jh200027-ISH

超巨大ニューラルネットワークのための分散深層学習フレームワークの開発とスケーラビリティの評価

田仲 正弘（情報通信研究機構）
田浦 健次朗（東京大学大学院情報理工学系研究科）
塙 敏博（東京大学情報基盤センター）

概要

我々は従来から、モデルパラレルによる大規模ニューラルネットワーク学習を行うミドルウェア RaNNC の開発を進めてきた。今年度は RaNNC の大幅な拡張を行い、GPU 利用率のためのパイプライン並列を導入すると共に、新たなニューラルネットワークの分割アルゴリズムを導入した。そこで、本課題を通じてチューニングや動作検証を実施した。研究代表者の所属組織の計算資源を用いた機能拡張を含めた総合的な改善の結果、モデルパラレルを用いるフレームワークである Megatron-LM と比較して、大規模化した BERT の事前学習において、約 5 倍のパラメータを持つネットワークを学習できること、両方が学習可能な規模のネットワークでは、ほぼ同等の学習速度であることを確認した。また大規模化した ResNet の学習において、同じくモデルパラレルを用いるフレームワークであるである GPipe と比較して、実験した全ての条件で、顕著に優れた学習速度が得られた。

1　共同研究に関する情報

1.1　共同研究を実施した拠点名

東京大学

1.2　共同研究分野

■超大規模情報システム関連研究分野

1.3　参加研究者の役割分担

- 田仲正弘 (研究代表者): 実施の統括、深層学習フレームワークの開発
- 田浦健次朗, 塙敏博: 並列計算の高速化

2　研究の目的と意義

近年、深層学習で用いられるニューラルネットワークの大規模化が進んでいる。例えば言語処理分野では、2018 年に Google から発表された BERT[1] を端緒として、事前学習を用いる極めて大規模なパラメータを持つニューラルネットワークが次々に発表されてきた。直近の例では、GPT-3 [2] と呼ばれるニューラルネットワークは、約 1750 億ものパラメータを持つ。

従来、深層学習の大規模化には，主にデータパラレルと呼ばれる並列化が用いられてきた。データパラレルでは、学習例のミニバッチを分割し、複数の GPGPU 上で並列に計算する。そのため、入力データやニューラルネットワークの逆伝播等のために保持されるデータは複数の GPGPU に分割配置できるが、ニューラルネットワークの学習パラメータは各 GPGPU

上に複製される。従って、ニューラルネットワークのパラメータが多く、GPGPU のメモリに収まらない場合には適用できない。

そのため近年、ニューラルネットワークを分割し、複数 GPGPU に配置するモデルパラレルと呼ばれる方式が注目されている。モデルパラレルを用いて大規模なネットワークを学習するフレームワークとして、Megatron-LM [3]、Mesh-TensorFlow [4]、GPipe [5] などがある。しかしこれらを適用するには、高い学習速度が得られるように、通信オーバヘッドなどを考慮しながら、ユーザが複雑なニューラルネットワーク定義の大半を書き換えてモデル分割を実装する必要があり、利用のハードルが高い。

そこで我々は、モデルパラレルのためのモデル分割を自動的に行うフレームワーク RaNNC (Rapid Neural Network Connector) を開発してきた。PyTorch で記述されたニューラルネットワークの定義を変更することなく、高い学習速度が得られるような分割を自動で決定するため、モデルパラレルによる学習を大幅に簡単化できる。本課題では、今年度に実施した RaNNC の機能拡張について、チューニングや動作検証を行った。本課題を通じて RaNNC の性能と堅牢性が向上することで、これまで学習が困難であった極めて大規模なパラメータを持つニューラルネットワークを、多くのユーザが容易に学習できるようになり、深層学習の研究が大きく加速されると期待される。

