jh190023-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

近年、極めて多数のパラメータを持つ大規模なニューラルネットワークが高い学習性能を示す例が多く報告されている。深層学習の分散並列手法として主流であったデータパラレルは、ネットワークの全体を複製するため、そうした大規模ニューラルネットワークの学習には適用できない。そこで研究代表者らは、ニューラルネットワークを分割し複数の GPU に格納して分散計算を行う、モデルパラレルと呼ばれる分散学習を行うフレームワーク RaNNC を開発してきた。本課題では RaNNC を拡張し、モデルパラレルとデータパラレルのハイブリッドを実現した。従来研究と比較して、既存のニューラルネットワークを改変することなく適用できる点で優れている。RaNNC を用いて、最大 480 枚の GPU 上で BERT-Large を 5 倍以上の規模に拡大したネットワークを事前学習を行った結果、BERT-Large より顕著に優れた学習性能が得られることを確認した。また、従来研究と比較し、多くの条件でより高速に学習できることが示された。

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

東京大学情報基盤センター

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

本研究は、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言語処理分野における深層学習への適用、及びその有効性の検証を目的とする。言語処理を目的とした深層学習利用においては、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、高い学習性能を示している。直近の例では、T5[2] と呼ばれるニューラルネットワークが、各種タスクを全てテキスト変換として統一的に扱いながらも、主要なタスクで最高性能を更新しており、約 110 億ものパラメータを持つ。

しかしながら、一般的な GPU サーバ環境でこうした超巨大ニューラルネットワークを学習することは容易ではない。一般に、GPU を用

いた深層学習では、パラメータや逆伝播のために保持される計算結果を、GPU に保持する必要がある。しかし、現時点で一般的な GPU のデバイスメモリは最大 16GB (Google の TPU で 64GB) であり、パラメータ更新におけるバッファの保持に要するメモリ等を考慮すると、学習可能なニューラルネットワークの規模は、半精度浮動小数点数で計算を行ってもせいぜい数億パラメータ程度である。

既存の多くの深層学習フレームワークは、データパラレルと呼ばれる方式をサポートしており、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。

そこで申請者らは、モデルパラレルと呼ばれる並列処理方式によって、超巨大ネットワークを用いた深層学習を行うためのフレームワーク RaNNC (Rapid Neural Network Connector) を独自に開発してきた。モデルパラレルは、ニューラルネットワークを分割し、複数 GPGPU に配置するため、巨大ニューラルネットワークの学習に適する。従来、モデルパラレルのためのフレームワークは非常に少なかったが、BERT 等の大規模なパラメータを持つニューラルネットワークの成功から、最近になってモデルパラレルの重要性に注目が集まっている。代表的なものとして、Mesh-TensorFlow[3] や Megatron-LM[4] がある。ニューラルネットワークで計算されるテンソルを分割するもので、BERT 等に用いられる Transformer で特に有効に働く。しかし、これらのフレームワークを利用するには、ニューラルネットワークの構造や計算ハードウェアの特性をよく把握した上で、効率的な分散方法を実装する必要がある。既存のニューラルネットワークを大規模化する際も、多くの改修が必要で、依然としてモデルパラレル学習のハードルは高い。

一方、本研究で用いる、申請者らが独自に開発したフレームワークは、ほとんどのケースで、既存フレームワークのために実装されたニューラルネットワークの実装をそのまま利用でき、またモデルパラレルのための分割を自動で決定するなど、適用可能性や利便性において優れている。本課題を通じて申請者らが開発するフレームワークが完成することで、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 [5, 6, 7, 8, 9] など）、及びそれらを用いた対話システム WEKDA を含む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。その結果、従来は数枚程度の GPU で、ごく基本的なネットワークを学習できていた段階であったが、本課題の期間中に、数百枚規模の GPU で、BERT-Large を 5 倍以上のパラメータ数に拡大した巨大ネットワークの学習に成功した。また、従来の BERT-Large と比較して、実際に学習誤差が少ない結果を得られることが確認できた。

