みベクトルにおいて 32bit 固定小数点／ 32bit 浮動小数点が多く採用されている.

目次

図目次

表目次

第 1 章　はじめに

ソーシャルネットワークを始めグラフ構造で表現可能な様々な関係に対して, クラス分類／回帰問題等を扱うグラフニューラルネットワーク（GNN）[1] およびグラフ畳み込みニューラルネットワーク（GCN）[2] が注目されており, その推論速度と電力効率の改善のために様々な取り組みがされている. これらをハードウェア実装により推論／トレーニングの速度を改善したり, 消費電力の削減をするためには, 入力として与えられるグラフを行列形式で表現した際に, ①非ゼロの要素のデータ分布が非常にアンバランスである点, および, ②行列サイズが非常に大きい点を考慮した設計をする必要がある. 従来のニューラルネットワークの行列乗算に適している GPU は, 多数のスレッドを並列に実行できるアーキテクチャを持ち, 大規模な行列演算において優れた性能を発揮するが, 主に密な行列演算に最適化されているためスパース行列の非ゼロ要素の不均一な分布に対しては処理効率が低下する. また CPU は, そもそも大規模な行列演算に適したアーキテクチャではない. したがって, 従来の汎用的なハードウェアアーキテクチャはグラフ構造を効率よく処理するためには適していないため, それらをハイブリッドで実行するための仕組みを構築したり, FPGA や ASIC で実装する等, 専用の最適化が不可欠である.

非ゼロのデータ分布がアンバランスである場合, 特定の演算ユニットに対して過剰な負荷がかかる一方で, 他の演算ユニットがアイドル状態になることがある. これは, 現実世界の様々な関係をグラフ構造で表現した際に, 非ゼロの要素が特定の領域に集中し, 他の領域がほとんど空であるということが頻繁に発生することに起因する. 結果として, 全体としての計算効率が低下しリソースの無駄が生じることに加え, アイドル状態の演算ユニットが増えることに伴いエネルギー効率も低下し, 無駄な電力消費が発生する. この課題は特にスパース行列の操作において顕著であり, 効率的な計算処理を妨げる大きな要因となる. この問題を解決するためには, 負荷状況に基づいて動的にタスクの割り当て先の演算ユニットを切り替えたり, 推論／トレーニングを開始する前にグラフ特性を考慮したグラフの並び替えを行うことで, 各演算ユニットに均等に処理を分配するといった設計上の工夫を行うことが重要である. また, 非ゼロの要素の位置情報を効率的に管理するために効率的なデータ圧縮技法を検討することも有効な対策となりうる.

非常に大きな行列サイズに対処するためには, 大規模なデータを効率的に処理できるハードウェアアーキテクチャが求められる. これを実現するにあたって, まず

メモリ階層の最適化が必要である. 大規模な行列データを効率的に取り扱うためには, オンチップメモリとオフチップメモリの両方を効果的に活用することが求められる. オンチップメモリを利用して頻繁にアクセスされるデータの局所性を高め, オフチップメモリへのアクセスを最小限に抑えることで, メモリ帯域幅の制約を克服し全体のスループットを向上させることができる. また, 並列処理能力の向上も欠かせない要素である. 大規模な行列計算を効率的に行うためには, 多数の演算ユニットを並列に協調動作させるアーキテクチャを採用することで, 計算効率を大幅に向上させることができる. さらに, データの局所性を最大限に活用するためのキャッシュ階層の最適化や, データの再利用を促進するためのバッファ管理も重要な要素となる. これらのアプローチにより, メモリアクセスのオーバーヘッドを削減し, 計算処理のスループットを向上させることが可能となる.

GNN ／ GCN のハードウェア実装の既存研究について, HyGCN[9] では, GCN の推論処理が集約と結合という, 2 つの性質の異なるフェーズで構成されることに着目して, それぞれのフェーズに特化したモジュールを持つハイブリッド・アーキテクチャを提案している. AWB-GCN[10] では, 入力データである隣接行列について, 非ゼロの要素が極めて多くかつ不均一に存在するため, 不要なゼロ乗算が多く発生して計算効率が悪化するという問題に対応するために, 各演算ユニットに割り当てる負荷を分散するための自動チューニング技術を提案している. I-GCN[11] では, 推論処理における行列乗算を行うにあたってデータの局所性を高めるために, 隣接行列のデータを複数の L 字型クラスタに並び替える手法を提案している. PipeGCN[12] では, GCN データセットをサブグラフに分割して複数の GPU に割り当てて, それぞれでトレーニングを実行する分散 GCN トレーニングという手法の通信コストを改善することで, トレーニングの速度を改善する手法を提案している. また COIN[13] では, アナログ回路をメモリ上に実装することでインメモリ・アクセラレータを実現して, 推論速度や消費電力の改善をするための手法を提案している. 多くの論文が推論／トレーニングの実行時間の短縮や消費電力の削減を目的としてハードウェア実装の手法を提案しているが, それらを実現するためのアーキテクチャは多種多様である.

GNN ／ GCN の要素技術についてこれまで数々の研究がなされてきたが, GNN ／ GCN のハードウェア実装に関する知識の整理が不完全であるという課題がある. 要因として, ①新しい手法が頻繁に提案されている研究領域であること, ②入力として与えるグラフやネットワーク構造, ハードウェアデバイス種別等, 何らかの

制約条件に特化した手法が多数存在するため, 手法間で条件を揃えた状態で性能比較することが容易ではないこと, ③論文の手法を再現するために必要なハードウェアを揃えたり, 不足している技術的な詳細情報について検討した上で実装するコストが大きいこと, 等が考えられる. 以上の要因により, 対象問題／用途に対してどのようなネットワーク構成／システム構成が相応しいのかが不明瞭であるのが現状である.

本研究では, GNN ／ GCN のハードウェア実装に特に注目して, 関連するこれまでの研究について, 分類および性能との関係を明らかにすることにより GNN ／ GCN 分野の知識の整理の一助となる点で重要性がある. これにより, ①従来の手法によって解決していない問題の顕在化, ②GCN ハードウェア実装の知識を習得したい初学者の学習効率化に貢献することを目指す.

本論文の構成は, 第 2 章で GNN ／ GCN のアプリケーション例や, GNN ／ GCN の概要, GNN ／ GCN の実行プラットフォームについて論じた後, 第 3 章では既存研究の GNN ／ GCN アクセラレータの分類を行う. そして第 4 章では, 特徴的な最適化手法／スケジューリングを提案しているものをいくつか紹介して, 第 5 章の今後の課題と展望では現状のアクセラレータ群で解決されていない事柄について示す.

第 2 章　背景

2.1　GNN ／ GCN の実世界のアプリケーション例

GNN ／ GCN は, グラフ構造という汎用的なデータ構造を取り扱うネットワークであり, 多くの現実世界の関係を表現することができるため, 実社会での応用範囲が非常に広く, 今後適用される領域はさらに広がっていくことが予想される. 本節では, GNN ／ GCN の様々な適用・応用例を示す.

ソーシャル領域について, SocialGCN[14], MSGCN[15] では, ユーザーの過去の行動履歴（レビューや「いいね」等）と, ユーザー間のソーシャルネットワーク構造（フォロー関係等）を活用して, ユーザーが好むと予測されるコンテンツ（店舗, 画像, 他の利用ユーザー等）を推薦する手法を提案している. CayleyNets[16], GUCD[17] では, 電力網運用において, ソーシャルネットワーク等のグラフ構造における共通の趣味や属性を持つノードの集合を発見するための手法を提案している. NESA[18], DTGRU[19] では, ソーシャルネットワークの世論識別等を目的として各テキストの感情分析をするために GCN を適用している. また, M-GCN[20], KMGCN[21], KMAGCN[22] では, フェイクニュースの検出のために GCN が活用されている.

交通領域について, ASTGCN[23], T-MGCN[24] では, 道路網の交通状況の空間的依存関係, 時間的依存関係等をもとに交通流予測をするための手法を提案している. この手法は, 渋滞の予測, 事故リスクの予測, 公共交通機関/タクシーの運行管理等への応用が想定される.

電力インフラ領域について, GCMCN[24] では, 風力発電において, 風速, 風向といった空間的特徴量と, 時間的特徴量（時系列データ）を同時に考慮した風力発電量の予測手法を提案している. Kim 他 [26] は, 送電線が不測の事態で過負荷になるのを防ぐための最適な負荷遮断比を予測する手法を示している.

創薬領域について, Decagon[27] では, 多数の種類の薬剤を同時併用することによる副作用のリスクが臨床試験で見逃されやすく予測が難しいという課題を解決するために, 副作用予測をするための手法を提案している. Ma 他 [28] では, 薬物間の類似性を予測するために, 化学構造, 薬物間相互作用, 副作用等, 様々なドメインから得られる薬物の情報をもとにグラフ構造を構築してトレーニングして, 解釈性の高い予測モデルを構築している. Gao 他 [29] では, ある薬物分子の化学構造式と

タンパク質のアミノ酸配列を入力として, その組み合わせが結合する可能性が高いかどうかを予測する手法を提案している.

金融領域について, SemiGNN[30], TA-Struc2Vec[31] では, ユーザー間の関係性や属性情報等を統合して, グラフ構造に基づいた金融詐欺の検出をするための手法を提案している.

以上のように, グラフデータを扱う技術の進化と共に GNN ／ GCN の適用領域は今後さらに広がっていき, 様々な産業や研究分野における新しい応用が次々と登場することが期待される.

2.2 GNN ／ GCN の基本概念と定義

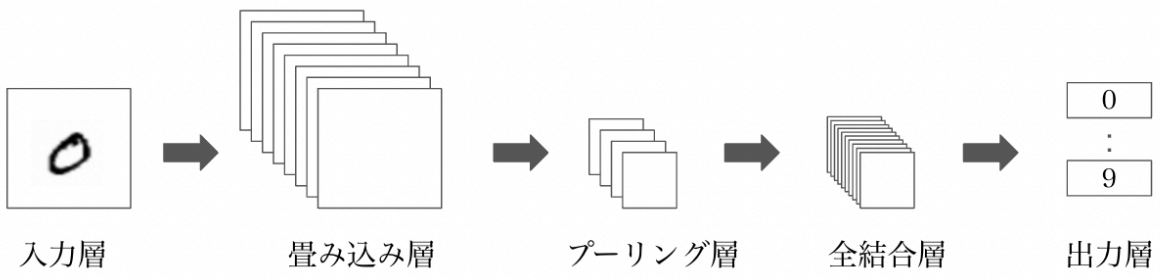

図 1: CNN の全体像の簡易図

GNN は, CNN[32] における畳み込み演算を非構造グラフデータに適用できるように一般化したものである.

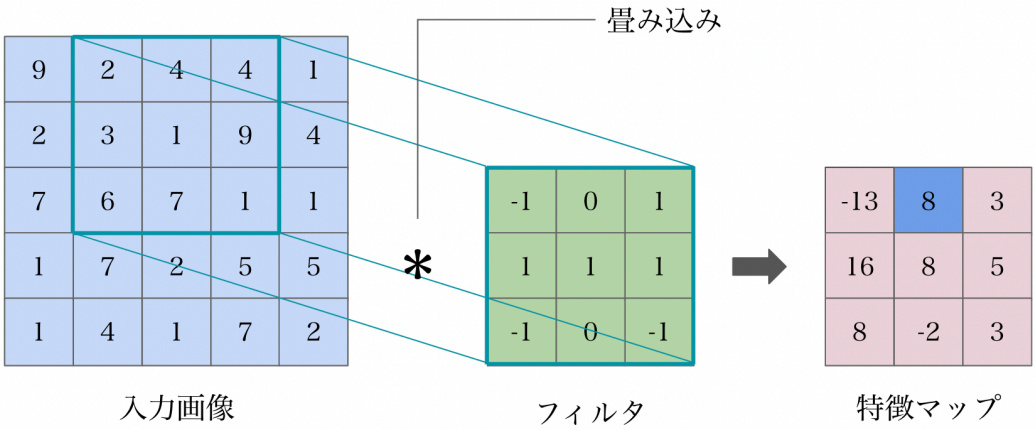

図 2: 入力画像から特徴マップの生成

前提として, CNN は主に画像認識や画像解析の分野で活用されるディープラーニングのアルゴリズムで, 画像データから特徴を抽出し, 分類や検出等のタス

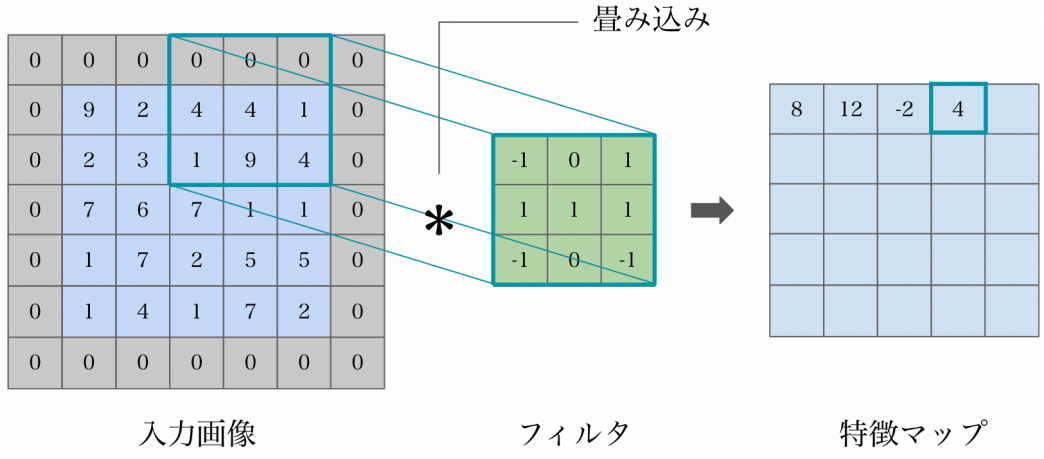

図 3: ゼロパディング

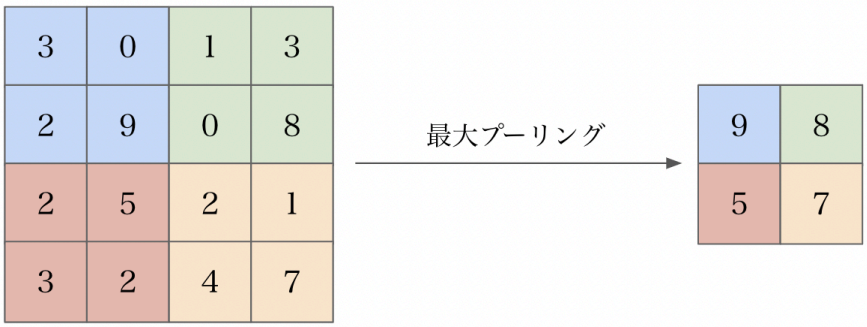

図 4: 最大プーリング

クを行う用途で用いられる. CNN は主に 5 つの層（入力層, 畳み込み層, プーリング層, 全結合層, 出力層）で構成される（図 1）. 入力層では, 画像データがピクセル値としてネットワークに与えらえる. 畳み込み層では, フィルタ（カーネル）を用いて入力データに対してフィルタのウィンドウを一定の間隔（ストライド）でスライドさせながら, フィルタの要素と入力データの対応する要素を乗算してそれらの和を求め（積和演算）, 出力結果の対応する場所（左上から順）に格納する. 図 2 において, 特徴マップの青背景上の 8 は以下のように計算される: $8 = 2 \times -1 + 4 \times 0 + 4 \times 1 + 3 \times 1 + 1 \times 1 + 9 \times 1 + 6 \times -1 + 7 \times 0 + 1 \times -1$. 畳み込み演算の結果として, 入力画像から特徴マップが生成され, 画像の局所的な特徴が強調される. ただし, 畳み込みを行うとカーネルが入力の端まで到達できないことに伴い画像のサイズが徐々に小さくなり, 入力の端に近い情報が失われる. これを防ぐためにパディング（図 3）という手法により入力画像のピクセルの周りにピクセルを囲むことで, 入力の端の情報を保持しながら、畳み込み演算を適用する. 次にプーリング層では, 特徴マップのサイズの縮小を行なうことで全結合層で必要な計

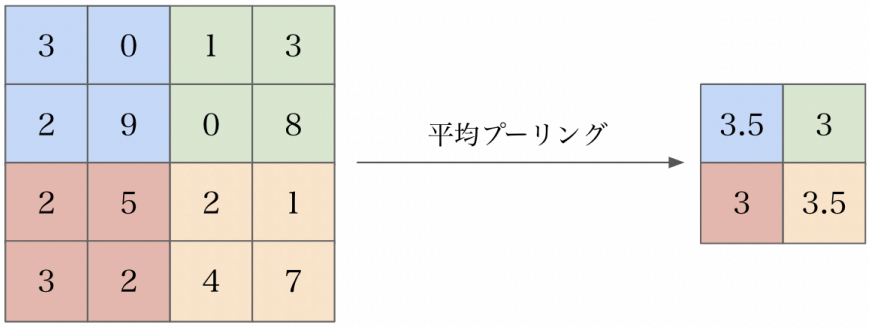

図 5: 平均プーリング

算量の削減を行う. 一般的なプーリング手法としては, 最大プーリング（図 4）や平均プーリング（図 5）がある. 最大プーリングでは領域内で最も大きな値が選択され, 平均プーリングでは, 領域内のピクセル値の平均値が算出される. これらにより, 特徴マップの重要な情報を保持しつつ, データのサイズが圧縮される. 続いて全結合層で特徴マップによって得られたそれぞれの特徴量を 1 つのノードに集約を行った後, 出力層において活性化関数を適用することで最終的な結果が得られる.

CNN は, 主に画像データの処理に優れているが, いくつかの問題点がある. まず, CNN は規則的なグリッド状のデータ（画像等）に適しているが, グラフのような多次元構造には適していない. 画像データは, ピクセルが規則的に配置された 2 次元のグリッド構造を持っていて, CNN は, このような規則的な構造を活かして局所的な特徴を効率的に抽出することができる. 畳み込み層では, フィルタを用いて画像の局所的な特徴を抽出し, プーリング層では, 特徴量を圧縮・要約することで画像の重要な特徴を効果的に抽出する. しかし, グラフデータは多次元の構造を持ちノード間の関係性が複雑であるため, CNN のような規則的なグリッド構造に基づく処理には適していない. グラフデータでは, ノード数や接続関係が一定ではなく, ノード間の距離も一様ではない. さらに, グラフデータにはノードの順序に意味がないという性質, つまり, ノードの並び方を変えてもグラフの構造や意味が変わらないという性質がある. 例えば, ソーシャルネットワークのグラフで, ユーザー A とユーザー B が友人関係にあるとき, 「A と B が友人である」ことと「B と A が友人である」ことは同じ意味を持つ. この性質に伴い, グラフデータを扱う際にはノードの順序を考慮する必要がない. しかし, CNN は入力データの空間的な配置を考慮して特徴量を学習するため, ノードの順序に意味がないグラフデータでは CNN で生成する特徴量が意味を持たない. また, CNN は固定サイズの入力を想定しているため, ノード数が可変であるグラフデータを直接扱うことは難しい. このように, グ

ラフデータの持つ多次元構造, ノード数や接続関係の可変性, ノードの順序に意味がないという性質により, CNN をグラフデータに直接適用することは困難である.