3 当拠点公募型研究として実施した意義

研究代表者らの所属する情報通信研究機構は、300 億ページ規模の Web コーパスと、AAAI, ACL 等に採択された深層学習技術（成果 [6, 7, 8, 9, 10] など）、及びそれらを用いた大規模 Web 情報分析システム WISDOM X[*1], 次世代音声対話システム WEKDA, 高齢者向けマルチモーダル音声対話システム MICSUS[*2] などの大規模自然言語処理アプリケーションを有している。一方、共同研究拠点となる東京大学情報基盤センターに所属する副代表者らは、並列分散処理による大規模計算における実績を持つ。このように、異なる専門性を持つ研究代表者・副代表者らによる共同研究の体制によって、相補的な研究の遂行が可能になる。

実施においては、巨大ニューラルネットワークの学習に研究代表者が所属する情報通信研究機構の持つ大規模コーパスを用いると共に、各種の学習結果を比較対象として利用した。また、共同研究拠点に所属する副代表者らが、大規模化・高速化の指針を定めるという形で研究を進めた。

JHPCN で提供される計算資源の他、研究代表者らの計算資源等も合わせて使用し、包括的な RaNNC の機能強化を実施した結果、研究代表者と、共同研究拠点に所属する共同研究者らによる共著論文が、IPDPS 2021 に採択された。また、RaNNC をオープンソースとして GitHub で公開し、研究代表者の所属する情報通信研究機構と、共同研究拠点である東京大学で、共同のプレスリリースを行った[*3][*4]。

4 前年度までに得られた研究成果の概要

前年度が終了した時点で、RaNNC を用い、モデルパラレル・データパラレルのハイブリッ

[*1] https://wisdom-nict.jp
[*2] https://www.youtube.com/watch?v=gCUrC3f9-Go
[*3] https://www.nict.go.jp/press/2021/03/31-2.html
[*4] https://www.u-tokyo.ac.jp/focus/ja/press/z0310_00002.html

ドによる基本的な学習が実現できていた。また、BERT のパラメータ数を 17 億（原論文の約 5 倍）に大規模化したニューラルネットワークを、GPU 数百枚を用いて学習し、正常に実行できることを確認していた。一方、ネットワークが大規模化し、モデルパラレルによる多数の部分ネットワークへの分割が必要になると、GPU 利用率が低下するという問題があった。

5　今年度の研究成果の詳細

本年度は、パイプライン並列 [11, 12] と、それを前提とする新たなネットワーク分割アルゴリズムを導入し、処理可能なニューラルネットワークの規模と学習速度を大幅に向上させた。これらの機能強化について、基本的なアルゴリズムの実装には主に研究者代表者の所属機関の計算資源を用い、様々な設定でのチューニングと動作検証のために、本課題に提供された Reedbush の計算資源を用いた。以降では、関連研究について紹介した後、本年度実施した RaNNC の機能拡張について、概要と性能評価について述べる。

5.1　関連研究

近年、モデルパラレルによる分割を行うフレームワークは、研究レベルのものを含めて複数提案されている（表 1）。これらの研究は、テンソルの分割を行うものと、ニューラルネットワークを計算グラフと見なし、グラフ分割を行うものの 2 種類に大きく分けられる。

テンソルの分割を行うフレームワークとして、Megatron-LM [3] と Mesh-TensorFlow [4] がよく知られている。しかしこれらのフレームワークを使用するには、通信オーバヘッドを軽減し効率よく並列計算が可能なように、ユーザが分割を自ら決定する必要がある。その際、分散計算のために提供される特殊な計算モジュールを使うように、既存のニューラルネットワーク定義を大幅に書き換える必要があり、利用のハードルは高い。

