一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CPSY2020-7,DC2020-7(2020-07)

深層学習を用いたCGRAの効率的なアプリケーションマッピング手法

小島　拓也†　　大和田　彩夏†　　天野　英晴†

† 慶應義塾大学大学院 理工学研究科　223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †{tkojima,ohwawa,hunga}@am.ics.keio.ac.jp

あらまし　粗粒度再構成可能デバイス CGRA(Coarse-grained reconfigurable architecture) は再構成の粒度をワード単位にすることで、FPGA などのビット単位で再構成を行うデバイスと比べ高いエネルギー効率を達成する。さらに、この粒度の粗さはコンパイラが担う処理を削減することにも繋がる。しかし、未だ配置配線を含む CGRA のアプリケーションマッピングは時間のかかる処理である。コンパイラが大規模なアプリケーションをマップする際、 配線の複雑さなどからリソース利用効率の悪化や最悪の場合はマッピング失敗という結果を招く。特にマッピングに失敗した場合、コンパイラはアプリケーションカーネルをいくつかのサブカーネルに分割し、同じフローをやり直す必要があるが、これはコンパイル時間を増大させてしまう。そこで、マップ不能なカーネルに対する不必要な処理を削減するために、コンパイラは特に時間のかかる配置配線を行わずに、カーネルがマップ可能か、加えてリソース消費量を推定する必要がある。本研究は、グラフ構造向けの深層学習モデル DGCNN(Deep Graph Convolutional Neural Network) を用いた推定モデルを提案する。さらに、これを用いて実際の配置配線を行わずに、カーネルの分割し効率的にマッピングを行う手法を提案する。評価の結果、提案したモデルは 92.3%の精度でマッピングの可否を推定し、リソース消費量は 6.89%の平均誤差で見積もることができた。このモデルを用いた新たなマッピングフローによって不必要な配置配線処理を 47%削減することができた。

キーワード　CGRA, グラフ畳み込み, アプリケーションマッピング

Takuya KOJIMA†, Ayaka OHWADA†, and Hideharu AMANO†

† Graduate School of Science and Technology, Keio University　Hiyoshi 3–14–1, Kohoku-ku, Yokohama, Kanagawa, 223–8522 Japan
E-mail: †{tkojima,ohwawa,hunga}@am.ics.keio.ac.jp

1. はじめに

粗粒度再構成可能デバイス CGRA(Coarse-grained reconfigurable architecture) は近年注目を集める DSA(Domain Specific Architecture) を実現するためのハードウェアプラットフォームとして期待される。CGRA はデータフローレベルの再構成が可能で、消費電力と性能の両方に優れるデバイスである。一般に図 1 のような演算処理ユニット (PE) を 2 次元のアレイにして持ち、隣接する PE は互いに相互接続網で結ばれ、PE で実行する演算や接続網の構成をアプリケーションに応じて切り替えることができる。PE アレイは十分に最適化された設計を利用してチップとして製造されるため、プログラマビリティを保持したまま ASIC に近いエネルギー効率を達成する。また、FPGA と比較して CGRA の粗い再構成粒度は CAD ツールにおけるコンパイル処理の複雑さを緩和する。したがって、CGRA はコンパイル時間削減のために FPGA 上のオーバーレイとしても利用される [1, 2].

CGRA で実行するアプリケーションカーネルはデータフローグラフ (DFG) として表現される。DFG のノードは演算を表し、ノード間のエッジは演算間のデータ依存を表す。コンパイラは演算ノードを PE へ配置し、依存する PE 間を配線資源を用いて配線する。この処理はマッピングと呼ばれる。しかし、複雑で大きな DFG を PE アレイへマッピングする場合、探索空間が広くなり完了まで数時間を所要する場合がある [3–6]。さらに、数時間にわたりマッピングを試行した結果失敗に終わると、コンパイラが自動的で、あるいはプログラマが手動で DFG を複数の小さな DFG に分割し、再びマッピングを試行する。こうした時間の浪費はアプリケーションの開発効率を悪化させる。しかしながら、このような従来の設計フローにおける非効率さに対処する研究は手薄である。

本研究は、グラフ構造向けの深層学習モデル DGCNN(Deep Graph Convolutional Neural Network) を用いた、効率的なマッピングフローを提案する。DGCNN は任意のグラフ構造を入力として利用でき、グラフ分類を行うモデルとして提案されている [7]。提案フローでは DGCNN を用いて実際の配置配線を行わずにマッピング対象の DFG が PE アレイにマップ可能