CNN の課題を解決するために, グラフ構造を直接扱うことのできるニューラルネットワークとして, GNN が提案された. GNN では, 各ノードの特徴量を更新する際に隣接するノードの情報を集約することで, グラフ構造に応じた柔軟な特徴量の更新が可能になる.

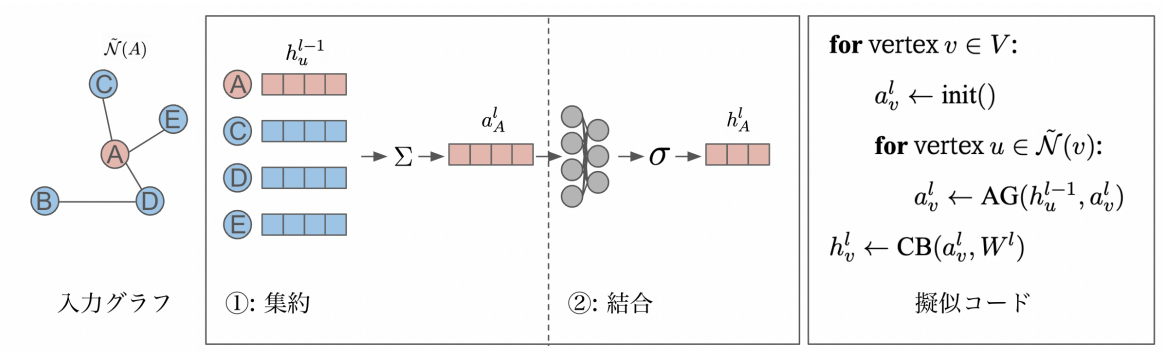

図 6: GNN の推論の流れ [33]

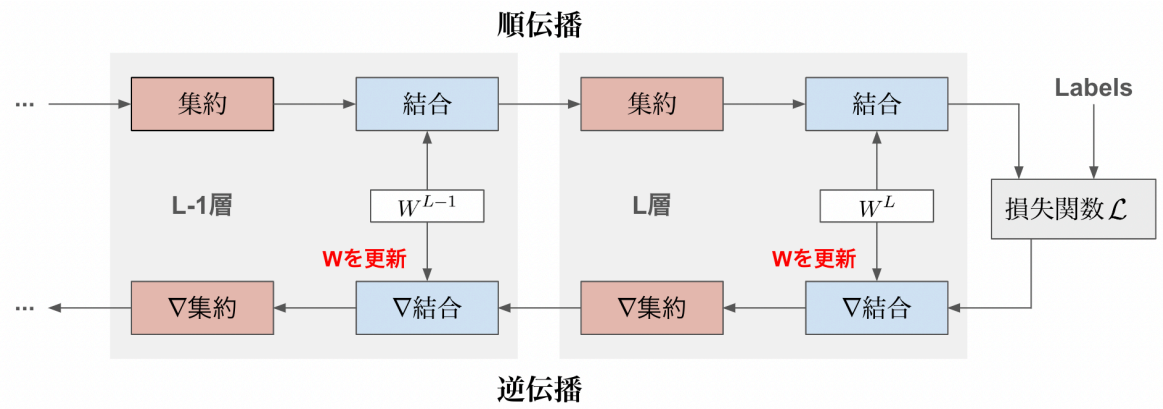

図 7: GNN のトレーニングの流れ [33]

一般に, GNN モデルは入力グラフ $G = (V, E)$ が与えられたとき（V と E は頂点と辺の集合である）, 以下のように抽象化される 2 段階の演算（式 1: 集約, 式 2: 結合）を行う.

$$a_v^l = \mathrm{AGGREGATE}\left(h_u^{l-1} \mid u \in \tilde{\mathcal{N}}(v)\right) \tag{1}$$

$$h_v^l = \mathrm{COMBINE}(a_v^l) \tag{2}$$

GNN の推論では, 順伝播の計算を畳み込み演算として行う. 順伝播は, ノードの特徴量を隣接ノードの特徴量から集約, 結合することでノードの新しい特徴量を計算するプロセスである（図 6）. GNN のトレーニングでは, 順伝播の計算, 損失計算, 逆伝播の計算, 重みの更新をモデルが収束するまで行う（図 7）.

表 1. GCN アルゴリズムに関する表記と説明 [33]

表記	説明
V, E	ノードとエッジの集合
$\mathcal{N}(v)$	ノード v の隣接ノードの集合
$\tilde{\mathcal{N}}(v)$	v と v の隣接ノードを含む集合: $\{\mathcal{N}(v)\} \cup \{v\}$
D_v	ノード v の次数
L	モデルの総レイヤー数
h_v^l	第 l 層のノード v の隠れ特徴ベクトル
d_v^l	第 l 層の隠れ特徴ベクトルの次元
a_v^l	第 l 層のノード v の集約された特徴ベクトル
W^l	第 l 層の重みベクトル
z_v^l	活性化関数適用前の特徴ベクトル, すなわち $a_v^l \cdot W^l$
σ	活性化関数, 例: ReLU
$\mathcal{L}$	出力とラベルを用いて計算された損失値
δ_v^l	h_v^l の勾配, すなわち $\frac{\partial \mathcal{L}}{\partial h_v^l}$
$\hat{\delta}_v^l$	h_v^l のマスクされた勾配: $\delta_v^l \odot \sigma'(z_v^l)$
$\hat{\delta}_v'^l$	ノード v の集約・マスク済勾配

GNN は, グラフ構造データを扱うニューラルネットワークの総称で, 様々なアーキテクチャが存在するが, その中でも代表的なものの 1 つが GCN である. GCN では, 入力グラフの各頂点に自己ループを追加することで, 隣接ノードだけでなく自身のノードも計算に加える実装になっている. GCN のアルゴリズムの議論で使用する表記を表 1 に示す.

具体的な計算について, まず集約フェーズ（式 3）では, 各ノード v に対して、その隣接ノードの特徴量 h_u^{l-1} を集約する. 結合フェーズ（式 4）では, 集約された特徴 a_v^l に対して、重み行列 W^l を適用し, 活性化関数 σ を通して新しいノード特徴 h_v^l を計算する.

$$a_v^l = \sum_{u \in \tilde{\mathcal{N}}(v)} \frac{1}{\sqrt{D_v D_u}} h_u^{l-1} \tag{3}$$

$$h_v^l = \sigma(a_v^l \cdot W^l) \tag{4}$$

次に, 損失計算について, GCN の代表的なタスクであるノード分類では交差エントロピー誤差を適用する（式 5）.

$$\text{正解ラベルとの誤差を算出} \quad \mathcal{L} = \frac{1}{N}\sum_{v=1}^{N} CrossEntropyLoss(h_v^L, y_v) \tag{5}$$

さらに, 逆伝播の計算（式 6）で, 重みの勾配を計算する.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W^l} &= \sum_{v=1}^{N} \frac{\partial z_v^l}{\partial W^l} \cdot \frac{\partial \mathcal{L}}{\partial z_v^l} = \sum_{v=1}^{N} {a_v^l}^T \cdot \frac{\partial \mathcal{L}}{\partial h_v^l} \cdot \frac{\partial h_v^l}{\partial z_v^l} \\ &= \sum_{v=1}^{N} {a_v^l}^T \cdot \delta_v^l \odot \sigma'(z_v^l) = \sum_{v=1}^{N} {a_v^l}^T \cdot \hat{\delta}_v \end{aligned} \tag{6}$$

最後に, 重みの更新を行う（式 7）.

$$W^l = W^l - \eta \frac{\partial \mathcal{L}}{\partial W^l} \tag{7}$$

GNN には GCN 以外にも様々なアーキテクチャがあり, 代表的なものとして, GIN(Graph Isomorphism Network)[34], GraphSAGE(Graph Sample and Aggregation)[35], GAT(Graph Attention Network)[36] が挙げられる. それぞれのアーキテクチャは異なる方法でノード間の情報を集約・結合することで, グラフデータの処理を行う (表 2).

表 2. GNN の派生形 [33]

派生形	集約	結合
GCN[1]	$a_v^l = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{D_v D_u}} h_u^{l-1}$	Relu$(a_v^l \cdot W^l)$
GIN[34]	$a_v^l = (1+\epsilon) \cdot h_v^l + \sum_{u \in \mathcal{N}(v)} h_u^l$	MLP(a_v^l, W^l, b^l)
SAGEConv[35]	$a_v^l = \text{Mean}(h_u^l \| u \in \mathcal{N}(v))$	Relu$(a_v^l \cdot W^l)$
GAT[36]	$\alpha_{vu} = \text{Softmax}(\sigma(a^T[W_1^l h_v^l \| \| W_2^l h_u^l]))$ $a_v^l = \sum_{u \in \mathcal{N}(v)} \alpha_{vu} \cdot h_u^l$	Elu$(a_v^l \cdot W^l)$

2.3 GNN ／ GCN の実行プラットフォームの概要

GNN ／ GCN の実行プラットフォームである, CPU, GPU, FPGA, ASIC のそれぞれのデバイスの特徴, 優位性について示す. 図 8 では, 各ハードウェアアーキテクチ

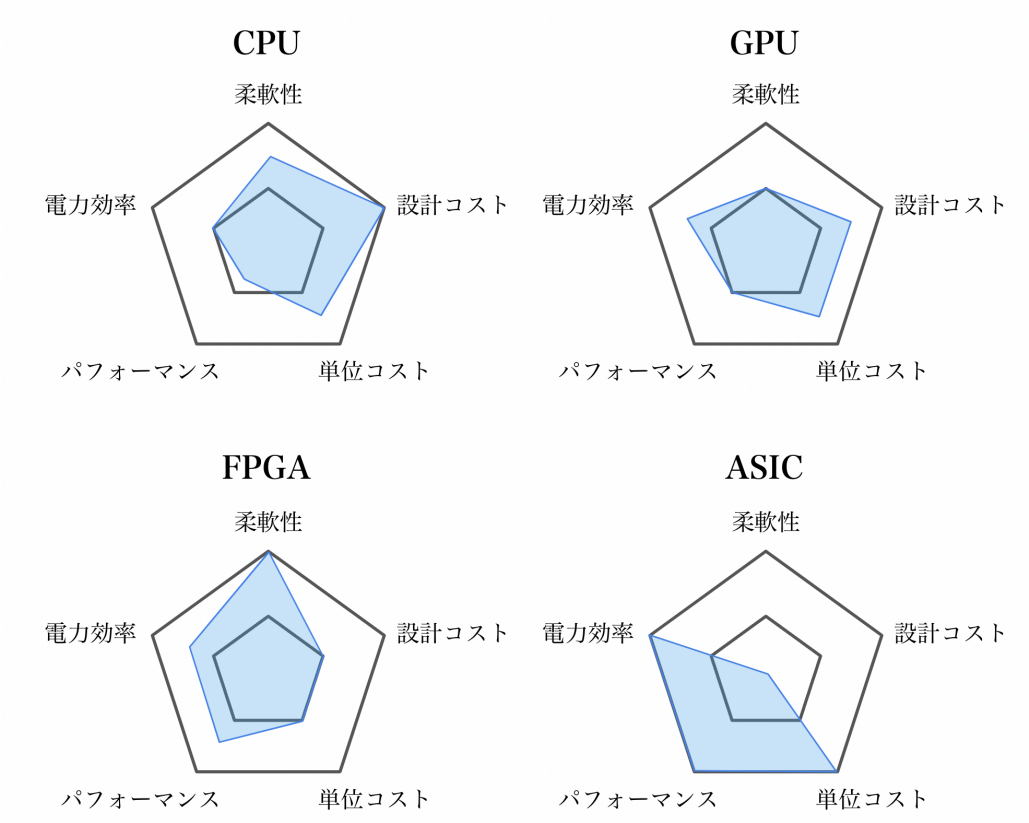

図 8: 各ハードウェアアーキテクチャの比較 [37]

ャがいくつかのパラメータで評価されている.

- Flexibility（柔軟性）: 様々なユースケースに対する適応性の高さを表す指標.
- Design Cost（設計コスト）: 設計に必要なコストを表す指標.
- Unitary Cost（単位コスト）: 完成品 1 つあたりの製造コスト（量産時のコスト）を表す指標.
- Performance（パフォーマンス）: 与えられたタスクに対するレイテンシやスループット等を表す指標.
- Power Efficiency（電力効率）: 実行に必要な電力を表す指標.

CPU(Central Processing Unit) は, 特定のアプリケーションに特化せずに汎用的な用途で使うことを前提に設計された汎用プロセッサである. CPU は図 9(a) のように, 主に制御ユニット, コア (ALU: 算術論理演算ユニット), DRAM(Dynamic Random Access Memory), キャッシュにより構成される. CPU は基本的に命令を 1 つずつ順番に実行するシーケンシャルな処理に最適化されていて, 多様なタスクを処理できるよう汎用性を重視して設計されている. プログラミングの容易/柔軟性

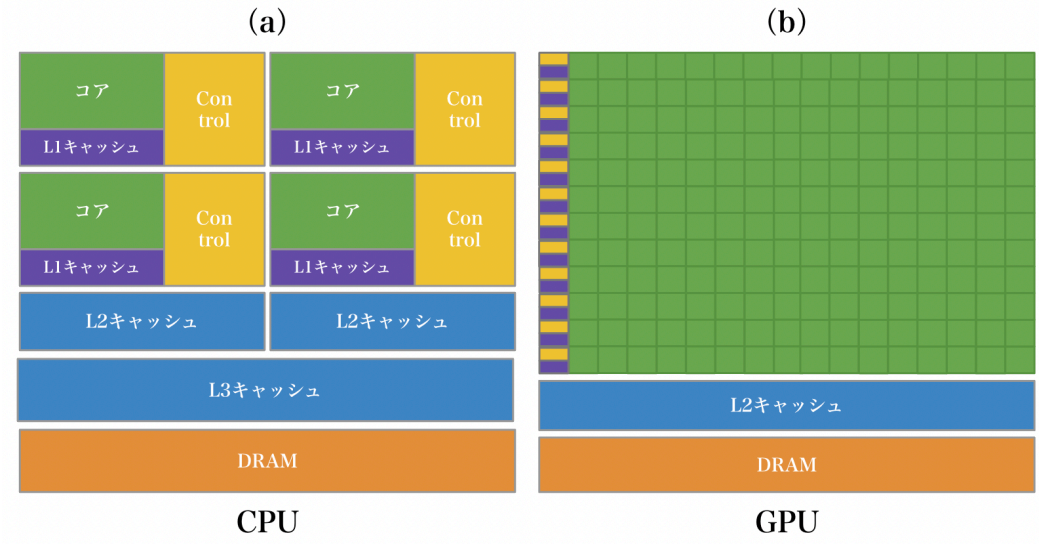

図 9: CPU と GPU の比較 [38]

さもあり, 最も汎用的なプラットフォームとして広く使われてきた. しかし, ディープラーニングのような大規模な行列乗算のタスクを実行するにあたっては, CPU のコア数は GPU 等と比べて少ないため並列処理を行うにあたってスループットが非常に悪化する. そのため, ディープラーニングの領域では他のアーキテクチャのハードウェアが適用されることが多い.

GPU(Graphics Processing Unit) は, ある程度の柔軟性を維持しながら特定の計算タスク（例えば, 3D グラフィックのレンダリング）を改善する用途で設計されたプロセッサである. しかし, ディープラーニングの計算要求と非常に適合していることから、現在では大規模な行列乗算用途でも積極的に使われている. CPU の面積の大半は制御ユニット, キャッシュ, DRAM が大部分を占めているが, 図 9(b) にあるように GPU はコア数が非常に多い. この構造は大規模な行列乗算に適していて, 広く使用されるようになっている. GPU が広く使われるようになるのに伴い, Tensorflow や Pytorch 等, GPU を使った処理に最適化されたライブラリが盛んに開発されるようになっている.

FPGA(Field Programmable Gate Array) は, 論理回路を後から変更できることが特徴のプログラマブルなハードウェアである. CPU や GPU のようなハードウェアの場合, 既に論理回路設計が行われていて後から変更することはできない. しかし, FPGA はそれが可能であり, また特定の命令セットも存在しないことに伴い命令セットの解釈に必要な処理を実装する必要がないため, CPU ／ GPU より消費電力を削減できる可能性がある. ただし, CPU ／ GPU に比べて設計工数がかかるという

デメリットがある.

ASIC(Application Specific Integrated Circuit) は, 直訳すると特定用途向け集積回路で, 特定のアプリケーションのために最適化された回路を持つカスタムハードウェアである. ASIC は特定の用途に最適化されていて不要な回路を持たないため, 他のハードウェアと比べて消費電力を抑えやすく, また同じ機能をより小さなチップサイズで実現することができる. ただし, 完全にカスタマイズで開発をすることになるので, 設計から製造まで多大な時間とコストがかかったり, 一度製造すると回路を変更できないというデメリットもある.

第 3 章　GNN ／ GCN アクセラレーションの分類

3.1　分類項目の定義

GNN ／ GCN のハードウェア実装は, 推論レイテンシの削減, 消費電力の削減, メモリアクセスの効率化等, 様々な最適化目標を持っている. また, 対象とする GNN の処理フェーズやデータセット, ネットワーク等, 前提とする条件も論文により大きく異なる. 本章では, 既存研究におけるこれらの実装の傾向を明らかにするために, 各手法において対象とするフェーズ, 目的, データセット, ネットワーク, デバイス, および量子化精度に基づいて整理する. 各分類項目の定義は次の通りである.

- ID：研究の識別番号
- 論文：方式名または著者名
- フェーズ：GNN の処理における対象フェーズ（例：推論, トレーニング）
- 目的：ハードウェア実装の主な評価観点（例：推論レイテンシの削減, 消費電力の削減等）
- データセット：使用されるデータセットの種類（例：表 3）
- ネットワーク：使用される GNN アーキテクチャの種類
- デバイス：使用されるデバイスの種類（ASIC, FPGA, CPU, GPU）
- 量子化精度：量子化における精度（例：32bit 固定小数点, 16bit 浮動小数点）

3.2　分類表と考察

GNN ／ GCN のハードウェア実装の対象フェーズは, ①推論（推論のみに対応）, ②トレーニング（トレーニングのみに対応）, ③推論とトレーニング（推論とトレーニングの両方に対応）の 3 種類に分類される. GNN ／ GCN ハードウェア実装の既存研究 100 本を, 対象フェーズ毎に 3.1 節の分類項目に基づいて分類した結果をそれぞれ表 4, 表 5, 表 6 にて示す.

ハードウェア実装の全体的な傾向として, フェーズについては, 推論フェーズのみを対象とする論文が 75 本, トレーニングフェーズのみを対象とする論文が 20 本, 推論／トレーニングの両方のフェーズを対象とする論文が 5 本であり, 推論フェーズを対象とする論文数が非常に多い. これは, 推論は訓練済みモデルを実際の

表 3. GNN ／ GCN のハードウェア実装で用いられる代表的なベンチマーク用データセット. クラスは, データセット内の分類のカテゴリを表す. 例えば Cora の場合, 強化学習, ニューラルネットワーク, 事例ベース, 遺伝的アルゴリズム, 確率的手法, ルール学習, 理論という 7 つの論文カテゴリが存在することを意味する. 特徴量は, 各ノードに対して付与されていて, ノードの特徴を表す. 例えば Cora の場合, 特徴量は論文要旨の単語の集合を表していて, 使用される語彙の数は 1,433 である.