GPipe [5] はグラフ分割によるモデルパラレルを行う代表的なフレームワークであるが、分割はユーザが定義する必要がある。ニューラルネットワークの自動分割を目指したものとして、PipeDream [18], PipeDream-2BW [20] などがある。しかしこれらのフレームワークは、分割で得られた部分ネットワークの間で、パラメータ更新を非同期的に行うため、学習性能が低下することがある。この問題は、parameter staleness と呼ばれる。大規模ニューラルネットワークの学習において、数値安定性が重要なことが指摘されており [3, 22]、大規模ニューラルネットワークの学習には適さないと考えられる。また、ある程度自動で分割を行うものの、モデル定義は大幅に修正が必要である。

グラフ分割によるモデルパラレルの課題として、分割で得られた部分グラフに依存関係があるときは、前段の部分グラフの処理が終了するまで、後に続く部分グラフが実行できず、GPU の利用率が低下することが挙げられる。そのため、多くの既存フレームワークは、パイプライン並列と呼ばれる方法を併用する。パイプライン並列では、ミニバッチを更に細かいデータに分割し、ステージと呼ばれる部分グラフに入力して計算を行う。各ステージでは、ある分割データを処理し終えた後、次の分割データの処理を実行できるので、GPU 利用率を向上できる。パイプライン並列では、最も長く計算時間がかかるステージが全体のボトルネックとなる。そのため、全てのステージの処理時間を均等に近づけることによって、GPU 利用率を向上し、処理速度を改善できる。RaNNC はグラフ分割によるモデルパラレルを行うため、このパイプライン並列を使用する。

表 1　モデルパラレルによる学習を行うフレームワーク

	分割対象	ハイブリッド並列	手動/自動	Parameter staleness
Mesh-TensorFlow [4], Megatron-LM [3]	テンソル	あり	手動	なし
OptCNN [13], FlexFlow [14], Tofu [15]	テンソル	あり	自動	なし
GPipe [5]	グラフ	なし	手動	なし
AMPNet [16], XPipe [17]	グラフ	なし	手動	あり
PipeDream [18], SpecTrain [19]	グラフ	あり	自動	あり
PipeDream-2BW [20], HetPipe [21]	グラフ	あり	自動	あり
RaNNC (Ours)	グラフ	あり	自動	なし

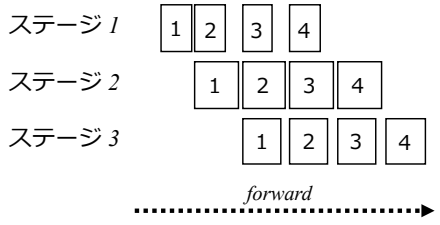

図 1　パイプライン並列（数字は分割データの対応を示す）

5.2　分割アルゴリズムの改善

RaNNC は PyTorch [23] で記述されたニューラルネットワークを計算グラフに変換し、計算時間・使用メモリ量などのプロファイリングに基づき、グラフ分割を決定する。分割で得られた部分ネットワークは、複数の GPU に配置され、並列に計算される。計算エンジンには、PyTorch を用いる。

ニューラルネットワークの分割は、図 2 に示す 3 つのステップで行う。ここでは、ニューラルネットワークに対応する計算グラフの部分グラフを subcomponent と呼び、始めに細粒度の subcomponent を特定してから、それらを組み合わせ、最終的にパイプライン並列におけるステージを構成する。なお、各ステップの詳細については、アルゴリズムの詳細については、7 章に挙げた研究成果に記載しており、ここでは概要のみ説明する。

最初のステップである atomic-level partitioning では、与えられた PyTorch のモデルを計算グラフに変更した後、入力されたミニバッチの内容によって結果が変化する処理を 1 つのみ含む subcomponent を特定する。計算グラフは、学習パラメータや定数に対する計算など、入力されるミニバッチによって変化しない処理を含むが、そうした処理のみからなる subcomponent をデータパラレルのために複製することは意味がない。以降のステップは、subcomponent を組み合わせて最終的な分割を決定するため、このステップで、データパラレルのために複製して意味のあるような、最小の単位を特定するのが目的である。