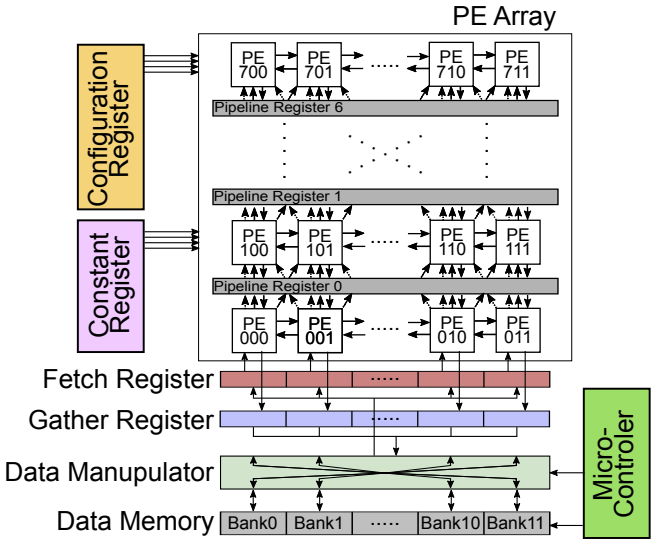


図 1: VPCMA の概要

であるかどうかを推定する。これによって、不要なマッピング試行を削減し、より高速に有効なマッピングを得ることができる。さらに、本手法で用いる DGCNN は対象の DFG が利用するハードウェア消費量も見積もる。本フローは対象の DFG がマップ可能であると推定された場合でも、分割した方がリソースの利用効率が向上すると判断した場合にはさらに分割を行う。

2. 背　　景

2.1 Coarse-Grained Reconfigurable Architectures

CGRA は一般にアクセラレータとして利用され、アプリケーションの計算強度の高いループ部を効率的に処理する。CGRA は再構成の方式に応じて次の 2 つのクラスに分類される: 1) 空間的マッピング、2) 時間的マッピング。空間的マッピング方式に則った CGRA として RSPA[8], DT-CGRA[9] などがある。この方式では、DFG のすべての演算ノードを一度に PE アレイにマップし、タスクが完了するまで同じ構成 (コンフィギュレーション) を利用し続ける。スループットを向上させるためにコンパイラはなるべく少ない数の PE にマッピングし、できる限り多くのイテレーションをループ展開できるようマッピングを最適化する [3, 10, 11]。

対して、時間的マッピングの CGRA として ADRES[12] や FloRA[13] などが提案されている。この方式ではサイクル単位で PE アレイのコンフィギュレーションを切り替え、時分割的に PE アレイを利用する。したがって、コンパイラは配置配線に加えてスケジューリングも行う必要がある。一般に、ソフトウェアパイプラインの 1 種であるモジュロスケジューリングが用いられ、コンパイラはイテレーションの発行間隔 (Initial Interval) を最小化するよう最適化を行う [14, 15].

時間的マッピングは毎サイクルコンテキストスイッチを行うため、消費エネルギーのオーバーヘッドが大きく、空間的マッピング方式の CGRA の方がエネルギー効率に優れる。しかし、複雑で大規模な DFG をマッピングする場合、空間的マッピングは時間的マッピングより不利になる。時間的マッピングの場合、スケジュール時間を延長することで仮想的に PE アレイのサイズを拡張することができるが、空間的マッピングでは物理的に存在する PE しか一度に利用することができない。したがって、対象の DFG が PE アレイに収まりきらない場合は、DFG を分割する必要がある。本研究では、比較的大きなアプリケーションカーネルを想定し、空間的マッピング方式の CGRA に焦点を当てるが、提案する推定モデルや DFG 分割手法は時間的マッピング方式の CGRA にも利用可能である。