データセット	説明	クラス	ノード	エッジ	特徴量
Cora [3]	機械学習関連の論文の引用関係を表すデータセット	7	2,708	5,429	1,433
Citeseer [3]	情報科学関係の論文の引用関係を表すデータセット	6	3,327	4,732	3,703
Pubmed [3]	糖尿病に関する医学論文の引用関係を表すデータセット	3	19,717	44,338	500
PPI [4]	タンパク質間の関係を表すデータセット	121	14,755	225,270	50
Flickr [5]	画像共有サイトにおけるアノテーション付き画像のデータセット	7	89,250	899,756	500
Yelp [5]	レビューサイトのレビューやユーザに関するデータセット	100	716,847	6,977,410	300
Reddit [6]	フォーラムサイトの投稿の関係を表すデータセット	41	232,965	11,606,919	602
Amazon [7]	E コマースサイトの商品の購入情報を表すデータセット	107	1,598,960	132,169,734	200
OGBN-Products [8]	E コマースサイトにおけるよく一緒に購入される商品関係を表すデータセット	47	2,449,029	61,859,140	100

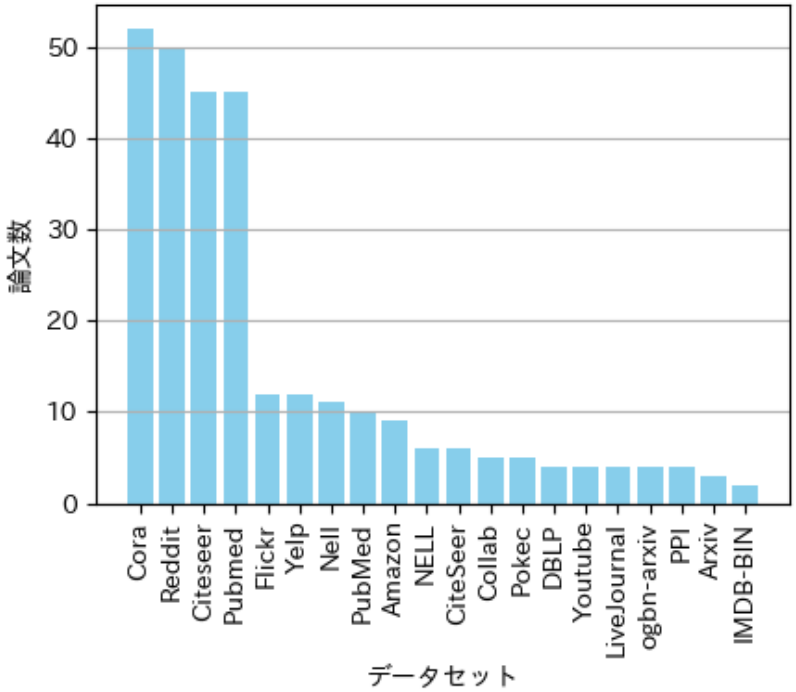

図 10: 推論のみを対象とする GNN ／ GCN ハードウェア実装のデータセット毎の論文数

アプリケーションで使用する段階であり, 多くの実際のアプリケーションで実行するにあたって低遅延の処理が求められていて, 推論の高速化に対する需要がトレーニングと比べて高いことが要因として考えられる. また, 高速化以外でも, 例えばモバイルデバイスや IoT デバイスにおいては電力効率が重要な課題であり, 主に推論フェーズが実行される. これらのデバイスでは, 電力効率の最適化は実行時の省電力化に直接寄与するため研究分野としての需要が大きいことも要因の 1 つとして挙げられる. 多くの論文では, 推論かトレーニングかいずれかのフェーズに最適化することを目標としていて, 両方に対応する論文は少ない.

目的については, 大半の研究がレイテンシと消費電力の削減の 2 つを挙げているが, 他にも, オフチップメモリアクセスの削減（HyGCN[9], AWB-GCN[10], GCNAX[41], Zhang 他 [43], I-GCN[11], ReGNN[55] 等）やチップ面積の削減（ReGNN[55], COIN[13], FusedGCN[79]）, オフチップメモリ帯域幅使用量の削減（GCoD[48], G-CoS[49], Accel-GCN[101]）を目的とする既存研究もある. ただし, オフチップメモリアクセスの削減, チップ面積の削減, オフチップメモリ帯域幅使用量の削減等, いずれもレイテンシと消費電力の削減という最終目標を達成するための定量指標としての位置付けになっていると解釈できる.

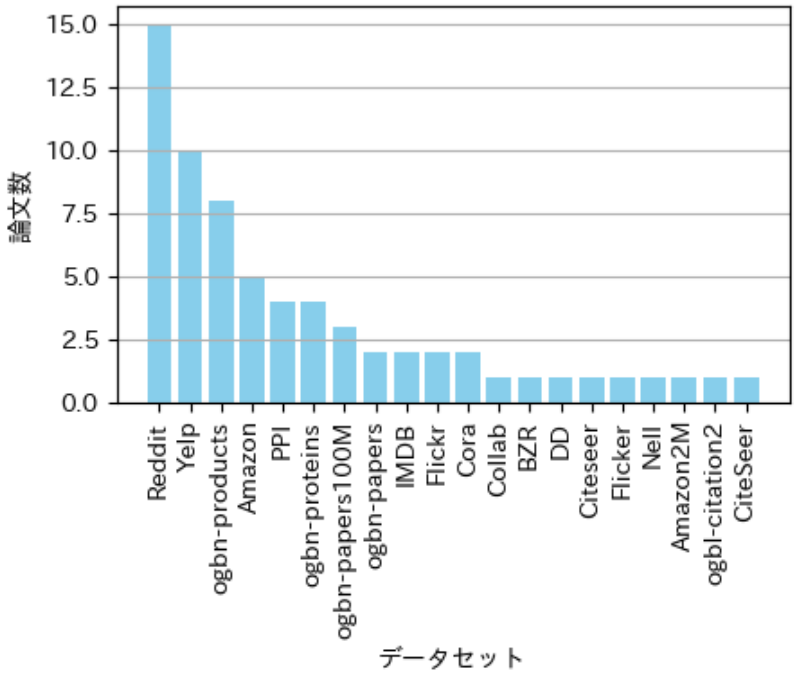

図 11: トレーニングのみを対象とする GNN ／ GCN ハードウェア実装のデータセット毎の論文数

データセットについて, 図 10, 図 11, 図 12 で①推論, ②トレーニング, ③推論／トレーニングの両方の 3 つの対応フェーズ毎に上位 20 件の対象データセットを示す. ①推論の対象データセットについて, 上位 4 つの Cora, Reddit, Citeseer, Pubmed を対象としている論文は 40 本以上ありこれらが大半を占めていて, 次に Flickr, Yelp, Nell, PubMed はそれぞれ 12 本, 12 本, 11 本, 10 本あり, これら以外のデータセットを採用する論文数は全て 10 本より少ない. ①推論のみを対象とするハードウェア実装の上位 4 つのデータセットのノード／エッジ数について, 表 3 の通り, Cora はノード数が 2,708 でエッジ数が 5,429, Reddit はノード数が 232,965 でエッジ数が 11,606,919, Citeseer はノード数が 3,327 でエッジ数が 4,732, Pubmed はノード数が 19,717 でエッジ数が 44,338 というように, Reddit を除けば比較的ノード数／エッジ数が少ないデータセットが多用されている. ②トレーニングのみを対象とするハードウェア実装については, ①と同様 Reddit と, Yelp（ノード数：716,847 ／エッジ数：6,977,410）, OGBN-products（ノード数：2,449,029 ／エッジ数：61,859,140）, Amazon（ノード数：1,598,960 ／エッジ数：132,169,734）が上位 4 つで, GNN ／ GCN のハードウェア実装論文で取り扱うデータセットの中でも特にデータ量が多いものを採用している. これは, 推論と比べてトレーニングを対象とする論文においては大規模なデータセットを積極的に使うことでスループ

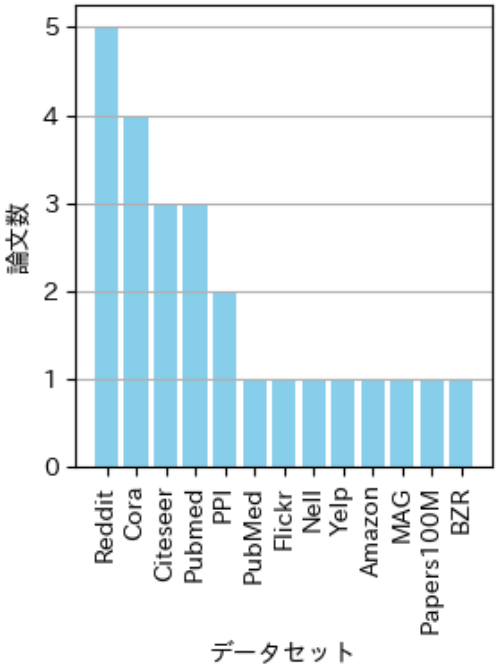

図 12: 推論／トレーニングの両方を対象とする GNN ／ GCN ハードウェア実装のデータセット毎の論文数

ットの改善をすることを重視していることが要因だと考えられる. ③推論／トレーニングの両方を対象とするハードウェア実装についても, Reddit, Cora, Citeseer, Pubmed 等, ①／②と同様のデータセットが採用されている.

ネットワークについては, 図 13 で先述の①, ②, ③ の 3 つの対応フェーズ毎に上位 10 件の対象ネットワークを示す. これら 3 つ全てにおいて最も使用されているネットワークは GCN で, それぞれ 68 本, 14 本, 5 本と, 特に①において非常に多くの論文で採用されている. また全ての対応フェーズにおいて 2 番目に採用されているのは GraphSAGE であり, それぞれ 30 本, 13 本, 1 本である. ①で 3 番目, 4 番目に採用されているのは, GIN, GAT で, それぞれ 23 本, 21 本で, ②で 3 番目, 4 番目に採用されているのは, GAT, GIN でそれぞれ 6 本, 5 本である. ネットワークについては, データセットとは異なり, 推論とトレーニングのフェーズによって採用するネットワークの性質が大きく異なるということはなかった.

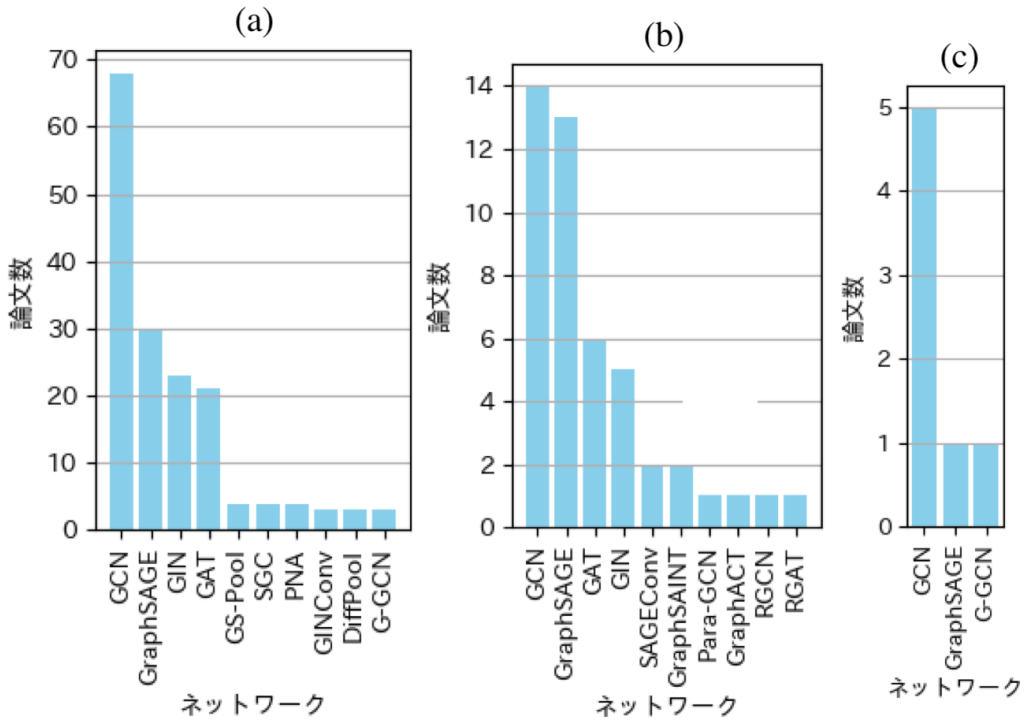

図 13: GNN ／ GCN ハードウェア実装のネットワーク毎の論文数 (a) 推論が対象 (b) トレーニングが対象 (c) 推論／トレーニングの両方が対象

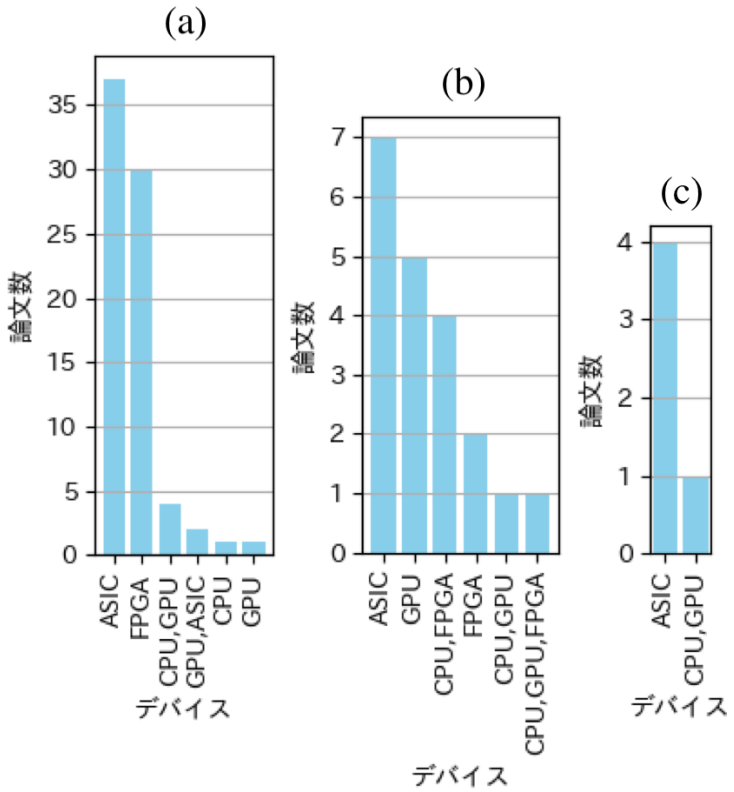

図 14: GNN ／ GCN ハードウェア実装のデバイス毎の論文数 (a) 推論が対象 (b) トレーニングが対象 (c) 推論／トレーニングの両方が対象

デバイスについて, 図 14 で対応フェーズ毎の対象デバイスにおける論文数を示す. 推論のみが対象の論文について, 使用されているデバイスの大半が ASIC と FPGA であり, 全体の 89.3% を占めている. 他にも, 計算特性に応じて CPU ／ GPU を使い分けている論文（Zhang 他 [73], MergePathSpMM[112], SEIGN[101], TGOpt[103]）, GPU ／ ASIC を使い分けている論文（Jin[88], Slice-and-Forge[89]）がある. トレーニングのみが対象の論文については, ASIC が 7 件と最も多く, 次に GPU が 5 件, CPU ／ FPGA が 4 件と続く. 推論のみを対象とする論文では GPU のみを採用している論文は 1 件しかないが, トレーニングのみを対象とする論文は 5 件あった. 推論／トレーニングの両方が対象の論文では, 5 件中 4 件が ASIC である.

量子化精度については, 論文中で明示されていないものも半数近くあるが, 明示されているものの中では, 特徴ベクトル／重みベクトルにおいて 32bit 固定小数点／ 32bit 浮動小数点が多く採用されていて, 中には 16bit, 8bit を採用している論文もある. さらに精度が低い論文として, 4bit 固定小数点を採用している手法（COIN[13]）, 1bit を採用している手法（Bi-GCN[128]）もある.