次に、block-level partitioning で、atomic subcomponent をグループ化し、より粗粒度な subcomponent を作る。subcomponent の組み合わせの探索によって、最終的にパイプライン並列のためのステージを決定するが、大規模モデルでは atomic subcomponent は多数得られるため、組み合わせの探索空間が非常に大きくなる。そこで、探索の前に subcomponent を組み合わせて数を減らすことにより、大

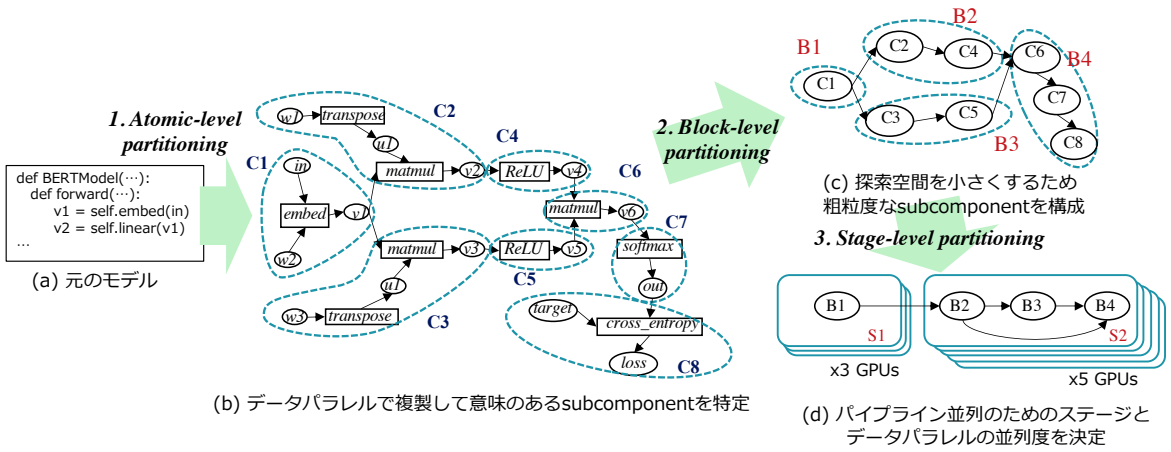

図 2　分割のステップ

幅に探索空間を削減できる。グループ化する atomic subcomponent を決める際は、異なる GPU に配置されたときの通信時間を削減するような組み合わせを優先的に選択する。

最後に、stage-level partitioning は、パイプライン並列のためのステージを決定するとともに、それぞれのステージについてのデータパラレルの並列度を決定する。このとき、パイプライン並列で高いスループットを達成するため、ステージの処理時間が均等になるように、先のステップで粗粒度化された subcomponent の組み合わせと、データパラレルの並列度を探索する。

5.3　実験結果

RaNNC のモデル規模に関するスケーラビリティと、訓練の実行速度を確認するため、大規模化ニューラルネットワークの訓練を試み、他のフレームワークとの比較実験を行った。なお、本実験は、必要リソース量の問題から、研究代表者の所属機関のクラスタで実行した。各設定において、NVIDIA V100 を 8 枚備えたサーバ 4 台を用いている。ノード間接続は、InfiniBand により 100Gbps で接続されている。

始めに、大規模 BERT の事前学習におけるスループットの比較した。BERT の隠れ層サイズは 2048 に設定し、レイヤ数を最大 256 まで拡張した。256 レイヤの設定において、パラメータ数は 129 億に達する（BERT の原論文では、隠れ層サイズは 1024, レイヤ数は 24 で、パラメータ数は 3.4 億）。バッチサイズは 256, 系列長 512, 数値精度は FP32 に設定した。

比較対象として、データパラレルのみの設定の他、GPipe 、Megatron-LM、PipeDream-2BW と比較した。ただし、GPipe は原論文ではモデルパラレルのみを行う手法が提案されているため、ここでは PipeDream-2BW の著者らによるデータパラレルと併用する実装（GPipe-Hybrid）を用いた。