4 前年度までに得られた研究成果の概要

本課題は当年度に開始したものである。

5 今年度の研究成果の詳細

本課題では、課題代表者らが開発したモデルパラレルのための深層学習フレームワーク RaNNC の機能強化及び性能改善を行った。主な成果は以下の通りである。

- モデルパラレル・データパラレルのハイブリッド学習
- BERT を大規模化したニューラルネットワークの学習

以降でこれらの成果の詳細について述べる。

5.1 RaNNC の概要

本課題における成果に先立ち、使用したフレームワーク RaNNC の概要について説明する。本項目の多くは、課題開始前にすでに実現されていたが、本課題で得た成果の前提となるため、ここで説明する。

RaNNC のアーキテクチャを図 1 に示す。RaNNC は既存の深層学習フレームワークをバックエンドとして利用する。現時点で、RaNNC は PyTorch を利用するが、エンジンは交換可能な設計となっている。RaNNC は、バックエンドとなる深層学習フレームワークを用い、学習対象モデルの IR (intermediate representation) を出力する。IR は、ニューラルネットワークの表現として、事実上の標準となっている ONNX[*1]の形式を想定する。ONNX はニューラルネットワークを計算グラフとして表現するものであり、RaNNC はこの計算グラフを分割し、各 GPU に対応するプロセスに割り当てる。

ONNX によるグラフ表現では、ノードは計算オペレータと、その入出力となる値からなる。グラフ分割に当たっては、グラフのカットとなる値を複製し、分割によって生じる部分グラフの両方に値を持たせる。分散された部分グラフ群の計算に当たっては、カットにあたる値が得られると、対応する部分グラフに転送する。図 2 に例を示す。円形のノードは値を示し、矩形のノードは計算オペレータを示す。v_1, v_2 がカットとして選択され、グラフは二つの部分グラフに分けられる。これらの部分グラフは異なる GPU 上で計算されるが、一方のグラフで v_1, v_2 の値が計算された後、もう一方のグラフに転送される。

5.2 ハイブリッド並列

従来 RaNNC はモデルパラレルのみによる並列化に対応していた。しかし、大規模モデルの学習において、データパラレルを併用した高速化は必須である。そのため、本課題の期間において、モデルパラレル・データパラレルのハイブリッド並列を実現した。以下ではその課題と解決について説明する。