表 4. 推論のみを対象とする GNN ／ GCN ハードウェア実装の既存研究の分類

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
1	HyGCN[9]	• レイテンシ削減 • 消費電力削減 • オフチップメモリアクセス数の削減 • DRAM 帯域幅利用率の向上	• IMDB-BIN • Cora • Citeseer • COLLAB • Pubmed • Reddit	• GCN • Graph-SAGE • GINConv • DiffPool	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
2	AWB-GCN[10]	• レイテンシ削減 • 消費電力削減 • オフチップメモリアクセス数の削減 • メモリ帯域幅利用率の向上 • Adaptive Logic Modul 使用量の削減 • オンチップストレージ需要の削減 • PE 使用率の向上	• Cora • Citeseer • Pubmed • Nell • Reddit	• GCN	• FPGA	N/A
3	EnGN[39]	• レイテンシ削減 • 消費電力削減 • チップ面積削減 • I/O コスト削減	• Cora • PubMed • Nell • CoraFull • Reddit • Enwiki • Amazon • Synthetic-A/B/C/D • AIFB • MUTAG • BGS • AM	• GCN • GS-Pool • Gated-GCN • GRN • R-GCN	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
4	Auten 他 [40]	• レイテンシ削減 • メモリ帯域幅利用率の向上	• Cora • Citeseer • Pubmed • QM9 • DBLP	• GCN • GAT • MPNN • PGNN	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
5	GC-NAX[41]	• レイテンシ削減 • オフチップメモリアクセス数の削減 • 消費電力削減 • チップ面積削減	• Cora • Citeseer • Pubmed • Nell • Reddit	• GCN	• ASIC	• 特徴ベクトル: 64bit 倍精度浮動小数点数 • 重みベクトル: 64bit 倍精度浮動小数点数
6	GRIP[42]	• レイテンシ削減 • オンチップメモリ容量の削減	• Youtube • Livejournal • Pokec • Reddit	• GCN • Graph-SAGE • G-GCN • GIN	• ASIC	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 16bit 固定小数点
7	Zhang 他 [43]	• レイテンシ削減 • オフチップメモリアクセス数の削減	• Flickr • Reddit • Yelp	• GCN	• FPGA	N/A
8	I-GCN[11]	• レイテンシ削減 • オフチップメモリアクセス数の削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Nell • Reddit	• GCN • Graph-SAGE • GIN	• FPGA	N/A

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
9	Poly-graph[44]	• レイテンシ削減 • オフチップメモリアクセス数の削減 • 消費電力の削減 • チップ面積削減	• Orku • LiveJournal • Twitter • IndoChina • RdUSE • RdUSW • CF • Big-mlens • Mlens • Pubmed • Cora	• GCN	• FPGA	N/A
10	Boost-GCN[45]	• レイテンシ削減 • オフチップメモリアクセス数の削減 • DRAM レイテンシの削減	• Flickr • Reddit • Yelp • Amazon	• GCN	• FPGA	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
11	Block-GNN[46]	• レイテンシ削減 • 消費電力削減	• Cora • Citeseer • Pubmed • Reddit	• GCN • G-GCN • GS-Pool • GAT	• FPGA	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
12	Deep-Burning-GL[47]	• レイテンシ削減 • 消費電力削減	• Cora • Pubmed • Reddit • AIFB • MUTAG • BGS • AM • ModelNet	• GCN • GC-Pool • R-GCN • EdgeConv	• FPGA	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 16bit 固定小数点
13	GCoD[48]	• レイテンシ削減 • オフチップメモリ帯域幅使用量の削減 • オフチップメモリアクセス数の削減	• Cora • Citeseer • Pubmed • NELL • OGBN-arxiv • Reddit	• GCN • GIN • Graph-SAGE • GAT • ResGCN	• FPGA	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 8bit 固定小数点
14	G-CoS[49]	• レイテンシ削減 • オフチップメモリ帯域幅使用量の削減	• Cora • Citeseer • Pubmed • Reddit	• GCN • GAT • GIN • Graph-SAGE	• FPGA	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 16bit 固定小数点
15	Chen 他 [50]	• レイテンシ削減 • processing unit の利用率の向上	• Blogcatalog • Citeseer • Coauthor • DBLP • Pubmed	• DAGNN • GCN • GWNN • SGC	• CPU	N/A
16	Huang 他 [51]	• レイテンシ削減 • スループットの向上消費電力の削減	• Cora • Citeseer • Pubmed • Nell • Reddit	• GCN • GIN • GAT • Graph-SAGE	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
17	Tailor 他 [52]	• レイテンシ削減 • オフチップメモリ使用量の削減	• Reddit • Code • Arxiv • ZINV	• GCN • GIN • Graph-SAGE • GAT • MPNN • PNA • EGC	• ASIC	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
18	Cambricon-G[53]	• レイテンシ削減 • キャッシュミス率の改善 消費電力の削減	• YouTube • Flickr • Pokec • Reddit • D&D • MN40 • Cycle • Colors • Enzymes	• GCN • Graph-SAGE • DiffPool • EdgeConv • DGMG • GUN	• ASIC	N/A
19	Challapalle 他 [54]	• レイテンシ削減 消費電力の削減	• Reddit • Pubmed • Citeseer • Cora • Collab	• GCN	• ASIC	N/A
20	Re-GNN[55]	• レイテンシ削減 • オフチップメモリアクセス数の削減 • 消費電力の削減 • チップ面積の削減	• Cora • BZR • PPI • Reddit • IMDB • Collab	• GCN • GS-Pool • G-GCN	• ASIC	N/A
21	GN-Nerator [56]	• レイテンシ削減	• Cora • Citeseer • Pubmed	• GCN • Graph-SAGE • Graph-SagePool	• ASIC	N/A
22	PIM-GCN[57]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Reddit	• GCN • Graph-SAGE • GIN	• ASIC	N/A
23	H-GCN[58]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Flickr • Reddit • Yelp • Amazon	• GIN	• ASIC	N/A
24	LW-GCN[59]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed	• GCN • Graph-SAGE	• FPGA	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 16bit 固定小数点 • 隣接行列の非ゼロ要素: 4bit 固定小数点 • 中間表現:32bit 固定小数点

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
25	Versa-GNN[60]	• レイテンシ削減 • スループットの向上 • 消費電力の削減	• Cora • Citeseer • Pubmed • IMDB-BIN • Reddit • Amazon • Collab	• GCN • Graph-SAGE • GIN • GAT	• ASIC	• 特徴ベクトル: 16bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
26	DyGNN [61]	• レイテンシ削減 • 推論精度の向上	• Cora • Citeseer • Pubmed • Reddit	• GCN • GAT	• ASIC	N/A
27	FPGAN [62]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed	• GAT	• FPGA	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 4bit 固定小数点
28	Lin-他 [63]	• レイテンシ削減 • 消費電力の削減	• Amazon • Reddit • Yelp	• GCN	• FPGA	N/A
29	Kwon-他 [64]	• レイテンシ削減 • 消費電力の削減	• Chmleon • Citeseer • Coraml • Dblpfull • CS • Corafull • Physics • Road-tx • Road-pa • Youtube • Road-ca • Wikitalk • ljournal	• GCN • Graph-SAGE	• FPGA	N/A
30	COIN[13]	• レイテンシ削減 • 消費電力の削減 • チップ面積の削減	• Cora • Citeseer • Pubmed • Nell • Extended Cora	• GCN	• ASIC	• 特徴ベクトル: 4bit 固定小数点 • 重みベクトル: 4bit 固定小数点
31	GROW [65]	• 消費電力の削減 • メモリ使用量の削減	• Cora • Citeseer • Pubmed • Flickr • Reddit • Yelp • Pokec • Amazon	• GCN	• ASIC	N/A
32	Ding 他 [66]	• 消費電力の削減 • スループットの向上	• Kinetics • NTU-RGB+D	• GCN	• FPGA	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 16bit 固定小数点
33	GNNIE [67]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Reddit • PPI	• GCN • GAT • Graph-SAGE • GINConv • DiffPool	• ASIC	N/A

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
34	ACE-GCN[68]	• レイテンシ削減	• Cora • Citeseer • PubMed • NELL • Reddit • Cont-201 • Fome20 • D6-6	• GCN	• FPGA	N/A
35	CoDG-ReRAM [69]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed	• GCN	• ASIC	• 特徴ベクトル: N/A • 重みベクトル: 2bit/4bit/8bit
36	Flow-GNN[70]	• レイテンシ削減 • 消費電力の削減	• Cora • CiteSeer • PubMed • Reddit • MolHIV • MolPCBA • HEP	• GCN • GIN • GIN + VN • GAT • PNA • DGN	• FPGA	N/A
37	FP-GNN[71]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Reddit	• GCN • Graph-SAGE • GAT	• FPGA	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
38	Sun 他 [72]	• レイテンシ削減 • 消費電力の削減 • オフチップメモリアクセスの削減	• Reddit • OGB • LiveJournal	• GCN • GIN • Graph-SAGE	• FPGA	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
39	Zhang 他 [73]	• レイテンシ削減	• Flickr • Reddit • OGBN-arxiv	• GCN • Graph-SAGE • GIN • GAT	• CPU • GPU	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
40	ReGNN [74]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Reddit	• GCN • GS-Pool	• ASIC	N/A
41	Li 他 [75]	• レイテンシ削減	• Citeseer • Collab • Pubmed	• GCN • GIN • Graph-SAGE	• ASIC	N/A
42	ReaDy [76]	• レイテンシ削減	• HepTH • Imdb • Epinions • Flickr • Youtube	• CD-GCN • TGCN • MPNN-LSTM	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
43	PAS-GCN[77]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Reddit	• GCN	• ASIC	N/A

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
44	Flow-GNN[78]	• レイテンシ削減 • 消費電力の削減	• MolHIV • MolPCBA • HEP (High Energy Physics) • Cora • CiteSeer • PubMed • Reddit	• GIN • GIN+VN • GCN • GAT • PNA • DGN	• FPGA	N/A
45	Fused-GCN[79]	• レイテンシ削減 • 消費電力の削減 • チップ面積の削減	• Cora • Citeseer • Pubmed • Reddit • Nell	• GCN	• ASIC	N/A
46	Jin 他 [80]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Nell • Reddit	• GCN	• GPU • ASIC	N/A
47	ZIPPER[81]	• レイテンシ削減 • 消費電力の削減	• ak2010 • coAuthorsD-BLP • hollywood-2009 • cit-Patents • soc-LiveJournal1 • europe-osm • Cora • Citeseer • Pubmed • Reddit	• GAT • Graph-SAGE • VGG • ResNET • PageRank	• ASIC	N/A
48	EGCN[82]	• レイテンシ削減 • オフチップメモリアクセス数の削減 • 消費電力の削減	• Cora • CiteSeer • PubMed • Nell • Flickr • Reddit • Reddit2 • Yelp	• GCN	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
49	Yoo 他 [83]	• レイテンシ削減	• Wiki • Pokec • Youtube • LiveJournal	• GCN	• ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
50	GPG-CN[84]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Nell	• GCN • GAT	• ASIC	• 特徴ベクトル: 32bit 単精度浮動小数点 • 重みベクトル: 32bit 単精度浮動小数点

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
51	SGCN[85]	• レイテンシ削減	• Cora • CiteSeer • PubMed • NELL • Reddit • Flickr • Yelp • DBLP • GitHub	• GCN • GINConv • Graph-SAGE	• ASIC	N/A
52	Zhu 他 [86]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Arxiv • Reddit	• GCN	• ASIC	• 特徴ベクトル: 12 or 8 bit 固定小数点 • 重みベクトル: 12 or 8 bit 固定小数点
53	QE-GCN[87]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed	• GCN	• FPGA	• 特徴ベクトル: 8 or 16 or 32 bit 固定小数点 • 重みベクトル: 8 or 16 or 32 bit 固定小数点
54	Graph-AGILE [88]	• レイテンシ削減	• Cora • Citeseer • Pubmed • PPI • Reddit • Yelp • Amazon-Products	• GCN • GAT • GIN • Graph-SAGE • SGC	• FPGA	N/A
55	Slice-and-Forge[89]	• レイテンシ削減	• Products • Citation • Pokec • YouTube • LiveJournal • Orkut • Reddit	• GCN	• GPU • ASIC	• 特徴ベクトル: 32bit 固定小数点 • 重みベクトル: 32bit 固定小数点
56	Stream-GCN[90]	• レイテンシ削減	• AIDS	• GCN	• FPGA	N/A
57	Cheng 他 [91]	• レイテンシ削減	• Cora • Citeseer • Pubmed	• GCN	• FPGA	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
58	NT-GAT[92]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • OGBN-arxiv • Reddit	• GCN	• FPGA	N/A
59	GCIM [93]	• レイテンシ削減 • メモリアクセス時間の削減	• Cora • Citeseer • Pubmed • DBLP • Reddit	• GCN • Graph-SAGE • GIN	• ASIC	N/A

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
60	MEGA [94]	• レイテンシ削減 • 消費電力の削減	• Cora • CiteSeer • PubMed • NELL • Reddit	• GCN • GIN • Graph-SAGE	• ASIC	N/A
61	Shen 他 [95]	• レイテンシ削減 • 消費電力の削減	• MovieLens-100K • Gowalla • Yelp • MovieLens-10M	• GCMC • NGCF • LightGCN	• ASIC	N/A
62	Merge Path-SpMM [96]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Oregon-1 • As-caida • Wiki-Vote • Nell • PPI • Amazon • coAuthorsD-BLP • soc-BlogCatalog • PROTEINS • Twitter-partial • DD • Yeast • OVCAR-8H • SW-620H	• GCN	• CPU • GPU	N/A
63	Ran 他 [97]	• レイテンシ削減	• PubMed • Flickr • Reddit • Yelp	• GCN	• FPGA	• 特徴ベクトル: 8 or 12 bit 固定小数点 • 重みベクトル: 8 or 12 bit 固定小数点
64	CoGNN [98]	• レイテンシ削減 • 演算ユニットの利用率の改善 • 消費電力の削減	• Reddit • OGBN-Products • OGBN-Papers • Yelp	• GCN • Graph-SAGE • GIN	• ASIC	• 特徴ベクトル: 8 or 12 bit 固定小数点 • 重みベクトル: 8 or 12 bit 固定小数点
65	FTW-GAT [99]	• レイテンシ削減 • 消費電力の削減	• Cora • Citeseer • Pubmed	• GAT	• FPGA	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 2bit 固定小数点
66	Nair 他 [100]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Reddit	• GCN	• FPGA	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 8bit 固定小数点
67	Accel-GCN[101]	• オフチップメモリ帯域幅使用量の削減	• am • Arxiv • com-amazon • Pubmed • SW-620H • Yelp	• GCN • GIN • Graph-SAGE	• GPU	N/A

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
68	SEIGN [102]	• 推論精度の向上	• UCI • Bitcoin • ogbl-collab • Company	• GCN • GAT • SEIGN	• CPU • GPU	N/A
69	IMGA [103]	• レイテンシ削減	• Cora • Pubmed • Citeseer • OGBN-arxiv • Ogbl-collab • Ogbl-ppa • Reddit	• GCN	• ASIC	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 8bit 固定小数点
70	TGOpt [104]	• レイテンシ削減	• jodie-lastfm • jodie-mooc • jodie-reddit • jodie-wiki • snap-email • snap-msg • snap-reddit	• TGAT	• CPU • GPU	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
71	GHOST [105]	• 推論の改善 • 消費電力の削減	• Cora • PubMed • Citeseer • Amazon • Proteins • Mutag • BZR • IMDB-binary	• GCN • GAT • Graph-SAGE • GIN	• ASIC	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 8bit 固定小数点
72	NEM-GNN [106]	• レイテンシ削減 • スループットの向上 • 消費電力の削減	• Cora • Citeseer • Pubmed • Reddit • Twitter	• GCN • GAT • Graph-SAGE	• ASIC	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 8bit 固定小数点
73	Chen 他 [107]	• レイテンシ削減	• Cora • Citeseer • Pubmed • Flickr • NELL • Reddit	• GCN • Graph-SAGE • GIN • SGC	• FPGA	N/A
74	Dyna-sparse [108]	• レイテンシ削減	• Cora • CiteSeer • PubMed • Flickr • NELL • Reddit	• GCN • Graph-SAGE • GIN • SGC	• FPGA	N/A
75	GNN-Builder [109]	• レイテンシ削減	• QM9 • ESOL • FreeSolv • Lipophilicity • HIV	• GCN • Graph-SAGE • GIN • PNA	• FPGA	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 32bit 固定小数点

表 5. トレーニングのみを対象とする GNN ／ GCN ハードウェア実装の既存研究の分類

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
76	Graph-ACT[110]	• レイテンシ削減 • DSP 利用率向上 • オンチップメモリアクセスの削減 • オフチップメモリアクセスの削減	• PPI • Reddit • Yelp	• Graph-SAGE • Para-GCN • GraphACT	• CPU • FPGA	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
77	Pipe-GCN[11]	• レイテンシの削減 • スループットの向上 • GPU 間の通信オーバーヘッドの削減	• Reddit • OGBN-Products • Yelp • OGBN-Papers 100M	• GCN	• GPU	N/A
78	Rubik[111]	• レイテンシの削減 • オフチップメモリアクセス数の削減 • 消費電力削減 • チップ面積削減	• Collab • BZR • IMDB • DD • Citeseer • Reddit	• GIN • Graph-SAGE	• ASIC	N/A
79	HP-GNN[112]	• スループットの改善	• Flicker • Reddit • Yelp • Amazon-Products	• GCN • Graph-SAGE	• CPU • FPGA	N/A
80	GC-Near[113]	• レイテンシの削減	• Reddit • Nell • Amazon • OGBN-Papers	• GCN • GIN • SAGEConv • GAT	• ASIC	N/A
81	GNNear [33]	• レイテンシの削減 • 消費電力の削減	• OGBN-Proteins • Reddit • Yelp • Amazon	• GCN • GIN • SAGEConv • GAT	• ASIC	N/A
82	Skeleton - GCN[114]	• レイテンシの削減	• PPI • Reddit • Yelp	• GCN • Graph-SAINT	• FPGA	• 特徴ベクトル: 16bit 符号付き整数 • 重みベクトル: 16bit 符号付き整数
83	Hu-Graph[115]	• レイテンシの削減	• Reddit • Yelp	• GCN	• FPGA	• 特徴ベクトル: 8bit 固定小数点 • 重みベクトル: 8bit 固定小数点
84	Ogbogu 他 [116]	• レイテンシの削減	• PPI • Reddit • Amazon • Flickr • Yelp • ogbl-citation2	• GCN • GAT • Graph-SAGE	• ASIC	• 特徴ベクトル: 16bit 浮動小数点 • 重みベクトル: 16bit 浮動小数点

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
85	MaxK-GNN[117]	• レイテンシの削減	• Reddit • OGBN-proteins • OGBN-products • Yelp • Flickr	• GCN • Graph-SAGE • GIN	• GPU	N/A
86	Hansson 他 [118]	• レイテンシの削減	• Cora • CiteSeer • PubMed	• GCN	• CPU • FPGA	• 特徴ベクトル: 4 or 8 or 16 or 32 bit 固定小数点 • 重みベクトル: 4 or 8 or 16 or 32 bit 固定小数点
87	FASTEN [119]	• レイテンシの削減	• AIFB • MUTAG • BGS • AM • Freebase • DBLP • ACM • IMDB	• RGCN • RGAT • HGT	• GPU	• 特徴ベクトル: 32bit 浮動小数点 • 重みベクトル: 32bit 浮動小数点
88	FARe [120]	• 欠陥のある ReRAM ハードウェア上での精度の改善	• PPI • Reddit • OGB-citation2 • Amazon	• GCN • GAT • Graph-SAGE	• ASIC	• 特徴ベクトル: 16bit 固定小数点 • 重みベクトル: 16bit 固定小数点
89	HitGNN [121]	• レイテンシの削減 • メモリ帯域幅効率の改善 • 消費電力の削減	• Reddit • Yelp • Amazon • OGBN-Products	• GCN • Graph-SAGE	• CPU • FPGA	N/A
90	RSC [122]	• レイテンシの削減	• Reddit • Yelp • OGBN-Proteins • OGBN-Products	• GCN • Graph-SAGE • GCNII • Graph-SAINT	• GPU	N/A
91	Marius GNN[123]	• レイテンシの削減	• Papers-100M • Mag240M-Cites • Freebase-86M • Wiki-KG90Mv2 • Hyperlink 2012 • Facebook15	• Graph-SAGE • GAT	• CPU • GPU	N/A
92	HyScale -GNN [124]	• レイテンシの削減	• OGBN-Products • OGBN-Papers100M • MAG240M	• GCN • Graph-SAGE	• CPU • GPU • FPGA	N/A

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
93	MGG [125]	• レイテンシの削減	• Reddit • Enwiki-2013 • it-2004 • OGBN-Paper100M • OGBN-Products • OGBN-Proteins • com-orkut	• GCN • GIN • Graph-SAGE • GAT	• GPU	N/A
94	FlashGNN [126]	• レイテンシの削減	• OGBN-Products • OGBN-Papers100M • soc-twitter-mpi-sws • soc-friendster	• Graph-SAGE	• ASIC	N/A
95	TT-GNN [127]	• レイテンシの削減 • 消費電力の削減	• Cora • Reddit • OGBN-Arxiv • OGBN-Products • OGBN-Papers100M	• Graph-SAGE	• ASIC	• 特徴ベクトル: 16bit 浮動小数点 • 重みベクトル: 16bit 浮動小数点

表 6. 推論／トレーニング両方を対象とする GNN ／ GCN ハードウェア実装の既存研究の分類

ID	論文	目的	データセット	ネットワーク	デバイス	量子化精度
96	Bi-GCN[128]	• 推論レイテンシの削減 • レイテンシの削減 • メモリ使用量の削減	• Cora • PubMed • Reddit • Flickr	• GCN	• ASIC	1bit
97	SGC-NAX[129]	• 推論レイテンシの削減 • 消費電力の削減 • オフチップメモリアクセス数の削減	• Cora • Citeseer • Pubmed • Nell • Reddit	• GCN	• ASIC	N/A
98	RelHD [130]	• 推論レイテンシの削減 • レイテンシの削減 • 消費電力の削減	• Cora • Citeseer • Pubmed • Reddit	• GCN	• ASIC	N/A
99	Liao 他 [131]	• 推論レイテンシの削減 • レイテンシの削減	• Reddit • PPI • Yelp • Amazon • MAG • Papers-100M	• GCN	• CPU • GPU	N/A
100	SaGNN [132]	• 推論レイテンシの削減 • レイテンシの削減	• Cora • Citeseer • BZR • PPI • Pubmed • Reddit	• GCN • Graph-SAGE • G-GCN	• ASIC	N/A

第 4 章　最適化手法

本章では, 調査した研究論文の中で特徴的な最適化手法を提案している手法（HyGCN[9], G-Cos[49]）を紹介する.