図 3 に示すように、RaNNC は最大で Megatron-LM の 5 倍のモデルを訓練できた[*5]。また同一条件で訓練できた場合では、ほぼ同等のスループットを得た。また、GPipe と

[*5] 本稿作成時点では、Megatron-LM にパイプライン並列機能が実装されているが、本実験の実施時には未実装であったため、使用していない。

比較すると、全ての設定でより優れたスループットを得られた。PipeDream-2BW と比較すると、幾分スループットが低い結果となっているが、前述の通り、PipeDream-2BW は parameter staleness により学習精度が低下する可能性がある。

また、ResNet [24] を大規模化したネットワークの学習も行った。大規模化にあたって、ResNet のフィルタサイズを 8 倍としており、152 層の場合で 37 億パラメータになる。

BERT での比較に用いた Megatron-LM は、BERT 等の Transformer 系のネットワークにしか適用できない。また、データパラレル・モデルパラレルの併用が可能な GPipe（GPipe-Hybrid）と、PipeDream-2BW は、BERT のみに対応した実装であったため、使用できなかった。そこで比較対象として、GPipe を PyTorch で実装した torchgpipe [25]（GPipe-Model）を用いた。ただし、データパラレルと併用できないこと、複数ノードの利用ができないことから、モデルパラレルのみで 8 GPU での比較を行った。図 4 に示すように、データパラレルでは学習不可能な規模のネットワークにおいて、GPipe より顕著に高速に動作することを確認した。

6　今年度の進捗状況と今後の展望

実施計画において、実行効率の改善や多様なネットワークへの適用を挙げていたが、従来 17 億パラメータ程度を上限として動作していたところ、本報告書に記載した新たな分割アルゴリズムの導入により、評価実験では 129 億パラメータでの動作を確認している。また、パイプライン並列の導入により、GPU 利用率は最大で数倍程度向上している。対象となるネットワークについても、包括的な評価実験の対象とした BERT, ResNet 以外に、T5, 3D U-Net など、GPU メモリサイズの制限のため従来学習が困難であったネットワークの学習が可能なことを確認した。

RaNNC のソースコードについては、ドキュメントや大規模ニューラルネットワーク学習のサンプルコード等を整備した上で、GitHub に公開している。

今後の拡張として、Megatron-LM 等の、テンソルを分割する方式とのハイブリッド化により大規模ニューラルネットワークへの対応を計画している。また、BERT 以外の大規模化ネットワークについては、ある程度学習が進むことを確認する段階であるが、今後、主要な最新のネットワークについて大規模化による最終的な学習性能を確認していく予定である。

7　研究業績一覧（発表予定も含む）

学術論文 （査読あり）

なし

国際会議プロシーディングス　（査読あり）

- Masahiro Tanaka, Kenjiro Taura, Toshihiro Hanawa and Kentaro Torisawa, Automatic Graph Partitioning for Very Large-scale Deep Learning, In the Proceedings of 35th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2021), pp. 1004-1013, May, 2021.

国際会議発表　（査読なし）

なし

国内会議発表　（査読なし）

- 超大規模ニューラルネットワークのための自動並列化深層学習ミドルウェア RaNNC, 田仲 正弘, GTC 2021 (日本国内向けセッション), 2021 年 4 月 14 日.