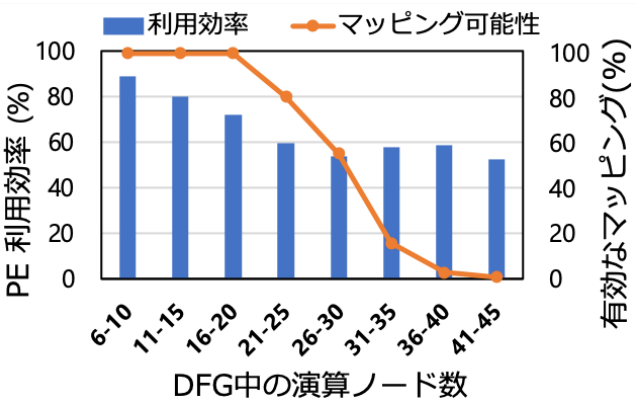


図 2: マッピング可能性とリソース利用効率の分析

2.2 対象アーキテクチャ

本研究では低電力 CGRA として提案された VPCMA(Variable Pipelined Cool Mega Array)[16] を評価に用いる。図 1 に VPCMA の基本的な構成を示す。他の CGRA と同様に PE アレイを持ち、空間的マッピング方式を採用している。各 PE は計算資源の ALU(arithmetic logic unit) と配線資源の SE(switching element) を持つ。PE アレイのサイズは 8×12 である。PE はレジスタファイルを持たず、PE にはクロック信号が入力する必要がない。その代わり、PE の行間にはパイプラインレジスタが挿入されている。

小規模なマイクロコントローラはメモリと PE アレイとの間のデータ転送を制御する。データマニピュレータはメモリと PE アレイとの間で効率的にデータ転送ができるよう設けられた接続網である。これによって、VPCMA は柔軟性と低消費電力性を同時に達成する。

予備評価として様々サイズの DFG に対して有効なマッピングが存在するのか、またリソースの利用効率を分析した。図 2 はその結果である。ここで、N_{op} を DFG の演算ノード数、マッピングに必要な最小アレイサイズの幅と高さをそれぞれ W と H としたときのリソース利用効率は $\frac{N_{op}}{W \times H}$ として計算される。つまり、$(W \times H - N_{op})$ 個の PE は配線だけに利用されているか、アイドル状態である。20 ノード以下の DFG であればほぼすべての DFG がマッピング可能であり、利用効率も 70%を超える。しかし、それより大きい DFG では急激にマッピング可能性が低下し、利用効率も持ちる。これは大規模な DFG の場合、依存する PE 間の距離が大きくなる傾向にあり、その結果多くの PE が配線にのみ利用されるためである。これらの予備評価から複雑で大規模な DFG の有効なマッピングを得るために、さらには、計算資源をより効率的に利用させるために DFG を適切に分割するよう必要があると分かる。本研究で提案するフローによってこの問題を解決する。

3. 深層学習による推定モデル

深層学習は画像識別や自然言語処理など様々な分野で応用さ

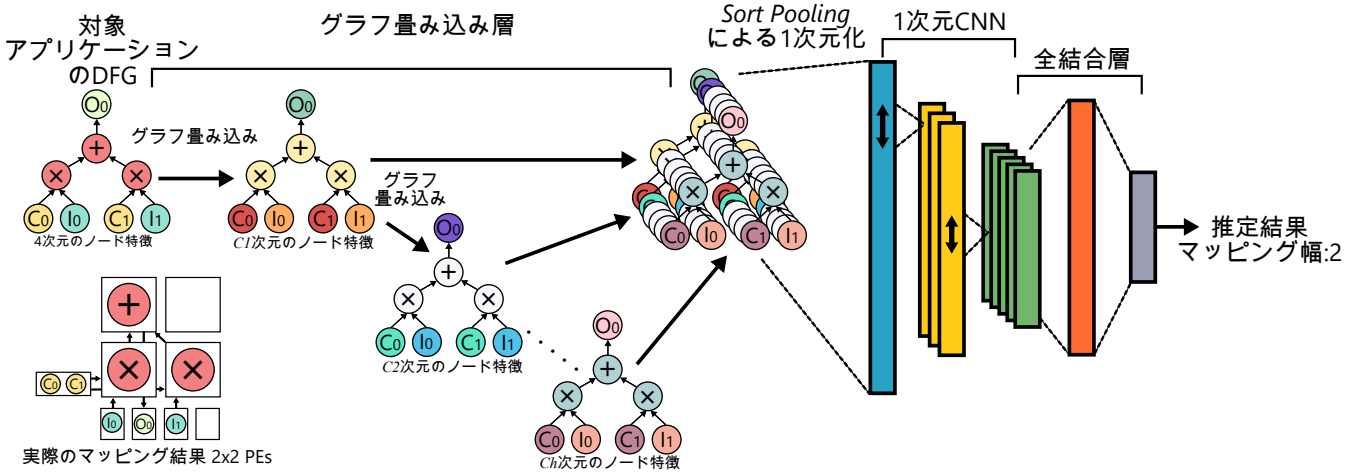


図 3: DGCNN による推定モデル

表 1: DFG ノードが持つ特徴量

次元	値
1	演算ノードの opcode
2	定数レジスタの場合の定数データ
3	データ入力の場合 1 それ以外は 0
4	データ出力の場合 1 それ以外は 0

れている。中でも CNN(Convolutional Neural Network) が主流となっており、入力画像に対して小さなフィルタを用いた畳み込み演算とプーリングを繰り返すことで高いレベルの特徴抽出と位置不変性を得ている。しかし、CNN では画素の数値の並びを特徴として抽出していくため、グラフ構造のようなデータの並び順を持たないデータ構造に対してはそのまま適用することができない。したがって、既存の深層学習をグラフ構造向けに拡張する様々な試みが報告されている [17]。