4.1　HyGCN

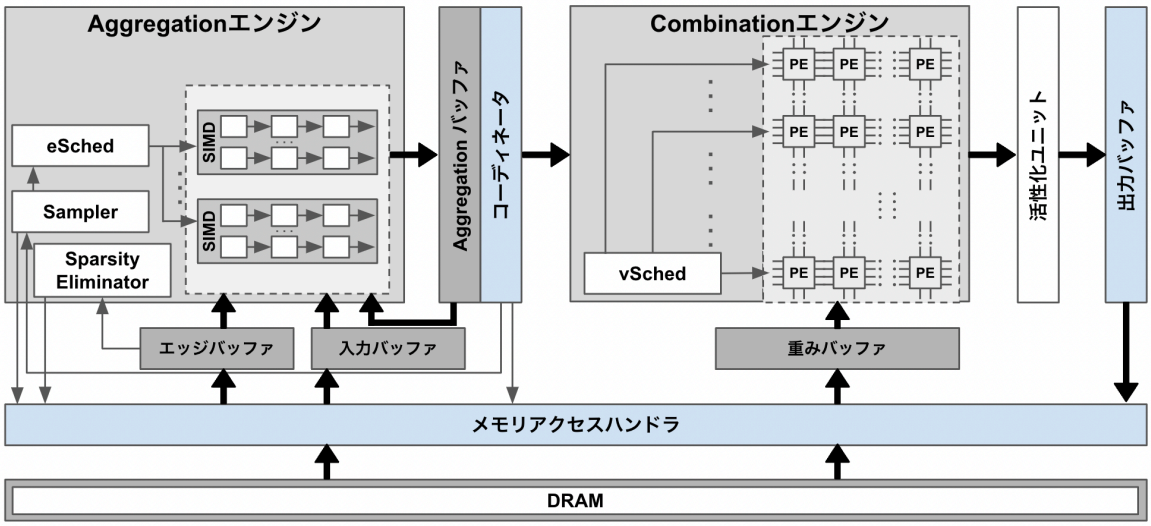

図 15: HyGCN のアーキテクチャの概要 [9]

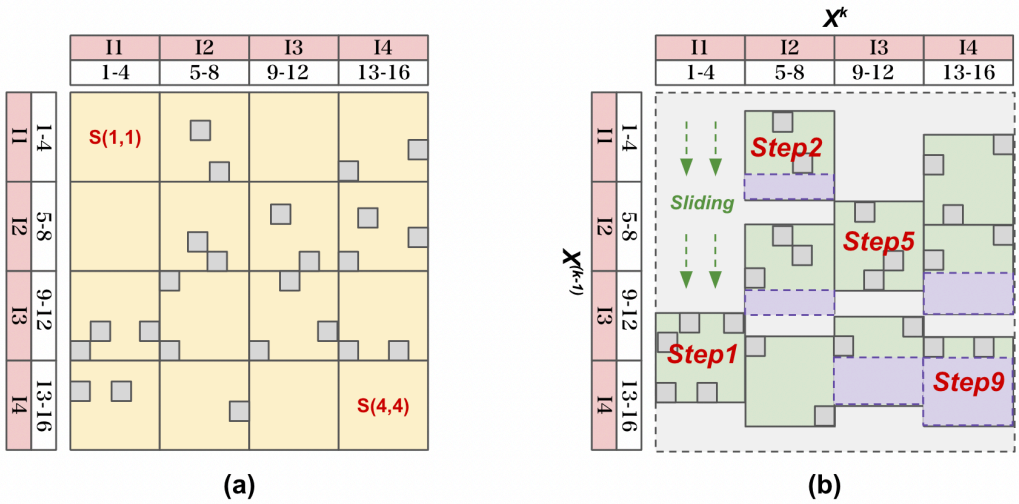

図 16: (a) データの局所性向上のための静的なグラフ分割 (b) スパース性除去のための Sliding Window アプローチ

HyGCN のアーキテクチャの概要を図 15 に示す. HyGCN は, GCN のメモリ集約的な「集約」処理を実行する Aggregation エンジンと, 計算集約的な「結合」処理を実行する Combination エンジンによって, 異なる計算特性の処理に対応するハイブリッドなアーキテクチャを採用している.

Aggregation エンジンの全体の処理フローは, Sampler により各頂点のエッジリストからエッジを選択して, eSched（タスクスケジューラ）により選択されたエッジ処理のタスクを SIMD コアに割り当て, 各 SIMD コアは割り当てられたエッジのスパース性の除去と集約処理を独立して実行して, 各 SIMD コアで生成された中間集約結果は集約バッファに集められ, 最終的な集約結果が生成される. スパース性の除去の前に行われる静的なグラフ分割を図 16(a) に示す. グラフのノードは, $I1$ から $I4$ の区間のそれぞれに 4 つずつ計 16 個存在する. グラフのエッジは, 4×4 のシャード, すなわち $S(1,1)$ から $S(4,4)$ の区間のそれぞれで最大 16 個存在する. 灰色の箇所は, 頂点間にエッジが存在することを意味する. 静的なグラフ分割によって隣接行列のように関係するエッジを近くに配置することで, データの局所性が向上して, 集約処理の実行時のメモリアクセスの効率が向上する. 静的なグラフ分割の後に行われるスパース性除去の処理フローを図 16(b) に示す。各頂点について, シャードウィンドウをノード区間毎に設けて, 最上段からスタートしてエッジが現れるまで 1 行ずつ下にスライドさせていく. エッジが現れた箇所を上方シャードウィンドウとする. 次に, 最下段からスタートして同様にエッジが現れるまで 1 行ずつ上にスライドさせていき, エッジが現れた箇所を下方シャードウィンドウとする. このような方法でスパース性を除去した後に, 集約処理が開始される.

Combination エンジンは, Aggregation エンジンにより実行された集約結果を取得して重みバッファから重み行列をロードした後, シストリックアレイという複数の演算ユニット (Processing Elements, PEs) が規則的に配列された構造によって結合処理を実行する. 各 PE は隣接する PE からデータを受け取り, 演算を行った結果を次の PE に送信する. この構造により, パイプラインのようにデータが連続して処理され, 高い並列度で結合処理が行われる. シストリックアレイで処理を行うにあたって, 各 PE が他の PE の処理経過を考慮せずに独立して実行する独立作業モードと, 他の PE の処理経過を考慮して実行する協調作業モードがある. 独立作業モードでは, 他の PE の作業完了を待たずに結合処理を行なう形であり, 小規模なデータセットを扱う場合はこのモードが適している. しかし, 大規模な行列を処理することになった際にメモリアクセスのコストが非常に増大してエネルギー効率

が悪化するという課題がある．これに対して協調作業モードでは，重み行列や特徴ベクトルのデータが一度ロードされるとこれを複数の PE が共有して使用することでメモリアクセスの回数を減らすことができるため，大規模なデータセットを取り扱う際は協調作業モードが適している．

4.2 G-CoS

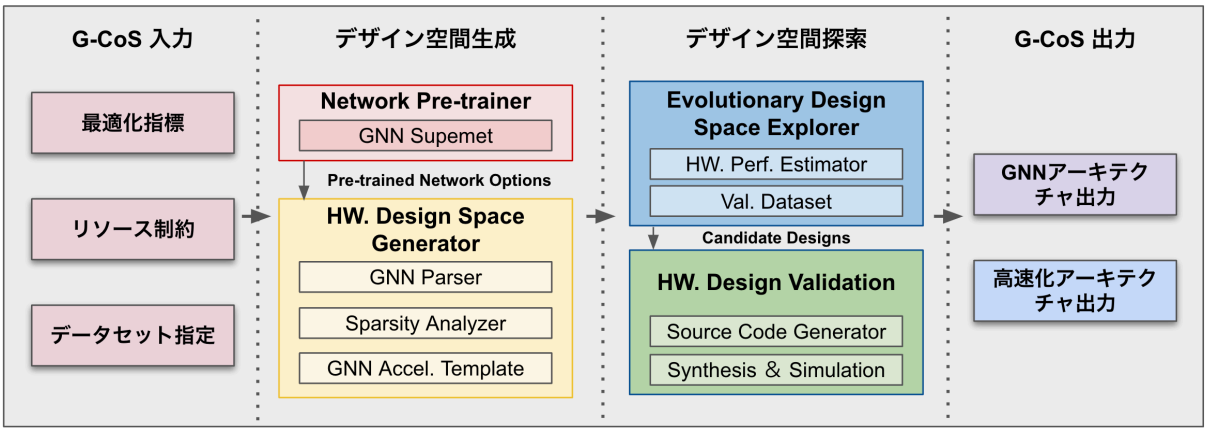

図 17: G-CoS フレームワークの概要 [49]

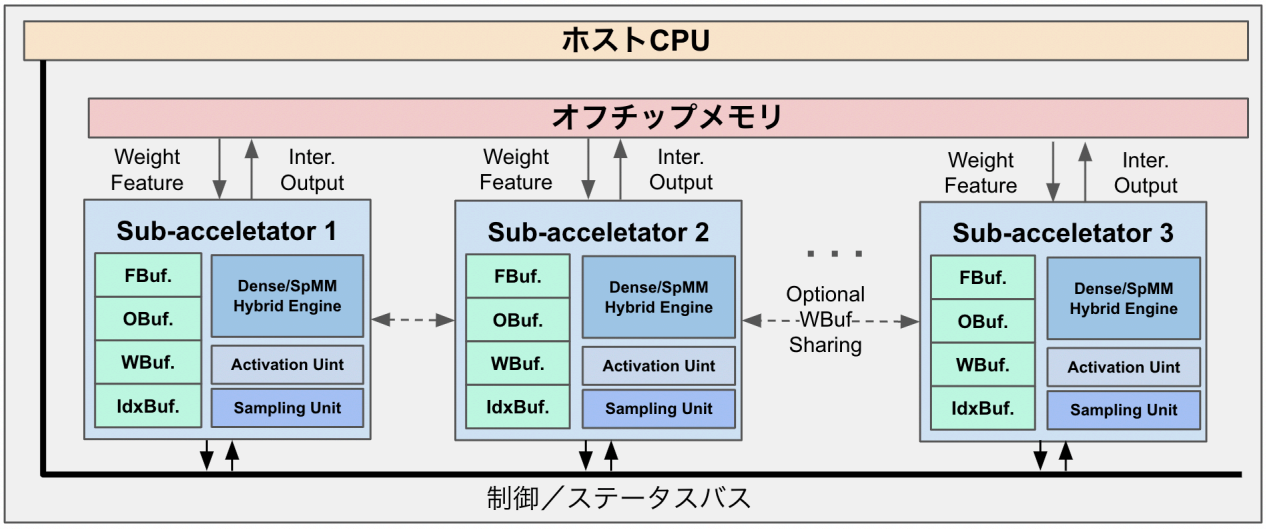

図 18: G-CoS マイクロアーキテクチャ・テンプレートの概要 [49]. 入力として与えた制約条件と，GNN のアーキテクチャに応じてサブアクセラレータの数やバッファの容量が異なる．

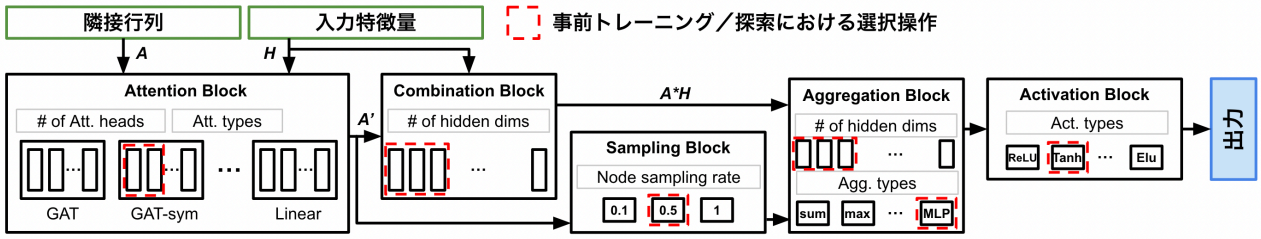

図 19: G-CoS の GNN スーパーネットのセグメント [49]

G-CoS は, 指定した制約条件に基づいて最適な GNN 構造とそれに適合する FPGA アクセラレータの構造を出力する手法である. 制約条件の入力から最終出力までの処理フローを図 17 に示す. G-CoS 入力で, 最適化指標（例えば, 精度, レイテンシ, 消費電力）, リソースの制約（使用可能なメモリリソースの容量等）, 指定されたデータセット（Cora, CiteSeer. Pubmed 等）を入力として与えると, G-CoS 出力では, 最適なネットワーク構造（層の数, 各層のニューロン数, 活性化関数, 正規化手法等）と, この構造に適した FPGA アクセラレータの構成情報（図 18）が出力される.

G-CoS の協調最適化プロセスは以下のように定式化される.

$$
\begin{aligned}
&\arg \min_{\{gnet, hw\}} L_{\text{val}}(\omega^*, gnet, hw) + \lambda L_{\text{cost}}(gnet, hw) \\
&\text{s.t.} \quad \omega^* = \arg \min_{\omega} L_{\text{train}}(\omega, gnet),
\end{aligned}
$$

最適化の目的は以下の 2 つの項目の合計を最小化することである. ここで, ω, L_{train}, L_{val}, L_{cost}, λ はそれぞれ, GNN の重み, 訓練中のタスク損失, 検証中のタスク損失, ハードウェアのコスト, コストの重みを調整するための係数である.

- $L_{\text{val}}(\omega^*, gnet, hw)$: GNN の構造 $gnet$ とアクセラレータ hw に対する検証損失.
- $\lambda L_{\text{cost}}(gnet, hw)$: GNN の構造 $gnet$ とアクセラレータ hw に対するハードウェアコスト. このコストは, 指定されたリソース制約や最適化指標に基づいて計算される.

最適化の流れとしては, 最初に GNN の訓練を行い, 訓練損失 $L_{\text{train}}(\omega, gnet)$ を最小化する重み ω^* を求める. 次に、その重み ω^* を用いて, 検証損失 $L_{\text{val}}(\omega^*, gnet, hw)$ とハードウェアコスト $\lambda L_{\text{cost}}(gnet, hw)$ の合計を最小化するように、GNN 構造 $gnet$ とアクセラレータ hw を共同で最適化する. この共同の最適化プロセスにより,

G-CoS はタスクの精度とハードウェア効率の両方を最大化する最適な GNN 構造とアクセラレータ構造の組み合わせを見つけることができる. 組み合わせを探索するにあたって, 膨大な数のニューラルネットワークアーキテクチャを探索してモデルを都度トレーニングすると多くの計算資源と時間がかかる. より効率的に探索するために, G-CoS はワンショット NAS（One-Shot Neural Architecture Search）メソッドを採用している. ワンショット NAS では, 初めに全ての候補アーキテクチャを包含する大規模なネットワーク（スーパーネット）を初めに 1 度だけトレーニングする（図 19）. 各サブネットはスーパーネットの一部であり, スーパーネットの訓練済みパラメータを使用しているため, サブネットの性能評価はスーパーネットの訓練後に迅速に行うことができ, 全てのアーキテクチャを個別に訓練する必要がなくなる.

第 5 章　タスクスケジューリング

本章では, 調査した研究論文の中で特徴的なタスクスケジューリング手法を提案している手法 (AWB-GCN[10], EnGN[39]) を紹介する.

5.1　AWB-GCN

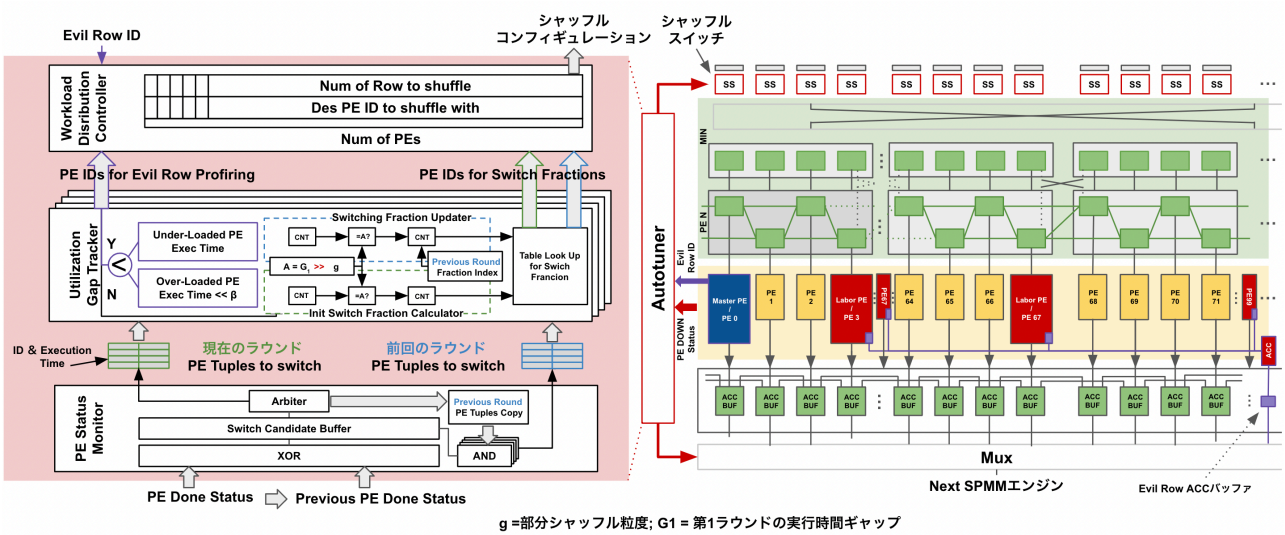

図 20: AWB-GCN アーキテクチャの概要 [10]

AWB-GCN のアーキテクチャの概要を図 20 に示す. AWB-GCN は, GCN の推論を FPGA で高速化するにあたって, PE 間の作業負荷の不均衡を解消するために 3 種類の自動チューニングを複数回のラウンドにわたって行う. 行列乗算のタスクを各 PE に割り当てることにより並列実行を行うが, 各 PE の負荷状況に応じて割り当て先を動的に切り替えているという意味で, 動的なスケジューリングを採用している. 1 ラウンドにつき計算フェーズと負荷分散フェーズがある. 行列乗算は, (特徴行列X × 重み行列W) × 隣接行列A の順番で行い, ラウンド数は出力行列の次元数と一致する. つまり, 1 ラウンドで出力行列の 1 列の結果が算出されることになる（図 21）. 負荷分散のために, 計算フェーズで Distribution smoothing を行い, 負荷分散フェーズで Remote switching, Evil row remapping を実行する (図 22).