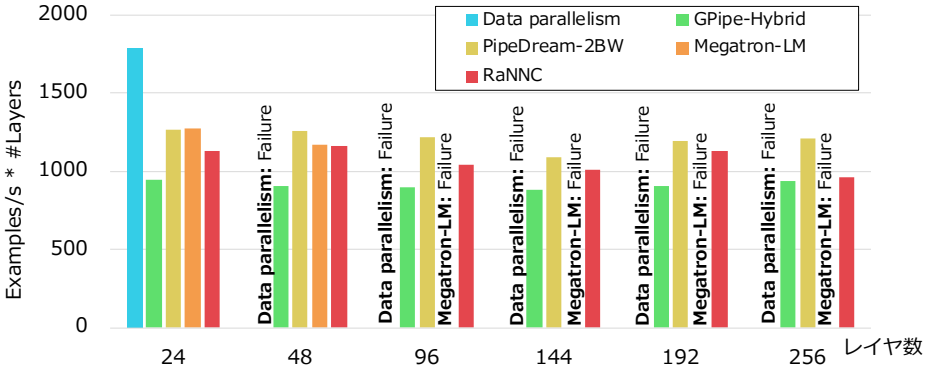

図 3　大規模 BERT 学習のスループット

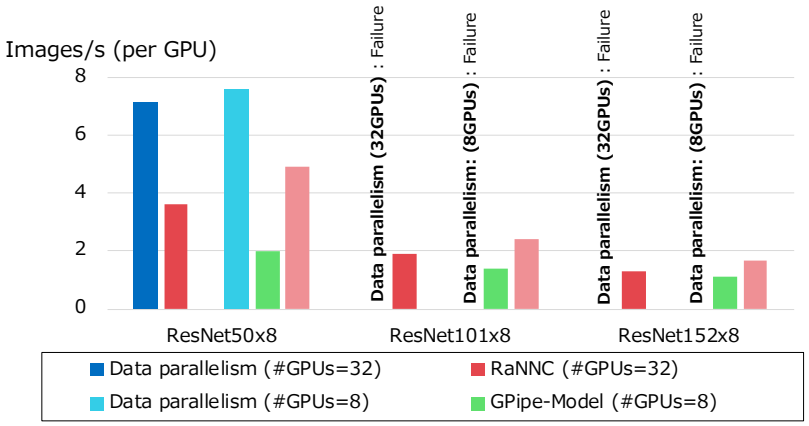

図 4　大規模 ResNet 学習のスループット

公開したライブラリ等

- RaNNC (Rapid Neural Network Connector) (`https://github.com/nict-wisdom/rannc`)

その他（特許，プレス発表，著書等）

- （プレス発表）自動並列化深層学習ミドルウェア RaNNC（ランク）をオープンソースで公開 ～超大規模ニューラルネットワークの学習が飛躍的に簡単に～, 国立研究開発法人情報通信研究機構, 国立大学法人東京大学, 2021 年 3 月 31 日 (`https://www.nict.go.jp/press/2021/03/31-2.html`, `https://www.u-tokyo.ac.jp/focus/ja/press/z0310_00002.html`)

参考文献

[1] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *NAACL-HLT 2019*, pp. 4171–4186 (2019).

[2] Brown, T. B., et al.: Language Models are Few-Shot Learners, *arXiv preprint, arXiv:2005.14165* (2020).

[3] Shoeybi, M., Patwary, M., Puri, R.,

LeGresley, P., Casper, J. and Catanzaro, B.: Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, *arXiv preprint, arXiv:1909.08053* (2019).

[4] Shazeer, N., Cheng, Y., Parmar, N. et al.: Mesh-TensorFlow: deep learning for supercomputers, *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, pp. 10435–10444 (2018).

[5] Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, M. X., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y. and Chen, Z.: GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, *arXiv preprint, arXiv:1811.06965* (2018).

[6] Kruengkrai, C., Torisawa, K., Hashimoto, C. et al.: Improving Event Causality Recognition with Multiple Background Knowledge Sources using Multi-Column Convolutional Neural Networks, *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 3466–3473 (2017).

[7] Kadowaki, K., Iida, R., Torisawa, K., Oh, J.-H. and Kloetzer, J.: Event Causality Recognition Exploiting Multiple Annotators' Judgments and Background Knowledge, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pp. 5820–5826 (2019).