DGCNN はグラフ分類を行うモデルの 1 種として提案されている [7]。DGCNN は図 3 に示すようにグラフ畳み込み層、*Sort Pooling* 層、そして古典的な 1 次元の CNN で構成される。本提案手法はマッピング対象の DFG がマッピング可能かどうか、そして PE アレイのうちの何列を消費するか (マッピング幅) を同時に推定するモデルとして DGCNN を利用する。2. 節で述べた通り、空間的マッピングではスループットの向上を目指すため、ループ展開に影響するマッピング幅を推定するのは重要である。例えば、図 3 に示すシンプルな DFG が入力された場合、学習済みの DGCNN は図中に示したような 2 列の PE にマップ可能であると推定する。このときに、実際のマッピング処理は行わずに推定を行っている。

3.1 DGCNN におけるグラフ畳み込み

DGCNN を用いる場合、n 個の演算ノードを持つ DFG は対応する隣接行列 $\mathbf{A}$ と特徴行列 $\mathbf{X} \in \mathbb{R}^{n \times c}$ に分割される。ここで、c はノードが持つ特徴量の次元数である。本手法では入力される DFG のノードは表 1 に列挙する 4 次元の特徴量を持つ。例えば、図 3 に示した DFG の場合, ノード C_0 は定数値を示すため、ノードの特徴ベクタは (N/A, C_0, 0, 0) となる。同様にデータ入力のノード I_0 の場合, 特徴ベクタは (N/A, N/A, 1, 0) となる。

このデータ表現を用いて、以下の計算に従いグラフ畳み込みが行われる。

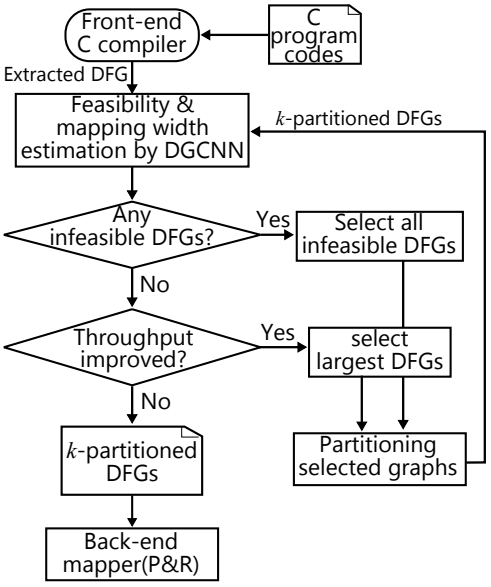


図 4: 提案するマッピングフロー

$$\mathbf{Z} = f(\widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W}) \tag{1}$$

ここで、$\widetilde{\mathbf{A}}$ は自己ループ付きの隣接行列 (i.e., $\mathbf{A}+\mathbf{I}$)、$\widetilde{\mathbf{D}}$ は $\mathbf{A}$ の次数行列、$\mathbf{W} \in \mathbb{R}^{c \times c'}$ は学習済みの重み行列、f は非線形活性化関数である。このグラフ畳み込み処理によって図 3 のように隣接ノード同士の特徴量が集約された同型のグラフが生成される。出力結果 $\mathbf{Z}$ は次の畳み込み層への入力として利用される。したがって、i 番目のグラフ畳み込み層の計算は次のように定義される。

$$\mathbf{Z}^i = f(\widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}\mathbf{Z}^{i-1}\mathbf{W}^{i-1}) \tag{2}$$

$\mathbf{Z}^0$ は $\mathbf{X}$ と考えることができ、重み行列 $\mathbf{W}^{i-1}$ は c_{i-1} 次元のノード特徴を c_i 次元に変換していく。

全 h 層のグラフ畳み込みが完了すると、各層のノード特徴は連結される。つまり、各ノードは $\sum_1^h c_i$ 次元のノード特徴を持つ。そして、連結されたノード特徴は *Sort Pooling* へと入力される。この層は後続する固定長入力の 1 次元 CNN への橋渡し的な役割を担う。これによって、DGCNN は任意の形状、任意のサイズの DFG を入力として利用することができる。DGCNN 自体のさらなる詳細に関しては [7] を参照されたい。