5.2.1 複製可能な部分グラフの特定

前述のように、ONNX によるグラフのノードは、値と計算オペレータからなる。しかしこうしたグラフを分割して得られるサブグラフ

[*1] https://onnx.ai/

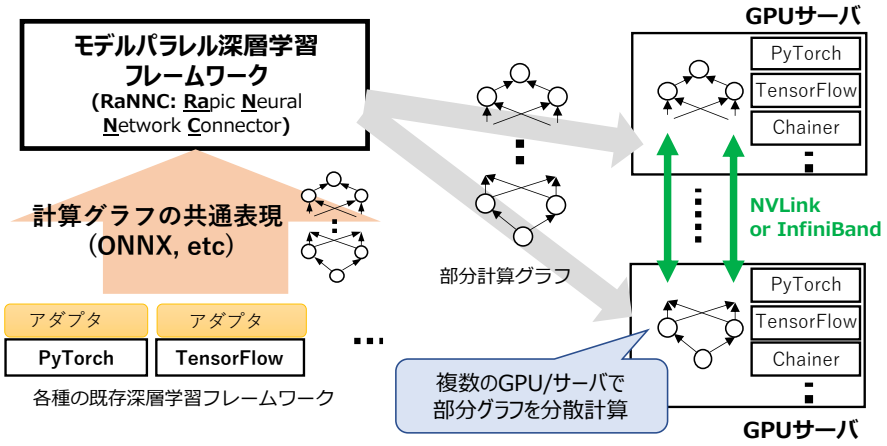

図 1　RaNNC の概要

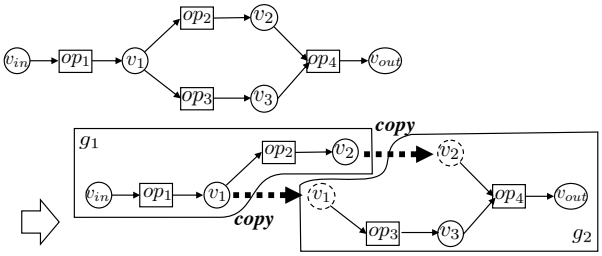

図 2　部分グラフの接続

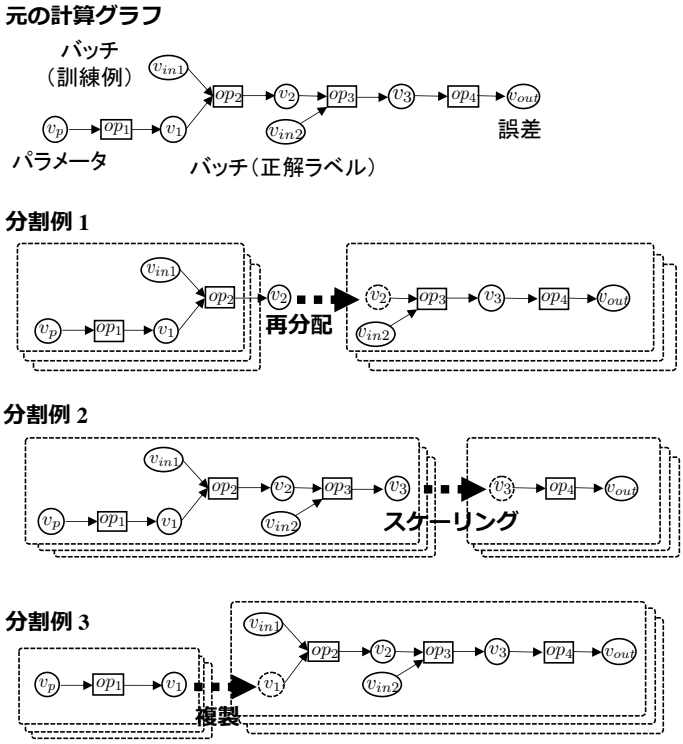

図 3　値の種類とグラフ分割

が、常にデータパラレルのために複製可能なわけではない。図 3 にグラフの例を示す。上部に示したものが、元の計算グラフである。入力として、バッチである v_{in1} と v_{in2}（一方は訓練例であり、もう一方が正解ラベルであると考える）、パラメータである v_p を取る。出力は v_{out} であり、誤差を意味するものとする。

この例が示すように、計算グラフには、バッチ、パラメータ、誤差などの異なる種類の持つ値が含まれる。図 3 は、異なる種類の値をカットとした分割を示している。分割例 1 は、バッチである v_2 で分割したものである。このとき、得られた部分グラフをデータパラレルで複製した場合、v_2 に含まれる訓練例の数を、両方のグラフの並列数に応じて再配分することで、適切に計算できる。分割例 2 では、誤差である v_3 で分割している。順伝播においては、送信元・送信先のグラフの並列数に応じたスケーリングを行うことで、適切な値を計算できる。し

かし、逆伝播にあたって、送信元グラフが処理した訓練例の集合に対応する誤差を得ることが出来ない。そのため、訓練時に誤差を表す値をカットとすることはできない。分割例 3 では、パラメータの値をカットとしている。しかし、送信元グラフは訓練例を処理するような内容を含まないため、こうしたグラフをデータパラレルによって並列化することは無意味である。