[8] Ishida, R., Torisawa, K., Oh, J.-H., Iida, R., Kruengkrai, C. and Kloetzer, J.: Semi-Distantly Supervised Neural Model for Generating Compact Answers to Open-Domain Why Questions, *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, pp. 5803–5811 (2018).

[9] Iida, R., Kruengkrai, C., Ishida, R., Torisawa, K., Oh, J.-H. and Kloetzer, J.: Exploiting Background Knowledge in Compact Answer Generation for Why-questions, *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pp. 142–151 (2019).

[10] Oh, J.-H., Kadowaki, K., Kloetzer, J., Iida, R. and Torisawa, K.: Open Domain Why-Question Answering with Adversarial Learning to Encode Answer Texts, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 4227–4237 (2019).

[11] Giacomoni, J., Moseley, T. and Vachharajani, M.: FastForward for Efficient Pipeline Parallelism: A Cache-Optimized Concurrent Lock-Free Queue, *The 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 43–52 (2008).

[12] Gordon, M. I., Thies, W. and Amarasinghe, S.: Exploiting Coarse-Grained Task, Data, and Pipeline Parallelism in Stream Programs, *ACM SIGOPS Operating Systems Review*, Vol. 40, No. 5, pp. 151–162 (2006).

[13] Jia, Z., Lin, S., Qi, C. R. and Aiken,

A.: Exploring Hidden Dimensions in Parallelizing Convolutional Neural Networks, *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)* (2018).

[14] Jia, Z. and Zaharia, M.: Beyond Data and Model Parallelism for Deep Neural Networks, *The 2nd SysML Conference* (2018).

[15] Wang, M., Huang, C.-c. and Li, J.: Supporting Very Large Models using Automatic Dataflow Graph Partitioning, *The 14th EuroSys Conference 2019 (EuroSys '19)*, pp. 1–17 (2019).

[16] Gaunt, A. L., Johnson, M. A., Riechert, M. et al.: AMPNet: Asynchronous Model-Parallel Training for Dynamic Neural Networks, *arXiv preprint, arXiv:1705.09786* (2017).

[17] Guan, L., Yin, W., Li, D. and Lu, X.: XPipe: Efficient Pipeline Model Parallelism for Multi-GPU DNN Training, *arXiv preprint, arXiv:1911.04610* (2019).

[18] Harlap, A., Narayanan, D., Phanishayee, A., Seshadri, V., Devanur, N., Ganger, G. and Gibbons, P.: PipeDream: Fast and Efficient Pipeline Parallel DNN Training, *The 27th ACM Symposium on Operating Systems Principles*, p. 1–15 (2018).

[19] Chen, C.-C., Yang, C.-L. and Cheng, H.-Y.: Efficient and Robust Parallel DNN Training through Model Parallelism on Multi-GPU Platform, *arXiv preprint, arXiv:1809.02839* (2018).

[20] Narayanan, D., Phanishayee, A., Shi, K., Chen, X. and Zaharia, M.: Memory-Efficient Pipeline-Parallel DNN Training, *arXiv preprint, arXiv:2006.09503* (2020).

[21] Park, J. H., Yun, G., Yi, C. M., Nguyen, N. T., Lee, S., Choi, J., Noh, S. H. and Choi, Y.-r.: HetPipe: Enabling Large DNN Training on (Whimpy) Heterogeneous GPU Clusters through Integration of Pipelined Model Parallelism and Data Parallelism, *arXiv preprint, arXiv:2005.14038* (2020).

[22] Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N. and Chen, Z.: GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding, *arXiv preprint, arXiv:2006.16668* (2020).

[23] Paszke, A., Gross, S., Massa, F. et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, pp. 8024–8035 (2019).

[24] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pp. 770–778 (2016).

[25] Kim, C., Lee, H., Jeong, M., Baek, W., Yoon, B., Kim, I., Lim, S. and Kim, S.: torchgpipe: On-the-fly Pipeline Parallelism for Training Giant Models, *arXiv preprint, arXiv:2004.09910* (2020).