3.2 分類ラベル

本手法では対象 DFG を $W+1$ 種類のラベルに分類する。ここで、W は PE アレイの列数 (幅) である。もし、ある DFG がラベル “2” に分類された場合は図 3 の例のように 2 列の PE を必要とすると推定されたことを意味する。もしラベル “0” に分類された場合、この DFG はマッピング不能であることを意味する。

ただし、明らかな予測ミスをなくすために DGCNN による予測結果に加え、リソース制約に基づく単純な解析モデルによる推定結果も併用する。よって、最終的に予測されるマッピング幅 w_{pred} は以下のように求められる。

$$w_{pred} = max(w_{DGCNN}, \lceil \frac{N_{op}}{H} \rceil, N_{load}, N_{store}) \tag{3}$$

ここで、w_{DGCNN} は DGCNN による推定結果であり、H は PE アレイの高さ、N_{load}/N_{store} は DFG におけるデータ入力/出力の数を示す。本研究で利用する VPCMA の場合、PE 列あたり 1 つのデータ入力およびデータ出力ポートが利用可能である。したがって、I/O ポートの制約から N_{load} および N_{store} 未満の幅にマップすることはできない。他の CGRA でもメモリアクセスが可能な PE は各列で 1 つに制限されることが多い。さらに、いくつかの CGRA では乗算器や除算器など高機能な演算ユニットは PE 行または PE 列で共有される。そのような行単位、列単位のリソース制約がある場合は、同様に考慮する必要がある。

4. 提案フロー

本節では前述の推定モデルを利用したマッピングフローを提案する。図 4 に本フローを示す。対象アプリケーションは C 言語で記述され、フロントエンドコンパイラによってループカーネルが DFG として抽出される。抽出された DFG はマップ不能なパートがなるなるまで複数のパーティションに分割される。DFG の分割方法は後述する。

さらに、すべての分割された DFG がマップ可能になっても、そのうちのいくつかはリソースの利用効率が悪い場合がある。この場合ループ展開がうまくいかず結果的にスループットを悪化させてしまう。そこで、本フローではスループットの改善を目指してさらなる分割を行う。最終的に得られた分割後の DFG はバックエンドコンパイラによってマッピングが行われる。もし、ここで実際にはマッピング不能な DFG が見つかった場合は前のステップに戻りその DFG をさらに分割する。本研究では遺伝的アルゴリズムを用いたバックエンドコンパイラ [11] を使用する。

4.1 スループットの評価指標

DFG の分割がスループット向上に寄与するかどうかを判断するための指標が必要となる。本フローでは次のようにスループットを近似する。

$$\frac{1}{Throughput} = \sum_{i}^{k} \frac{1}{N_{unroll,i}} \tag{4}$$

$$N_{unroll,i} = \lfloor \frac{W}{w_{pred,i}} \rfloor \tag{5}$$

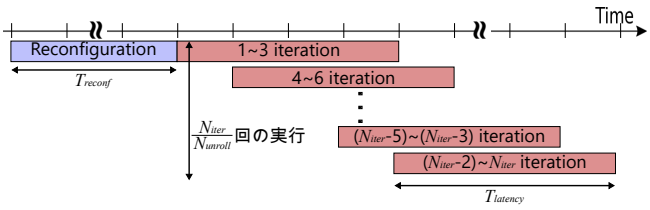
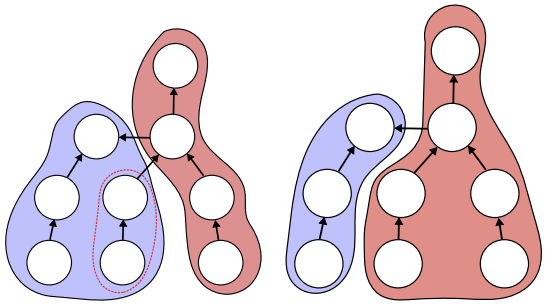