このように、分割によって得られる部分グラフをデータパラレルによって並列化可能なものとするには、カットとなる値はバッチに限られる。しかし、ONNX の計算グラフは、値の種類についての情報を含まない。そこで RaNNC では、入出力となる値についての簡単なルールに基づき、値の種類を推定する。例えば、ある演算オペレータの入力がバッチとパラメータで、出力が複数次元のテンソルである場合には、出力もバッチであると見なす。また、入力として 2 つのバッチを取り、スカラー値を出力する場合、出力は誤差であると見なす。元のモデルが持つ情報から、グラフへの入力がパラメータであるかどうか判別可能であり、またそれ以外の入力はバッチであると見なし、以降のグラフの値の種類について推論を繰り返すことで、分割のカットとして選択可能なバッチの値を発見する。

モデルパラレルのためのグラフの分割にあたっては、指定された分割数で、できるだけ部分グラフの必要メモリ量が等分になるように行う。部分グラフの必要メモリ量は、含まれる値のデータサイズに基づいて推定した。

5.3 高速化

本節では、モデルパラレル・データパラレルのハイブリッド並列のために、本研究で実施した高速化について述べる。

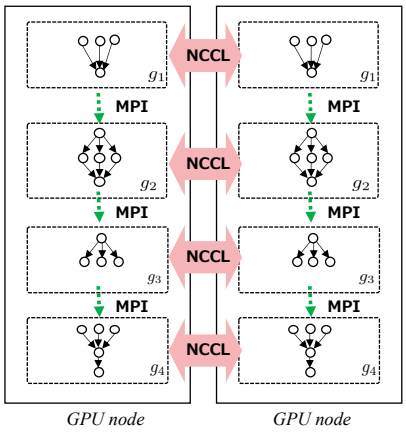

図 4　GPU 割り当て

5.3.1 GPU 割り当て

ハイブリッド並列では、モデルパラレルは多くの場合 8 程度が上限であるが、データパラレルは最大で数十以上の並列数が想定される。データパラレルにおける勾配の同期のための allreduce 通信を、高速なノード内通信のみで完結させることは出来ない。一方、モデルサイズや計算機環境によっては、モデルパラレルのためのサブグラフ間の通信は、GPU 内通信のみで実現できる。そこで、モデルパラレルの通信経路を、可能な限りノード内通信のみで完結させるように GPU の割り当てを行う（図 4）。なお、モデルパラレルのための、部分グラフ間の通信は、CUDA-aware MPI を用いた。また勾配の同期には、NCCL[*2]を用いている。

5.3.2 パイプライン並列

グラフ分割に基づくモデルパラレルのアプローチでは、グラフ間の依存性のため、通常は複数の部分グラフを同時に計算できず、GPU の利用率が低くなることが大きな課題である。そこで、RaNNC では、バッチをさらに細かく分割した訓練例の集合（マイクロバッチ）を順次与えることで、GPU 利用率を改善するパイ

[*2] https://developer.nvidia.com/nccl

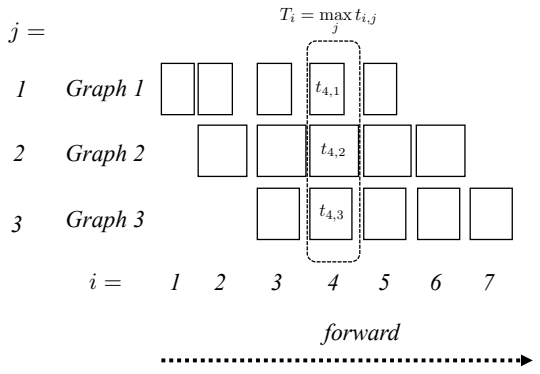

図 5　パイプライン並列

プライン並列を用いる。図 5 に例を示す。ここでは、モデルパラレルのための分割で得られた 3 つのグラフに、マイクロバッチ数を 5 としている。ステップ i におけるグラフ j の計算時間を $t_{i,j}$ とするとき、ステップ i でのグラフ全体の計算時間 T_i は $max_j t_{i,j}$ となる。