計算フェーズでは, Distribution smoothing で, Task Distributor & Queue (TDQ) が新しいタスクを受け取る際, 自身の PE のタスク数と近隣の PE のタスク数を比較する. もし自身のタスク数が多い場合、最もタスク数の少ない近隣の PE にタスク

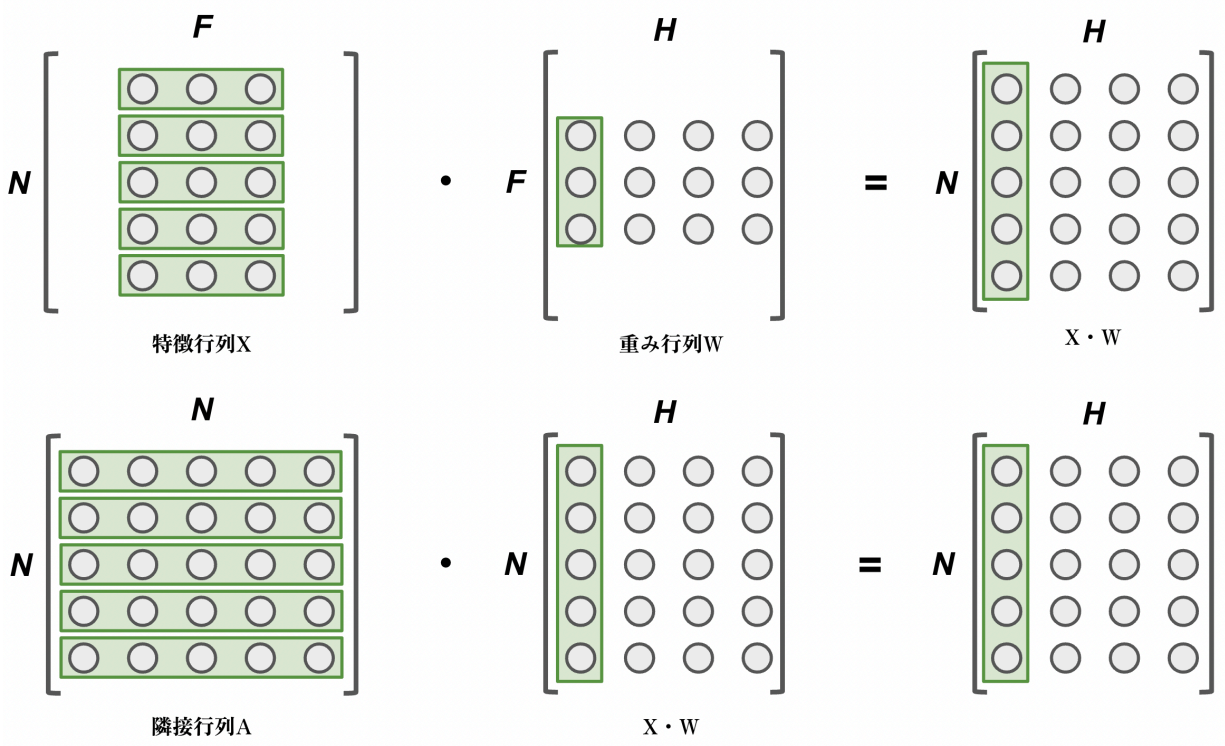

図 21: 1 ラウンドで $N \times H$ の出力行列の 1 列の結果が算出されることを表す図. N はグラフのノード数, F は入力特徴量の次元数, H は出力特徴量の次元数を表す.

を割り当てた後に行列乗算を行い, 自身のタスク数が少ない場合はそのまま行列乗算を行う. 全ての PE の行列乗算処理が完了後, 負荷分散フェーズに移る. 負荷分散フェーズの Remote switching は, Distribution smoothing では解決できない大規模な作業負荷の不均衡を検出した場合のみ実行される. PE Status Monitor が, 最も負荷の高い PE と最も負荷の低い PE を特定して, 次のラウンドではその 2 つの PE 間でタスクを部分的あるいは完全に交換する. 交換の割合は, 前のラウンドでの PE の利用率に基づいてオートチューナーによって動的に決定される. Evil row remapping は, Distribution smoothing と Remote switching では対処できないほど極端に負荷が高い PE(evil row) が存在する場合に起動される. Utilization Gap Tracker が最も負荷の高い PE と最も負荷の低い PE の負荷の差を計算して, その差が閾値を超えている場合は, Workload Distribution Controller が Super-PE にタスクを割り当てて, 複数の Labor PE に分割される. Evil row remapping は, 一般的には処理の初期段階で一度だけ行われるが, 新たな evil row が発見された場合は追加で実行が行われることもある. これらの技術により, AWB-GCN はデータの特性に基づいて動的にワークロードを調整して, 全体としての処理効率と性能を最大化することがで

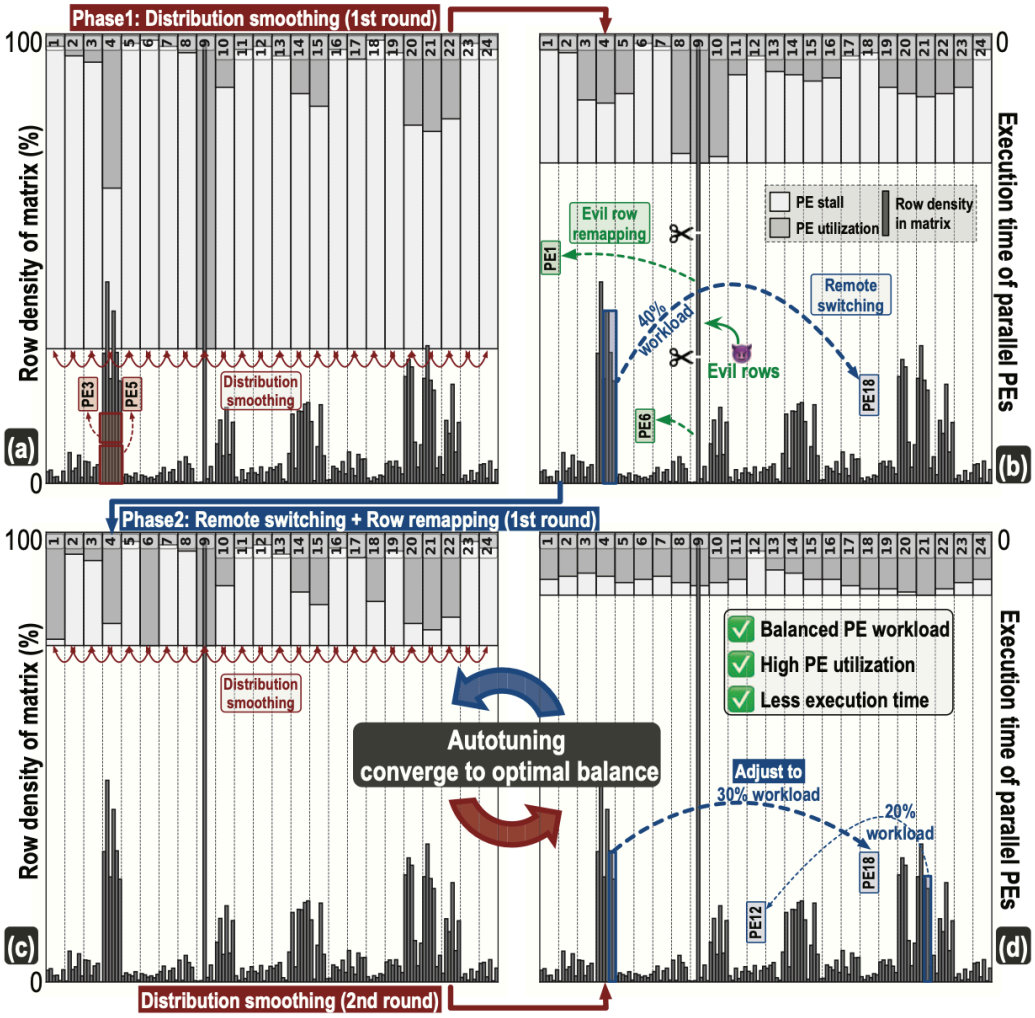

図 22: 負荷分散の流れ（[10] から転載. ©2020 IEEE.）. 計算フェーズで Distribution smoothing を実行後, 負荷分散フェーズで Remote switching と Row remapping を実行する.

きる.

5.2 EnGN

EnGN のアーキテクチャの概要を図 23 に示す. 処理の流れについて, まずメモリコントローラで外部メモリから入力データ（隣接行列, 特徴ベクトル, 重みベクトル）をそれぞれエッジバンク, 特徴バンク, 重みバンクに格納する. 次に, NGPU 内にある各 PE で特徴行列と重み行列の乗算処理（特徴抽出）を行い, その結果と隣接行列の乗算処理（集約）を行う. 集約実行時は, Ring-Edge-Reduce というリング構造の仕組みを使うことで, 外部メモリを使わずに自身の PE の特徴抽出結果を

隣接する PE に連携することができる. 最後に, PE 内にある XPE がこれらの演算結果に活性化関数を適用して, 結果バンクに格納する. ベクトル処理ユニットは, GCN 以外の GNN モデルの処理を行う際に使用される. EnGN の特徴的な手法は, Ring-Edge-Reduce とエッジの並び替えであるが, AWB-GCN とは異なり, 他の PE の負荷状況を考慮したタスク割り当て先の切り替えを行なっていないという意味で, 静的なスケジューリングを採用している.

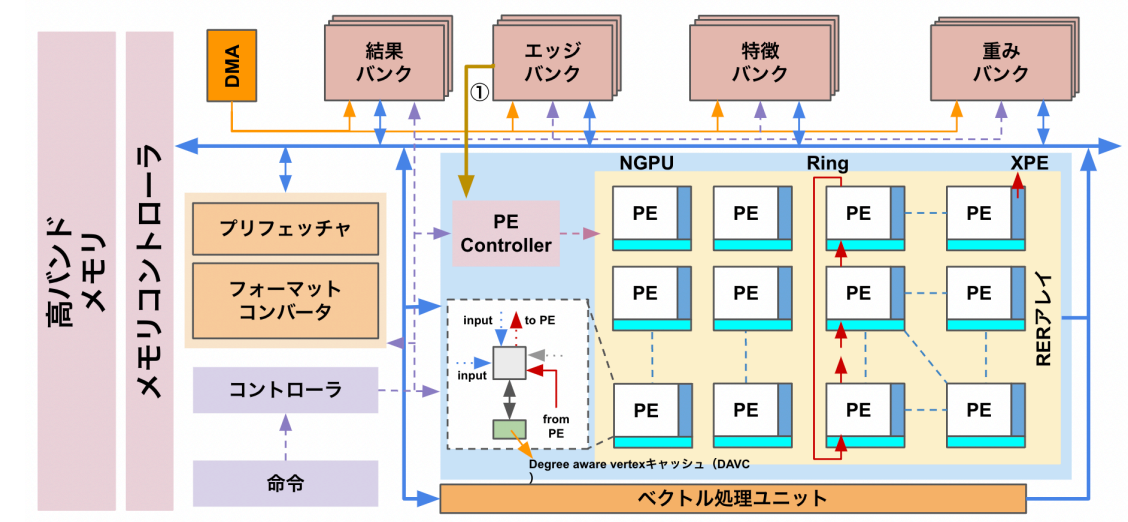

図 23: EnGN のアーキテクチャの概要 [39]

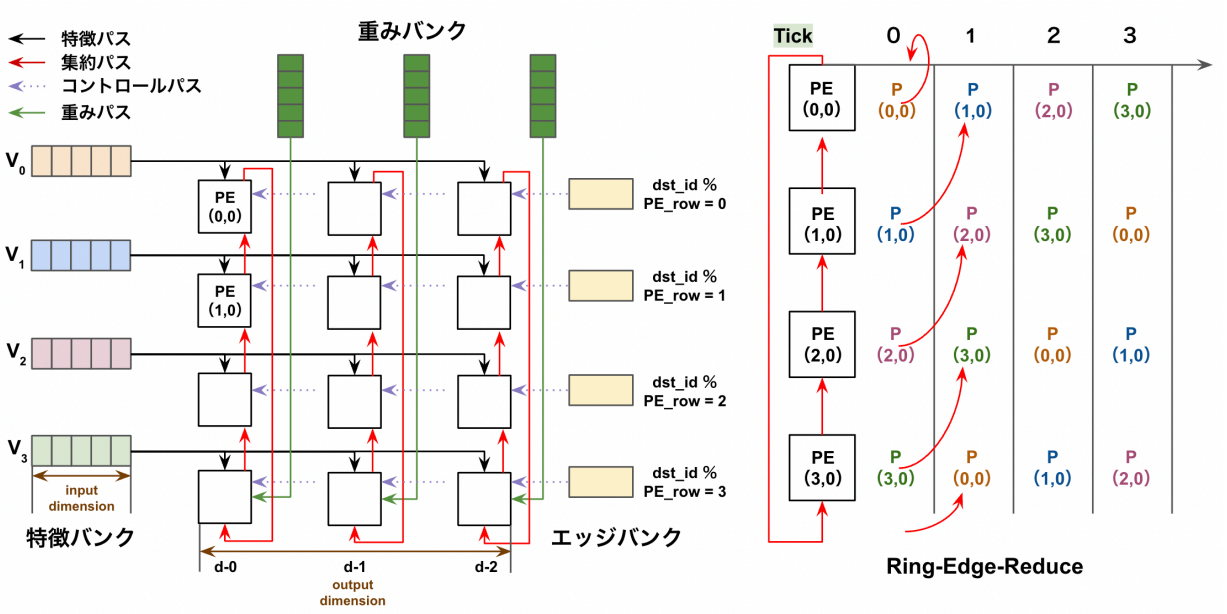

図 24: Ring-Edge-Reduce の仕組み [39]

Ring-Edge-Reduce は, 外部のメモリを介さずに特徴抽出ステージで生成した結果を他の PE に連携する仕組みである. 各 PE は, 特徴抽出ステージで生成した自身の頂点特徴量を同じ列にある全ての PE に渡す必要があるが, 外部メモリを介して実行すると外部メモリに対する I/O コストが大きくなる. これを防ぐために Ring All Reduce[133] を参考にして, データを外部メモリに移動する代わりに PE 間で直接データを連携する手法を提案している. 図 24 の通り, 各 PE は隣接する PE と接続していて, 自身の持つ特徴抽出結果を 1 Tick 毎に 1 つだけ隣接する PE に渡している. これにより, 消費電力を抑えつつ隣接 PE にデータを渡すことができる.

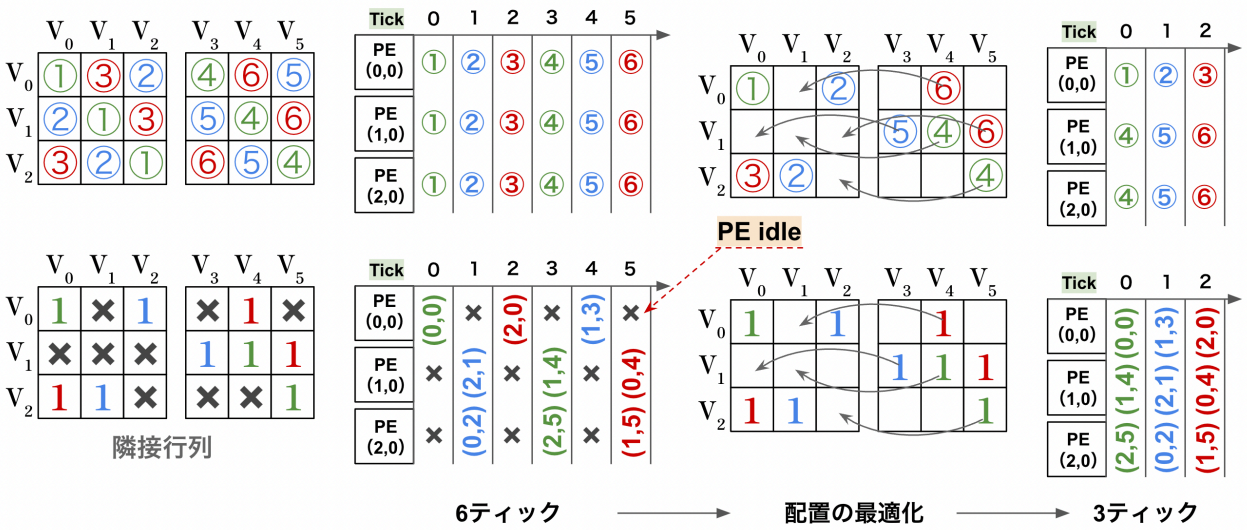

図 25: エッジの並び替え [39]

スパースな隣接行列を取り扱うにあたって, グラフの並び替えをすることで不要なゼロ乗算が発生しないようにしている. 図 25 の左下の隣接行列について V_n は頂点を表していて, 1 はエッジが存在すること, × はエッジが存在しないこと, つまり値が 0 であることを意味する. 初期状態の隣接行列には多くの × が存在していて, これにより不要なゼロ乗算が発生することで Tick を余分に消費する. そこで非ゼロ要素を左側にシフトすることによって要素の配置を最適化する. この再配置によって不要なゼロ乗算がなくなることで, 少ない Tick 数で処理を完了させることが可能になる.

第 6 章　今後の課題と展望

GNN ／ GCN のアプリケーションのハードウェア実装についてこれまで様々な手法が提案されてきたものの, 現状のハードウェア実装群ではまだ十分には解決されていない課題がいくつかある. 本章では, それらの課題について示す.

まず対象とするネットワークについて, 一部のネットワークに偏っていてより広範囲の GNN アーキテクチャに対応できていないという課題がある. 図 13(a) では, 推論のみを対象とする研究において GCN は 68 本の論文で使用されているが, GCN, GraphSAGE, GIN, GAT 以外のネットワークは全て 4 本以下である. トレーニングのみを対象とする図 13(b) についても (a) と同様に, GCN, GraphSAGE, GIN, GAT を採用している論文が多いが, それ以外の論文数が比較的少ない.