図 5: PE アレイにおけるパイプライン実行の様子



(a) 無効なパーティション　(b) 有効なパーティション

図 6: 分割の例

ここで、k は分割された総 DFG 数、$w_{pred,i}$ と $N_{unroll,i}$ は i 番目の DFG の推定マッピング幅とループ展開数を表す。2. 節で述べた通り、VPCMA の場合 W は 12 である。

$$T_{exec} = \sum_{i}^{k} \left(\frac{N_{iter}}{N_{unroll,i}} + T_{latency,i} + T_{reconf,i} \right) \tag{6}$$

$$\approx N_{iter} \sum_{i}^{k} \frac{1}{N_{unroll,i}} \tag{7}$$

ループの実行サイクル数は式 (6) により求まる。N_{iter} はイテレーション回数であり、$T_{latency,i}$ と $T_{reconf,i}$ は i 番目の分割された DFG の計算レイテンシと再構成時間を示す。ループアンローリングに加えて、空間的マッピング方式の CGRA では図 5 のように複数のイテレーションをパイプライン的に実行していくのが一般的である。この例では 3 つのイテレーションが展開され同時に発行されており、一つのイテレーションは計算に 4 サイクルを要する ($T_{latency} = 4$)。N_{iter} が十分に大きく、CGRA における再構成は十分に高速であることを考慮すると式 (7) のように近似することができる。以上から式 (4) をスループットの指標として用いる。

4.2 DFG 分割アルゴリズム

本フローで用いる分割手法は VLSI の設計などで広く利用されるヒューリスティック KL 法 [18] をベースとしている。KL 法は入力グラフをカットサイズがなるべく小さくなるように、そしてなるべく同じ大きさの二つのサブグラフへ分割する手法である。しかし、KL 法は有向エッジを考慮できないため、そのままでは図 6(a) のような互いに依存するデッドロックが発生する分割が得られる。そこで、本手法では一方向の依存となるように、どちらかの向きのデータ依存を解消するためにノードを移動する。例えば、図 6(a) において点線で囲った 2 つのノードが他方の DFG に移動され、依存が解消した図 6(b) のパーティションを得る。KL 法はランダムに分割した初期状態から反復操作を繰り返し、質を向上させていく。そこで、本フロー

表 2: 訓練データセットにおける精度

マッピング可能性の推定	
真陽性	617
真陰性	860
偽陽性	7
偽陰性	116
精度	92.3 %
偽陽性率	0.807%
マッピング幅の推定	
正確な推定	540 ケース (87.5%)
平均誤差	6.88%

では乱数のシードを変えながら複数のパーティションパターンを得て、その中で最もバランスのとれたものを採用する。

5. 評　　価

5.1 評価環境

まずはじめに、様々なサイズの DFG を約 1600 個ランダムに生成し、バックエンドコンパイラによってそれぞれの DFG のマッピング可能性、マッピング幅を得た。このうち 733 個はマッピング可能、867 個はマッピング不能なものであった。このランダムに生成したグラフと [11] で利用されたベンチマークを訓練用のデータセットとした。本評価では AMD Ryzen Threadripper 3960X CPU、128GB DDR4-SDRAM、NVIDIA RTX-2080Ti GPU、PyTorch 1.4.0 を用いて訓練を行った。

この訓練用データは未知のデータセットであり、DGCNN におけるグラフ畳み込み層の数や、中間層における特徴次元数などのいわゆるハイパーパラメータを調整する必要がある。本研究ではハイパーパラメータのチューニングに optuna[19] を用いた。ただし、紙面の都合上最適化されたハイパーパラメータの結果は割愛する。

5.2 推定モデルの精度

最適化されたネットワークに対して 100-fold cross-validation で訓練データを学習した。表 2 にその結果を示す。ここで、マッピング可能なケースを陽性、マッピング不能なケースを陰性とする。本フローでの利用ケースを考えると偽陽性は本来は不必要なマッピング試行を引き起こすため、この影響を最小限に抑えなくてはならないが、学習済みの DGCNN は十分高い 92.5% の精度と 0.807%の十分に小さい偽陽性率で推定可能であることを示した。さらに、マッピング可能であると推定した DFG のうち 87.5%は正しくマッピング幅も推定できた。それ以外の DFG に関しては推定されたマッピング幅は誤差を含むものの平均 6.88%の相対誤差にとどまった。

5.3 推定にかかる時間

DGCNN による推定にかかる時間を測定した結果バッチ処理なしで平均で 23.8 ミリ秒を所要した。本評価に用いたバックエンドコンパイラは最低でも 1 時間を要することを考慮すればリソース利用量とマッピング可能性の判定においては劇的に所要時間を削減できることがわかる。

5.4 実アプリケーションによる評価

最後に提案フローの有効性を確認する。表 3 に挙げる 5 つのアプリケーションを評価に用いた。ただし、これらの DFG は訓練データセットには含まれていない。本フローを用いたパーティショニングの結果を表 3 にまとめる。探索 DFG 数は本フローによって出現する中間 DFG も含めたすべての DFG 数である。マッピング試行回数の削減は推定モデルによってバックエンドコンパイラを用いたマッピングを行う必要がないと判断した回数である。

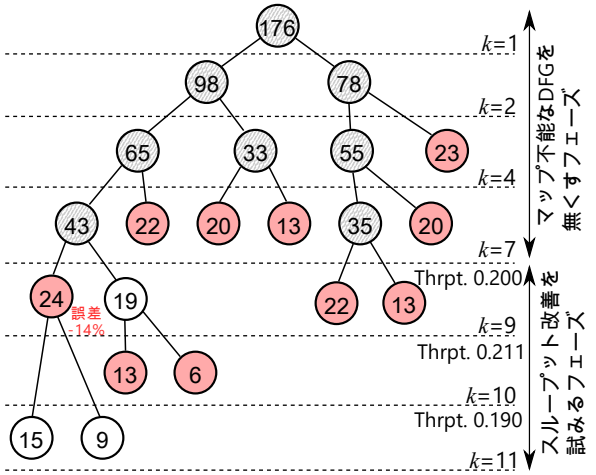


図 7: *MD5* における DFG 分割の状態変化

初めの 2 つのアプリケーション *AES, Radix4-FFT* はパーティショニングをせずともマップ可能なものであり、分割してもスループット向上には貢献しないと判断された。*RGB-to-YCC* は他のアプリケーションと比較して DFG は小さいが、多くの定数値を必要とする。評価対象の VPCMA は PE 行で 2 つの定数レジスタを共有するため、定数レジスタの不足によりマッピングが失敗する。十分に学習された DGCNN は定数値の情報を特徴として抽出できるためこのような情報を見逃さない。結果として、提案フローはこの DFG を 1 度分割し、2 つに分けるのが最良だと判断することができる。しかし、マッピング幅の推定に誤差があり実際の結果より低く見積もってしまっている。*8point-DCT* では最初のパーティショニングの後に偽陰性の推定、つまり、本来はマッピング可能な DFG をマッピング不能であると推定した。その結果、本来は分割数 3 で 0.428 のスループットが得られたにも関わらず、分割数が 4 となった。*MD5* はベンチマークの中で最も大きなアプリケーションであり、これは明らかに分割を必要とする。図 7 は各ステップにおけるパーティショニングの状態を示したものである。1 つの円が分割された DFG を示し、その中の数字は演算数を示す。影のついた円はマッピング不能な DFG であり、色付きの円が最終的なパーティショニングに含まれる DFG を示す。探索の間、偽陽性、偽陰性の推定は起きなかったが、一度だけマッピング幅の推定に誤差があった。$k = 9$ のステップですべての DFG はマップ可能であると推定されているが、スループット向上を目指し、さらなるパーティショニングを行なっている。そして、$k = 11$ でスループットが悪化したため、$k = 10$ の結果を最終的なパーティショニングとして出力する。以上よりすべてのアプリケーションで平均 47%のマッピング試行回数を削減することができた。

6. 結　　論

本研究では CGRA のための効率的なマッピングフローを提

表 3: 実アプリケーションにおけるマッピング結果

App.	DFG の特徴				分割結果			探索	マッピングの
	Op. nodes	Const. values	Mem. load	Mem. store	k	$Throughput_{model}$	$Throughput_{real}$	DFG 数	試行回数削減
AES	45	3	4	4	1	1.00	1.00	3	2
Radix4-FFT	46	6	8	8	1	1.00	1.00	3	2
RGB-to-YCC	30	15	1	1	2	1.50	1.71	3	2
8point-DCT	63	11	8	8	4	0.353	0.353	8	4
MD5	176	25	20	4	10	0.211	0.191	21	8

案した。DFG で表現される対象アプリケーションはグラフ分類に特化した深層ニューラルネットワーク DGCNN を用いて分析され、バックエンドコンパイラによる時間のかかる配置配線処理をせずにマッピング可能性とリソース利用量が見積もられる。学習済み DGCNN は 92%の精度でマッピング可能性を見積もることができ、6.89%の誤差でリソース利用量を見積もることができた。さらに、マッピング不能な DFG やリソースの利用効率が低い DFG を事前に分割することで、無駄なマッピング試行をせずにスループットの向上を図ることができた。5 つの実アプリケーションを用いた評価により、47%のマッピング試行回数を削減することができた。

謝　　辞

本研究は、JSPS 科研費 (B) ビルディングブロック型計算システムにおけるチップブリッジを用いた積層方式 (18H03125) および科研費 3 次元積層技術を応用した粗粒度再構成可能デバイスの研究 (19J21493) の助成を受けたものである。

文　　献

[1] A. Werner, F. Fricke, K. Shahin, F. Werner and M. Hübner: “Automatic Toolflow for VCGRA Generation to Enable CGRA Evaluation for Arithmetic Algorithms”, International Symposium on Applied Reconfigurable ComputingSpringer, pp. 277–291 (2019).