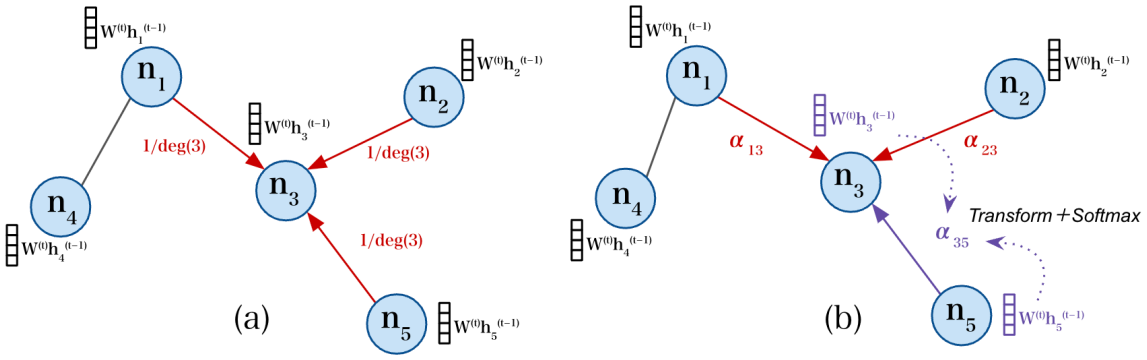

図 26: GCN における集約処理 (a) と GAT における集約処理 (b) の比較. GCN における集約処理では, 各ノードは隣接するノードから情報が重み付き和で集約される. 重みは隣接ノードの次数に基づいて決定され, 具体的には各重みは $\frac{1}{\deg(i)}$ となる. GAT ではアテンション係数 a_{ij} を使って集約される. アテンション機構により, 異なる隣接ノードに対して異なる重要度を割り当てることができる. GAT は GCN に比べて表現力と柔軟性が増す一方で, 計算の複雑さが増す.

次に, 比較的多く採用されている GAT についても, GAT の計算特性に十分に適合した手法を提案している論文は非常に少ない. 前提として, GAT は GNN の派生形であり, アテンション機構を取り入れた GNN アーキテクチャである. GAT では, 各ノードがその近傍ノードの重要度を動的に評価し, 重要なノードからの情報をより重視することで, グラフ構造データの複雑なパターンを効果的に捉えるようになっている. ノードの重要度は, アテンション係数（a_{vu}）という, あるノードがその近傍ノードの重要度を評価するために計算される値によって識別される（表 2）.

アテンション係数を計算する際に, ノード v とその近傍ノード u との間の関係性を測るために, $\sigma\left(a^T\left[W_1^l h_v^l || W_2^l h_u^l\right]\right)$ により, 重みベクトル W_1^l と特徴ベクトル h_v^l, 重みベクトル W_2^l と特徴ベクトル h_u^l がそれぞれ乗算処理を行う仕様になっている. 図 26 に GCN と GAT の集約処理の違いを示す. 一般に, 多くの GCN のハードウェア実装では, 集約フェーズを, 大規模な疎行列と密行列同士の乗算という意味で「メモリ集約な処理」, 結合フェーズを, 密行列同士の乗算という意味で「計算集約な処理」として 2 つのパートに分けてそれぞれに特化した設計を行なっている. しかし GAT の集約フェーズには, 疎行列と密行列の乗算だけでなく密行列同士の乗算処理もあるため, 「メモリ集約な処理」と「計算集約な処理」の双方が含まれている. そのため, GCN も GAT も同じグラフニューラルネットワークの一種という位置付けではあるが, 実際は計算特性が異なるので両方のネットワークに有効なハードウェア実装をするのは容易ではない. したがって, GCN と GAT の双方の計算特性を十分に考慮して同時にサポートをする論文は非常に少ないのが現状である.

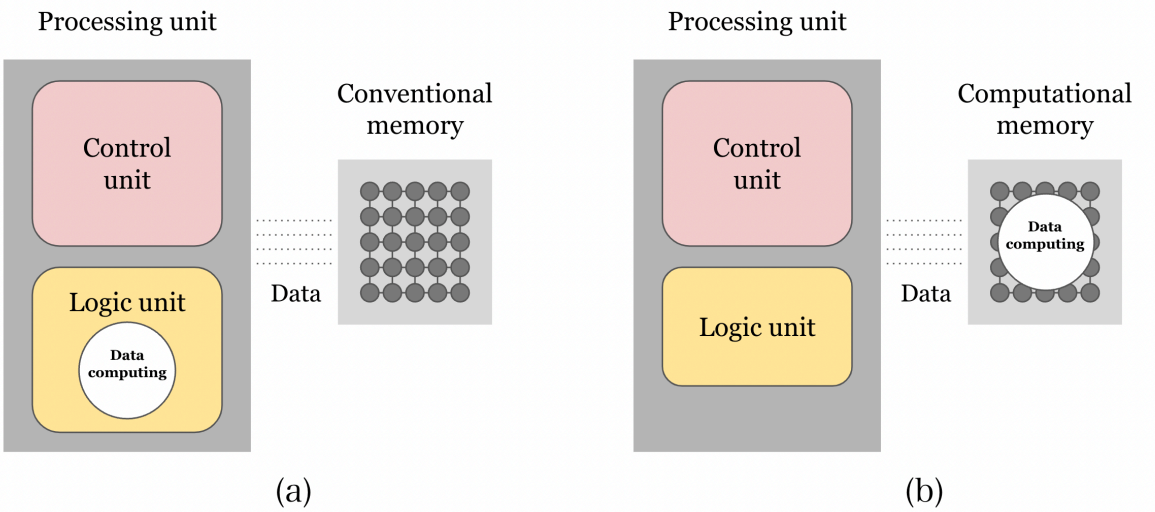

図 27: (a) ノイマン型アーキテクチャと (b) インメモリ型の比較. ノイマン型アーキテクチャでは CPU とメモリが分離していて, CPU がメモリから命令を取り出して実行して結果をメモリ等の記憶装置に格納する処理を逐次的に行う. これに対してインメモリ型では, メモリチップ内に計算回路が組み込まれていて, CPU とメモリ間のオーバーヘッドがない状態で命令を実行していくことができる.

さらに, GNN ／ GCN のハードウェア実装において大量に発生するメモリアクセスの削減のためにいくつかの論文 (ReGNN[55], COIN[13]) がインメモリアクセラレータの提案をしているが, 依然として課題が残されている. 従来のノイマン型とインメモリ型の違いを図 27 に示す. ノイマン型を前提としたデジタル演算では

信号の精度に誤差が生じない利点があるものの, メモリアクセス時のオーバーヘッドが無視できない大きさになる. インメモリ型ではそのようなオーバーヘッドはないが, アナログ回路による演算機能の実装が必要で, デジタル回路のような離散的な信号ではなく連続的な信号を取り扱うことになる. 一般に, 連続的な信号を取り扱うことになると外部環境からのノイズの影響を受けて信号の精度が劣化しうることがアナログ回路における最大の弱点であるが, ディープラーニングの文脈では信号の精度に多少揺らぎがあっても全体としての推論／トレーニングの精度に影響が出ないためインメモリ形式のアクセラレータの適用が試みられてきた. 現状の課題としては, インメモリ形式の GNN ／ GCN ハードウェア実装はいくつかあるものの, GNN ／ GCN に特有のスパース行列乗算における計算特性を十分に考慮した論文はまだ存在しない認識である. そのため, インメモリアクセラレータの手法を提案するにあたっては, そのような技術課題に対処していくことが求められる.

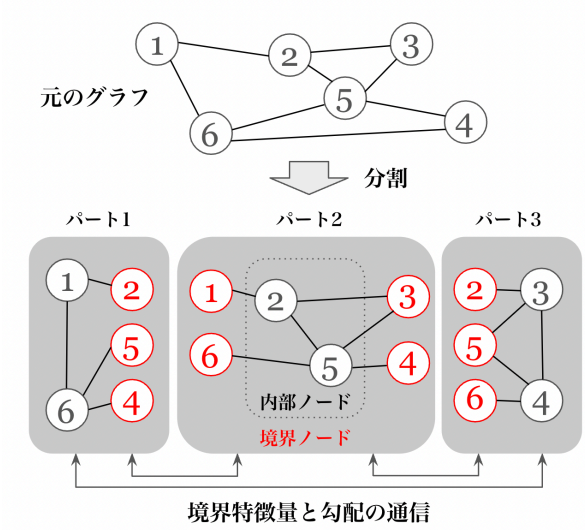

図 28: PipeGCN の分散トレーニングの仕組み [11]. 元のグラフを複数のグラフに分割して, 複数のパートをそれぞれ異なる GPU にアサインしてトレーニングを行う.

最後に, 現状のテスト用のデータセットよりさらに大規模なデータセットに対応する必要がある. これまで数多くのグラフを処理するシステムが作られてきたが, 認知度の高いデータセットとして最大 6.6 億エッジのベンチマークデータセット（Yahoo web[135]）で評価されてきた. しかし Ching 他 [135] によると, 数千億から最大 1 兆エッジのデータセットを取り扱う企業もある. このようなさらに大規模なデータセットのトレーニングのレイテンシを削減するためには, 図 28 のように, 1 つのデバイスではなく複数のデバイスにタスクを割り当てて協調通信しながら

処理を行う必要がある. しかし, GNN / GCN のハードウェア実装の研究においては, このような分散トレーニングの既存研究はほとんどないため, 分散トレーニングにおいてデバイス間の通信効率を改善する等の最適化手法をさらに探究していく必要がある.

第 7 章　おわりに

本研究では, GNN ／ GCN のアプリケーション例, 基本概念と定義, 実行プラットフォームの概要を示した上で, GNN ／ GCN のハードウェア実装の既存研究をいくつかの分類項目に基づいて整理した. 整理するにあたっては, ①推論のみを対象とする手法, ②トレーニングのみを対象とする手法, ③推論／トレーニングの両方を対象とする手法の 3 種類に分けて分類を行った. その上で, 3 種類の区分毎の論文数や, 3 種類それぞれにおけるハードウェア実装の目的や使用するデータセット, ネットワーク, デバイス, 量子化精度面における全体的な傾向についての考察を行なった. また, 特徴的な最適化手法, タスクスケジューリング手法を採用しているハードウェア実装をいくつか取り上げて, GNN ／ GCN のハードウェア実装の基本的な設計課題（非ゼロの要素のデータ分布が非常にアンバランスである点, 行列サイズが非常に大きい点）に対してどのような解決方法を提案しているかについて具体的な内容を示した. 最後に, GNN ／ GCN のハードウェア実装における今後の課題と展望について考察を行なった.

参考文献

[1] Franco Scarselli, et al., ”The Graph Neural Network Model”, Proc. IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 20, NO. 1, JANUARY 2009.

[2] Thomas N. Kipf, et al., ”Semi-Supervised Classification with Graph Convolutional Networks”, arXiv:1609.02907, 2016.

[3] P. Sen, et al., ”Collective classification in network data”, AI magazine, vol. 29, no. 3, p. 93, 2008.

[4] M. Zitnik et al., ”Predicting multicellular function through multi-layer tissue networks”, Bioinformatics, vol. 33, no. 14, pp. i190-i198, 2017.

[5] H. Zeng, et al., ”GraphSAINT: Graph sampling based inductive learning method”, in Proc. Int. Conf. Learning Representations, 2020, pp. 1-19.

[6] W. Hamilton, et al., ”Inductive representation learning on large graphs”, in Proc. of NIPS, 2017, pp. 1024-1034.

[7] K. Xu, et al., ”How powerful are graph neural networks?” in Proc. Int. Conf. Learning Representations, 2019, pp. 1-17.

[8] Weihua Hu, et al., ”Open Graph Benchmark: Datasets for Machine Learning on Graphs”, arXiv:2005.00687, 2021.

[9] M.Yan, et al., ”HyGCN: A GCN Accelerator with Hybrid Architecture”, Proc of IEEE Intl. Symp on High Performance Computer Architecture (HPCA), pp.15-29, 2020.

[10] T. Geng, et al., ”AWB-GCN: A Graph Convolutional Network Accelerator with Runtime Workload Rebalancing”, Proc. of IEEE/ACM Intl. Symp. Microarchitecture(MICRO), pp.922-936, 2020.

[11] Tong Geng, et al., ”I-GCN: A Graph Convolutional Network Accelerator with Runtime Locality Enhancement through Islandization”, Proceedings of MICRO ’21: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 1051-1063, 2022.

[12] Cheng Wan, et al., ”PipeGCN: Efficient Full-Graph Training of Graph Convolutional Networks with Pipelined Feature Communication”, arXiv:2203.10428, 2022.

[13] Sumit K. Mandal, et al., ”COIN: Communication-Aware In-Memory Acceleration for Graph Convolutional Networks”, Proc of IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS, VOL. 12, NO. 2, JUNE 2022.

[14] Le Wu, et al., ”SocialGCN: An Efficient Graph Convolutional Network based Model for Social Recommendation”, arXiv:1811.02815, 2019.

[15] Liang Chen, et al., ”Friend Recommendation Based on Multi-Social Graph Convolutional Network”, Proc of 2023 IEEE PES 15th Asia-Pacific Power and Energy Engineering Conference (APPEEC), pp. 43618-43629, 2023.

[16] Ron Levie, et al., ”CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters”, Proc of IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 67, NO. 1, JANUARY 1, 2019, pp. 97-109, 2019.

[17] Dongxiao He, et al., ”Community-Centric Graph Convolutional Network for Unsupervised Community Detection”, Proc of of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), pp. 3515-3521, 2021.

[18] Zhigang Jin, et al.,”Heterogeneous Graph Network Embedding for Sentiment Analysis on Social Media”, Proc of the Springer Nature SharedIt content-sharing initiative, pp. 81-95, 2021.

[19] Xiaoyang Liu, et al., ”Social network sentiment classification method combined Chinese text syntax with graph convolutional neural network”, Proc of Egyptian Informatics Journal, pp. 1-12, 2022.

[20] Guoyong Hu, et al., ”Multi-depth Graph Convolutional Networks for Fake News Detection”, Natural Language Processing and Chinese Computing 8th CCF International Conference, NLPCC 2019, pp. 698-710.

[21] Youze Wang, et al., ”Fake News Detection via Knowledge-driven Multimodal Graph Convolutional Networks”, ICMR ’20: Proceedings of the 2020 International Conference on Multimedia Retrieval, pp. 540-547, 2020.

[22] Shengsheng Qian, et al., ”Knowledge-aware Multi-modal Adaptive Graph Convolutional Networks for Fake News Detection”, ACM Transactions on Multimedia Computing, Communications, and ApplicationsVolume 17, pp. 1-23, 2021.

[23] Shengnan Guo, et al., ”Attention Based Spatial-Temporal Graph Convolutional

Networks for Traffic Flow Forecasting", Proceedings of the I Conference on Artificial Intelligence, pp. 922-929, 2019.

[24] Mingqi Lv, et al., "Temporal Multi-Graph Convolutional Network for Traffic Flow Prediction", Proc of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 22, NO. 6, JUNE 2021, pp. 3337-3347, 2021.

[25] Yue Song, et al., "Short-Term Forecasting Based on Graph Convolution Networks and Multiresolution Convolution Neural Networks for Wind Power", Proc of IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 19, NO. 2, FEBRUARY 2023, pp. 1691-1702. 2023.

[26] Cheolmin Kim, et al., "Graph Convolutional Neural Networks for Optimal Load Shedding under Line Contingency", Proc of IEEE Power & Energy Society General Meeting (PESGM), 2019.

[27] Marinka Zitnik, et al., "Modeling polypharmacy side effects with graph convolutional networks", Proceedings of Bioinformatics, Volume 34, Issue 13, pp. 457-466, 2018.

[28] Tengfei Ma, et al., "Drug Similarity Integration Through Attentive Multi-view Graph Auto-Encoders", Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3477-3483, 2018.

[29] Kyle Yingkai Gao, et al., "Interpretable Drug Target Prediction Using Deep Neural Representation", Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), 2018.

[30] Daixin Wang, et al., "A Semi-Supervised Graph Attentive Network for Financial Fraud Detection", Proc of 2019 IEEE International Conference on Data Mining (ICDM), pp. 598-607, 2019.

[31] Ranran Li, et al., "Internet Financial Fraud Detection Based on Graph Learning", Proc of IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, VOL. 10, NO. 3, JUNE 2023, pp. 1394-1401, 2023.

[32] Y. Lecun, et al., "Gradient-based learning applied to document recognition", Proceedings of the IEEE, 1998.

[33] Zhe Zhou, et al., "GNNear: Accelerating Full-Batch Training of Graph Neural Networks with near-Memory Processing", PACT '22: Proceedings of the International Conference on Parallel Architectures and Compilation TechniquesOctober 2022, pp. 54-68, 2022.

[34] Keyulu Xu, et al., "How Powerful are Graph Neural Networks?", Proc of International Conference on Learning 2019, 2019.

[35] Hamilton, et al., "Inductive representation learning on large graphs" in Advances in Neural Information Processing Systems, pp. 1024-1034, 2017.

[36] Velickovic, et al., "Graph attention networks", in Proc. Int. Conf. Learn. Represent. (ICLR), pp. 1-12, 2018.

[37] Eduardo Alcaín, et al., "Hardware Architectures for Real-Time Medical Imaging", Proc of Hardware Architectures for Real Time Image Processing, 2021.

[38] Yujie Wang, "Artificial-Intelligence integrated circuits: Comparison of GPU, FPGA and ASIC", Proceedings of the 3rd International Conference on Signal Processing and Machine Learning, pp. 99-104, 2023.

[39] Shengwen Liang, et al., "EnGN: A High-Throughput and Energy-Efficient Accelerator for Large Graph Neural Networks", Proceedings of IEEE TRANSACTIONS ON COMPUTERS, VOL. 70, NO. 9, SEPTEMBER 2021, pp. 1511-1525, 2021.

[40] Adam Auten, et al., "Hardware Acceleration of Graph Neural Networks", Proceedings of 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020.

[41] Jiajun Li, et al., "GCNAX: A Flexible and Energy-efficient Accelerator for Graph Convolutional Neural Networks", Proceedings of GCNAX: A Flexible and Energy-efficient Accelerator for Graph Convolutional Neural Networks, pp. 775-778, 2021.

[42] Kevin Kiningham, et al., "GRIP: A Graph Neural Network Accelerator Architecture", Proceedings of IEEE TRANSACTIONS ON COMPUTERS, VOL. 72, NO. 4, APRIL 2023, pp. 914-925, 2023.

[43] Bingyi Zhang, et al., "Hardware Acceleration of Large Scale GCN Inference", Proceedings of 2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 61-68, 2020.

[44] Vidushi Dadu, el al, ”PolyGraph: Exposing the Value of Flexibility for Graph Processing Accelerators”, Proceedings of 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), pp. 595-608, 2021.

[45] Bingyi Zhang, et al., ”BoostGCN: A Framework for Optimizing GCN Inference on FPGA”, Proceedings of 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 29-39, 2021.

[46] Zhe Zhou, et al., ”BlockGNN: Towards Efficient GNN Acceleration Using Block-Circulant Weight Matrices”, Proceedings of 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 1009-1014, 2021.

[47] Shengwen Liang, et al., ”DeepBurning-GL: an automated framework for generating graph neural network accelerators”, Proceedings of the 39th International Conference on Computer-Aided Design, pp. 1-9. 2020.

[48] Haoran You, et al., ”GCoD: Graph Convolutional Network Acceleration via Dedicated Algorithm and Accelerator Co-Design”, Proceedings of 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 460-474, 2022.

[49] Yongan Zhang, et al., ”G-CoS: GNN-Accelerator Co-Search Towards Both Better Accuracy and Efficiency”, Proceedings of 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2021.