[2] I. Taras and J. H. Anderson: “Impact of FPGA Architecture on Area and Performance of CGRA Overlays”, 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)IEEE, pp. 87–95 (2019).

[3] D. Liu, S. Yin, G. Luo, J. Shang, L. Liu, S. Wei, Y. Feng and S. Zhou: “Data-Flow Graph Mapping Optimization for CGRA with Deep Reinforcement Learning”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2018).

[4] M. J. Walker and J. H. Anderson: “Generic connectivity-based CGRA mapping via integer linear programming”, 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)IEEE, pp. 65–73 (2019).

[5] M. Hamzeh, A. Shrivastava and S. Vrudhula: “REGIMap: register-aware application mapping on coarse-grained reconfigurable architectures (CGRAs)”, Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEEIEEE, pp. 1–10 (2013).

[6] S. Yin, D. Liu, L. Sun, L. Liu and S. Wei: “DFGNet: Mapping dataflow graph onto CGRA by a deep learning approach”, Circuits and Systems (ISCAS), 2017 IEEE International Symposium onIEEE, pp. 1–4 (2017).

[7] M. Zhang, Z. Cui, M. Neumann and Y. Chen: “An end-to-end deep learning architecture for graph classification”, Thirty-Second AAAI Conference on Artificial Intelligence (2018).

[8] Y. Kim, M. Kiemb, C. Park, J. Jung and K. Choi: “Resource sharing and pipelining in coarse-grained reconfigurable architecture for domain-specific optimization”, Design, Automation and Test in EuropeIEEE, pp. 12–17 (2005).

[9] X. Fan, H. Li, W. Cao and L. Wang: “DT-CGRA: Dual-track coarse-grained reconfigurable architecture for stream applications”, Field Programmable Logic and Applications (FPL), 2016 26th International Conference onIEEE, pp. 1–9 (2016).

[10] J. W. Yoon, A. Shrivastava, S. Park, M. Ahn, R. Jeyapaul and Y. Paek: “SPKM: A novel graph drawing based algorithm for application mapping onto coarse-grained reconfigurable architectures”, Proceedings of the 2008 Asia and South Pacific Design Automation ConferenceIEEE Computer Society Press, pp. 776–782 (2008).

[11] T. Kojima, N. Ando, Y. Matshushita, H. Okuhara, N. A. V. Doan and H. Amano: “Real chip evaluation of a low power CGRA with optimized application mapping”, Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable TechnologiesACM, p. 13 (2018).

[12] B. Mei, F.-J. Veredas and B. Masschelein: “Mapping an H. 264/AVC decoder onto the ADRES reconfigurable architecture”, Field Programmable Logic and Applications, 2005. International Conference onIEEE, pp. 622–625 (2005).

[13] D. Lee, M. Jo, K. Han and K. Choi: “FloRA: Coarse-grained reconfigurable architecture with floating-point operation capability”, Field-Programmable Technology, 2009. FPT 2009. International Conference onIEEE, pp. 376–379 (2009).

[14] S. Dave, M. Balasubramanian and A. Shrivastava: “RAMP: resource-aware mapping for CGRAs”, Proceedings of the 55th Annual Design Automation ConferenceACM, p. 127 (2018).

[15] S. Yin, X. Yao, D. Liu, L. Liu and S. Wei: “Memory-aware loop mapping on coarse-grained reconfigurable architectures”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **24**, 5, pp. 1895–1908 (2016).

[16] N. Ando, K. Masuyama, H. Okuhara and H. Amano: “Variable Pipeline Structure for Coarse Grained Reconfigurable Array CMA”, 2016 International Conference on Field-Programmable Technology, pp. 231–238 (2016).

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu: “A comprehensive survey on graph neural networks”, arXiv preprint arXiv:1901.00596 (2019).

[18] B. W. Kernighan and S. Lin: “An efficient heuristic procedure for partitioning graphs”, The Bell system technical journal, **49**, 2, pp. 291–307 (1970).

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama: “Optuna: A next-generation hyperparameter optimization framework”, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631 (2019).