[50] Jiaxian Chen, et al., ”Towards efficient allocation of graph convolutional networks on hybrid computation-in-memory architecture”, Proceedings of Special Focus on Near-memory and In-memory Computing, 2021.

[51] Yu Huang, et al., ”Accelerating Graph Convolutional Networks Using Crossbar-based Processing-In-Memory Architectures”, Proceedings of 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 1029-1042, 2022.

[52] Shyam A. Tailor, el al., ”Do We Need Anisotropic Graph Neural Networks?”, Proceedings of ICLR 2022, pp. 1-19, 2022.

[53] Xinkai Song, et al., ”Cambricon-G: A Polyvalent Energy-Efficient Accelerator for Dynamic Graph Neural Networks”, Proceedings of IEEE TRANSACTIONS ON COMPUTER-

AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 41, NO. 1, JANUARY 2022, pp. 116-128, 2022.

[54] Nagadastagiri Challapalle, et al., "Crossbar based Processing in Memory Accelerator Architecture for Graph Convolutional Networks", Proceedings of Crossbar based Processing in Memory Accelerator Architecture for Graph Convolutional Networks, pp. 1-9. 2021.

[55] Cen Chen, et al., "ReGNN: A Redundancy-Eliminated Graph Neural Networks Accelerator", Proceedings of 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 429-443, 2022.

[56] Jacob R. Stevens, et al., "GNNerator: A Hardware/Software Framework for Accelerating Graph Neural Networks", Proceedings of 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 955-960, 2021.

[57] Tao Yang, et al., "PIMGCN: A ReRAM-Based PIM Design for Graph Convolutional Network Acceleration", Proceedings of 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 583-588, 2021.

[58] Chengming Zhang, et al., "H-GCN: A Graph Convolutional Network Accelerator on Versal ACAP Architecture", Proceedings of 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL), pp. 200-208, 2022.

[59] Zhuofu Tao, et al., "LW-GCN: A Lightweight FPGA-based Graph Convolutional Network Accelerator", Proceedings of ACM Transactions on Reconfigurable Technology and Systems, Volume 16, Issue 1, pp. 1-19, 2022.

[60] Feng Shi, et al., "VersaGNN: a Versatile accelerator for Graph neural networks", arXiv:2105.01280, 2021.

[61] Cen Chen, et al., "DyGNN: Algorithm and Architecture Support of Dynamic Pruning for Graph Neural Networks", Proceedings of DyGNN: Algorithm and Architecture Support of Dynamic Pruning for Graph Neural Networks, pp. 1201-1206, 2021.

[62] Weian Yan, et al., "FPGAN: An FPGA Accelerator for Graph Attention Networks With Software and Hardware Co-Optimization", Proceedings of IEEE Access Volume: 8, pp. 171608-171620, 2020.

[63] Yi Chien Lin, et al., ”GCN Inference Acceleration using High-Level Synthesis”, Proceedings of 2021 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-6, 2021.

[64] Miryeong Kwon, et al., ”Hardware/Software Co-Programmable Framework for Computational SSDs to Accelerate Deep Learning Service on Large-Scale Graphs”, Proceedings of the 20th USENIX Conference on File and Storage Technologies, pp. 147-163, 2022.

[65] Ranggi Hwang, et al., ”GROW: A Row-Stationary Sparse-Dense GEMM Accelerator for Memory-Efficient Graph Convolutional Neural Networks”, Proceedings of 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 42-55, 2023.

[66] Luchang Ding, et al., ”An FPGA Implementation of GCN with Sparse Adjacency Matrix”, Proceedings of 2019 IEEE 13th International Conference on ASIC (ASICON), pp. 1-4, 2019.

[67] Sudipta Mondal, et al., ”GNNIE: GNN inference engine with load-balancing and graph-specific caching”, Proceedings of the 59th ACM/IEEE Design Automation Conference, pp. 565 - 570, 2022.

[68] JOSÉ ROMERO HUNG, et al., ”ACE-GCN: A Fast Data-driven FPGA Accelerator for GCN Embedding”, Proceedings of ACM Transactions on Reconfigurable Technology and Systems, Vol. 14, No. 4, Article 21, pp. 21:1-21:23, 2021.

[69] Yixuan Luo, et al., ”CoDG-ReRAM: An Algorithm-Hardware Co-design to Accelerate Semi-Structured GNNs on ReRAM”, Proceedings of 2022 IEEE 40th International Conference on Computer Design (ICCD), pp. 280-289, 2022.

[70] Rishov Sarkar, et al., ”FlowGNN: A Dataflow Architecture for Real-Time Workload-Agnostic Graph Neural Network Inference”, Proceedings of 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 1099-1122, 2023.

[71] Teng Tian, et al., ”FP-GNN: Adaptive FPGA accelerator for Graph Neural Networks”, Proceedings of Future Generation Computer Systems Volume 136, pp. 294-310, 2022.

[72] Gongjian Sun, et al., ”Multi-Node Acceleration for Large-Scale GCNs”, Proceedings of IEEE TRANSACTIONS ON COMPUTERS, VOL. 71, NO. 12, pp. 3140-3152, 2022.

[73] Bingyi Zhang, et al., ”Low-latency Mini-batch GNN Inference on CPU-FPGA Heterogeneous Platform”, Proceedings of 2022 IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC), pp. 11-21, 2022.

[74] Cong Liu, et al., ”ReGNN: A ReRAM-based Heterogeneous Architecture for General Graph Neural Networks”, Proceedings of the 59th ACM/IEEE Design Automation Conference, pp. 469-574, 2022.

[75] Han Li, et al., ”Hardware Acceleration for GCNs via Bidirectional Fusion”, Proceedings of IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 20, NO. 1, pp. 66-69, 2021.

[76] Yu Huang, et al., ”ReaDy: A ReRAM-Based Processing-in-Memory Accelerator for Dynamic Graph Convolutional Networks”, Proceedings of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 41, NO. 11, pp. 3567-3578, 2022.

[77] Tao Yang, et al., ”PASGCN: An ReRAM-Based PIM Design for GCN With Adaptively Sparsified Graphs”, Proceedings of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 42, NO. 1, pp. 150-163, 2023.

[78] Rishov Sarkar, et al., ”FlowGNN: A Dataflow Architecture for Real-Time Workload-Agnostic Graph Neural Network Inference”, Proceedings of 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 1099-1112, 2023.

[79] Christodoulos Peltekis, et al., ”FusedGCN: A Systolic Three-Matrix Multiplication Architecture for Graph Convolutional Networks”, Proceedings of 2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 93-97, 2022.

[80] Hai Jin, et al., ”Accelerating Graph Convolutional Networks Through a PIM-Accelerated Approach”, Proceedings of IEEE TRANSACTIONS ON COMPUTERS, VOL. 72, NO. 9, pp. 2628-2640, 2023.

[81] Zhihui Zhang, et al., ”ZIPPER: Exploiting Tile- and Operator-level Parallelism for General and Scalable Graph Neural Network Acceleration”, arXiv:2107.08709, pp. 1-13, 2021.

[82] Yunki Han, et al., ”EGCN: An Efficient GCN Accelerator for Minimizing Off-Chip Memory Access”, Proceedings of IEEE TRANSACTIONS ON COMPUTERS, VOL. 71, NO. 12, pp. 3127-3139, 2022.

[83] Mingi Yoo, et al., ”Making a Better Use of Caches for GCN Accelerators with Feature Slicing and Automatic Tile Morphing”, Proceedings of IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 20, NO. 2, pp. 102-105, 2021.

[84] Wenkai Tang, et al., ”GPGCN: A General-Purpose Graph Convolution Neural Network Accelerator Based on RISC-V ISA Extension”, Proceedings of ELECTRONICS Volume 11, Issue 22, 2022.

[85] Mingi Yoo, et al., ”SGCN: Exploiting Compressed-Sparse Features in Deep Graph Convolutional Network Accelerators”, Proceedings of 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 1-14, 2023.

[86] Yu Zhu, et al., ”Exploiting Parallelism with Vertex-Clustering in Processing-In-Memory-based GCN Accelerators”, Proceedings of Design, Automation and Test in Europe Conference and Exhibition, pp. 652-657, 2022.

[87] Wei Yuan, et al., ”QEGCN: An FPGA-based accelerator for quantized GCNs with edge-level parallelism”, Proceedings of Journal of Systems Architecture, pp. 1-13, 2022.

[88] Bingyi Zhang, et al., ”GraphAGILE: An FPGA-Based Overlay Accelerator for Low-Latency GNN Inference”, Proceedings of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 34, NO. 9, pp. 2580-2597, 2023.

[89] Mingi Yoo, et al., ”Slice-and-Forge: Making Better Use of Caches for Graph Convolutional Network Accelerators”, Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, pp. 40-53, 2022.

[90] Atefeh Sohrabizadeh, et al., ”StreamGCN: Accelerating Graph Convolutional Networks with Streaming Processing”, Proceedings of 2022 IEEE Custom Integrated Circuits Conference (CICC), pp. 1-8, 2022.

[91] Qixuan Cheng, et al., ”Towards a Deep-Pipelined Architecture for Accelerating Deep GCN on a Multi-FPGA Platform”, Proceedings of Algorithms and Architectures for Parallel Processing(ICA3PP 2020), pp. 528-547, 2020.

[92] Wentao Hou, et al., ”NTGAT: A Graph Attention Network Accelerator with Runtime Node Tailoring”, Proceedings of the 28th Asia and South Pacific Design Automation Conference, pp. 645-650, 2023.

[93] Jiaxian Chen, et al., ”GCIM: Toward Efficient Processing of Graph Convolutional Networks in 3D-Stacked Memory”, Proceedings of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 41, NO. 11, pp. 3579-3590, 2023.

[94] Zeyu Zhu, et al., ”MEGA: A Memory-Efficient GNN Accelerator Exploiting Degree-Aware Mixed-Precision Quantization”, Proceedings of 2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 124-138, 2024.

[95] Xinyang Shen, et al., ”A heterogeneous 3-D stacked PIM accelerator for GCN-based recommender systems”, Proceedings of CCF Transactions on High Performance Computing Volume 6, pp. 150-163, 2024.

[96] Mohsin Shan, et al., ”MergePath-SpMM: Parallel Sparse Matrix-Matrix Algorithm for Graph Neural Network Acceleration”, Proceedings of MergePath-SpMM: Parallel Sparse Matrix-Matrix Algorithm for Graph Neural Network Acceleration, pp. 145-156, 2023.

[97] Shaolin Ran, et al., ”Software-hardware co-design for accelerating large-scale graph convolutional network inference on FPGA”, Proceedings of Neurocomputing Volume 532, pp. 129-140, 2023.

[98] Kai Zhong, et al., ”CoGNN: An Algorithm-Hardware Co-Design Approach to Accelerate GNN Inference With Minibatch Sampling”, Proceedings of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 42, NO. 12, pp. 4883-4896, 2023.

[99] Zerong He, et al., ”FTW-GAT: An FPGA-Based Accelerator for Graph Attention Networks With Ternary Weights”, Proceedings of FTW-GAT: An FPGA-Based Ac-

celerator for Graph Attention Networks With Ternary Weights, pp. 4211-4215, 2023.

[100] Gopikrishnan Raveendran Nair, et al., ”FPGA Acceleration of GCN in Light of the Symmetry of Graph Adjacency Matrix”, Proceedings of FPGA Acceleration of GCN in Light of the Symmetry of Graph Adjacency Matrix, pp. 1-6, 2023.

[101] Xi Xie, et al., ”Accel-GCN: High-Performance GPU Accelerator Design for Graph Convolution Networks”, Proceedings of 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 1-9, 2023.

[102] Xiao Qin, et al., ”SEIGN: A Simple and Efficient Graph Neural Network for Large Dynamic Graphs”, Proceedings of 2023 IEEE 39th International Conference on Data Engineering (ICDE), pp. 285-2863, 2023.

[103] Yuntao Wei, et al., ”IMGA: Efficient In-Memory Graph Convolution Network Aggregation With Data Flow Optimizations”, Proceedings of IMGA: Efficient In-Memory Graph Convolution Network Aggregation With Data Flow Optimizations, pp. 4695-4705, 2023.

[104] Yufeng Wang, et al., ”TGOpt: Redundancy-Aware Optimizations for Temporal Graph Attention Networks”, Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, pp. 354-368, 2023.

[105] Salma Afifi, et al., ”GHOST: A Graph Neural Network Accelerator using Silicon Photonics”, Proceedings of ACM Transactions on Embedded Computing Systems, Volume 22, Issue 5, pp. 1-25, 2023.

[106] Siddhartha Raman Sundara Raman, et al., ”NEM-GNN: DAC/ADC-less, Scalable, Reconfigurable, Graph and Sparsity-Aware Near-Memory Accelerator for Graph Neural Networks”, Proceedings of ACM Transactions on Architecture and Code Optimization, Volume 21, Issue 2, pp. 1-26, 2024.

[107] Paul Chen, et al., ”Exploiting On-Chip Heterogeneity of Versal Architecture for GNN Inference Acceleration”, Proceedings of Exploiting On-Chip Heterogeneity of Versal Architecture for GNN Inference Acceleration, pp. 219-227, 2023.

[108] Bingyi Zhang, et al., ”Dynasparse: Accelerating GNN Inference through Dynamic Sparsity Exploitation”, Proceedings of 2023 IEEE International Parallel and Dis-

tributed Processing Symposium (IPDPS), pp. 233-244, 2023.

[109] Stefan Abi-Karam, et al., "GNNBuilder: An Automated Framework for Generic Graph Neural Network Accelerator Generation, Simulation, and Optimization", Proceedings of 2023 33rd International Conference on Field-Programmable Logic and Applications (FPL), pp. 212-218, 2023.

[110] Hanqing Zeng, et al., "GraphACT: Accelerating GCN Training on CPU-FPGA Heterogeneous Platforms", Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 255-265, 2020.

[111] Xiaobing Chen, et al., "Rubik: A Hierarchical Architecture for Efficient Graph Neural Network Training", Proceedings of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 41, NO. 4, APRIL 2022, pp. 936-949, 2022.

[112] Yi-Chien Lin, et al., "HP-GNN: Generating High Throughput GNN Training Implementation on CPU-FPGA Heterogeneous Platform", Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 123-133, 2022.

[113] Zhe Zhou, et al., "GCNear: A Hybrid Architecture for Efficient GCN Training with Near-Memory Processing", arXiv:2111.00680, pp.1-14, 2021.

[114] Chen Wu, et al., "SkeletonGCN: A Simple Yet Effective Accelerator For GCN Training", Proceedings of SkeletonGCN: A Simple Yet Effective Accelerator For GCN Training, pp. 445-451, 2022.

[115] Letian Zhao, et al., "HuGraph: Acceleration of GCN Training on Heterogeneous FPGA Clusters with Quantization", Proceedings of IEEE Conference on High Performance Extreme Computing (HPEC), pp. 1-7, 2022.

[116] Chukwufumnanya Ogbogu, et al., "Data Pruning-enabled High Performance and Reliable Graph Neural Network Training on ReRAM-based Processing-in-Memory Accelerators", Proceedings of ACM Transactions on Design Automation of Electronic Systems, pp. 1-32, 2024.

[117] Hongwu Peng, et al., "MaxK-GNN: Extremely Fast GPU Kernel Design for Accelerating Graph Neural Networks Training", Proceedings of the 29th ACM Interna-

tional Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, pp. 683-698, 2024.

[118] Olle Hansson, et al., ”Deep Quantization of Graph Neural Networks with Run-Time Hardware-Aware Training”, Proceedings of Applied Reconfigurable Computing. Architectures, Tools, and Applications(ARC 2024), pp. 33-47, 2024.

[119] Keren Zhou, et al., ”FASTEN: Fast GPU-accelerated Segmented Matrix Multiplication for Heterogenous Graph Neural Networks”, Proceedings of the 38th ACM International Conference on Supercomputing, pp. 511-524, 2024.

[120] Pratyush Dhingra, et al., ”FARe: Fault-Aware GNN Training on ReRAMbased PIM Accelerators”, Proceedings of 2024 Design, Automation & Test in Europe Conference (DATE 2024), pp. 1-6, 2024.

[121] Yi-Chien Lin, et al., ”HitGNN: High-Throughput GNN Training Framework on CPU+Multi-FPGA Heterogeneous Platform”, Proceedings of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 35, NO. 5, pp. 707-719, 2024.

[122] Zirui Liu, et al., ”RSC: Accelerate Graph Neural Networks Training via Randomized Sparse Computations”, Proceedings of the 40th International Conference on Machine Learning, pp. 1-18, 2023.

[123] Roger Waleffe, et al., ”MariusGNN: Resource-Efficient Out-of-Core Training of Graph Neural Networks”, Proceedings of the Eighteenth European Conference on Computer Systems, pp. 144-161, 2023.

[124] Yi-Chien Lin, et al., ”HyScale-GNN: A Scalable Hybrid GNN Training System on Single-Node Heterogeneous Architecture”, Proceedings of HyScale-GNN: A Scalable Hybrid GNN Training System on Single-Node Heterogeneous Architecture, pp. 557-567, 2023.

[125] Yuke Wang, et al., ”MGG: Accelerating Graph Neural Networks with Fine-Grained Intra-Kernel Communication-Computation Pipelining on Multi-GPU Platforms”, Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation, pp. 779-795, 2023.

[126] Fuping Niu, et al., ”FlashGNN: An In-SSD Accelerator for GNN Training”, Proceedings of 2024 IEEE International Symposium on High-Performance Computer

Architecture (HPCA), pp. 361-378, 2024.

[127] Zheng Qu, et al., ”TT-GNN: Efficient On-Chip Graph Neural Network Training via Embedding Reformation and Hardware Optimization”, Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 452-464, 2023.

[128] Junfu Wang, et al., ”Bi-GCN: Binary Graph Convolutional Network”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1561-1570, 2021.

[129] Jiajun Li, et al., ”SGCNAX: A Scalable Graph Convolutional Neural Network Accelerator With Workload Balancing”, Proceedings of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 33, NO. 11, NOVEMBER 2022, pp. 2834-2845, 2022.

[130] Jaeyoung Kang, et al., ”RelHD: A Graph-based Learning on FeFET with Hyperdimensional Computing”, Proceedings of RelHD: A Graph-based Learning on FeFET with Hyperdimensional Computing, pp. 553-560, 2022.

[131] Ningyi Liao, et al., ”Scalable decoupling graph neural network with feature-oriented optimization”, Proceedings of The VLDB Journal(2024), pp. 1-17, 2024.

[132] Haoyang Wang, et al., ”SaGNN: a Sample-based GNN Training and Inference Hardware Accelerator”, Proceedings of 2023 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, 2023.

[133] Zehua Cheng, et al., ”Bandwidth Reduction using Importance Weighted Pruning on Ring AllReduce”, arXiv:1901.01544, pp. 1-6, 2019.

[134] Yahoo! altavista web page hyperlink connectivity graph, circa 2002, http://webscope.sandbox.yahoo.com/, 2012.

[135] Avery Ching, et al., ”One trillion edges: graph processing at Facebook-scale”, Proceedings of the VLDB Endowment, Volume 8, Issue 12, pp. 1804-1815, 2015.