パイプライン並列は、分割数を大きくするほど、GPU 利用率を改善できる。しかし、マイクロバッチの合計に相当するだけのメモリを消費するため、分割数には限度がある。そこで、Gradient checkpointing[10, 11] と呼ばれる技術を実装している。Gradient checkpointing は、順伝播では activation のためのデータを保持せず、逆伝播時に改めて順伝播の計算を行うものである。計算量は増加するが、パイプライン並列と併用することで、マイクロバッチ 1 件分のメモリしか消費せずに、多数のマイクロバッチを計算できる。

なお、これらのパイプライン並列や Gradient checkpointing は、gPipe[12] 等の既存研究でも採用されている。

5.3.3　勾配の集約

RaNNC では勾配の同期に NCCL の allreduce を用いているが、多数の小さなサイズのデータに対して allreduce を呼び出すと効率が低下する。そのため、RaNNC では、メモリ上の連続領域に勾配を確保するようにしている。同様の技術は Horovod[*3]でも利用されている。

5.4　実験

本課題を通じて開発した RaNNC のハイブリッド並列の有効性を検証するため、BERT を例題とした実験を行った。本節で示す実験では、RaNNC のバックエンドのエンジンとして、PyTorch 1.3.1 を用いた。通信に OpenMPI 4.0.0 及び NCCL 2.5.6 を用いた。

本課題のために付与された Reedbush のトークンは、チューニングのためのテスト実行及びハイパーパラメータの探索に用いたため、本節の実験では研究代表者の所属機関の計算クラスタを用いた。計算ノードは、Xeon Gold 6140 (2.30GHz) を 2 基, メモリ 768GB, 及び NVIDIA V100（メモリ 32GB）を 8 基を備え、ノード間通信は InfiniBand 100Gbps である。

5.4.1　学習性能

RaNNC の計算結果の妥当性及び大規模ニューラルネットワークの有効性を検証するため、BERT-Large を大規模化したネットワーク（以降、BERT-Tall）の事前学習を行った。学習データには、情報通信研究機構の持つ Web コーパス 22 億文を用いた。また、大規模化のために変更したハイパーパラメータは表 1 の通りである。パラメータ数は約 18.5 億であり、オリジナルの BERT-Large（3.3 億）の 5 倍以上の規模となる。

数値精度は FP16 としたが、Megatron-LM での設定にならい、layer normalization, embedding, 及び softmax を FP32 としている。また、独自に attention 構造の key における全結合層を FP32 とした。バッチサイズは 4096 とし、オプティマイザには LAMB を用いた（学習率 10^{-4}）。また、語彙数は 10 万である。そ

[*3] https://github.com/horovod/horovod

表 1　大規模化 BERT のハイパーパラメータ

	層数	隠れ層サイズ	ヘッド数
BERT-Large（参考）	24	1024	16
BERT-Tall	54	1920	20

の他の設定は、原論文 [1] と同じである。

学習の実行プログラムには、PyTorch での BERT の fine-tuning 用コード[*4]に、事前学習を想定した大規模データセットのためのメモリ効率化や、日本語対応などの改修を適用したコードを用いた。コードは訓練実行のための処理フローと、ニューラルネットワークの定義部に分かれているが、RaNNC の適用に当たって前者のみを改修し、後者の改変は行っていない。

系列長を 128 とした学習（原論文における phase 1）の 12 万ステップまで実行した際の誤差の推移を図 6 に示す。比較のため、BERT-Large での同等の学習での誤差の推移も示している。図に示すように、BERT-Large よりも低い誤差が得られることが確認できた。

この実験は GPU 枚数を 64 から 480 枚の間で変化させながら実施した。図 6 に示した結果は、64 枚で実行した際に得たものであるが、確認する限りにおいて、枚数にかかわらずほぼ同等の学習誤差が得られた。

5.5　速度

大規模ニューラルネットワーク学習における速度を評価するため、BERT を例として、Megatron-LM[4] と速度を比較した。Megatron-LM は、BERT を構成する multi-head attention と呼ばれる構造において、効率の良いモデルパラレルを実現するフレームワークである。BERT-Large の層数と、モデルパラレルの分割数を変化させた際の、訓練例の秒間の処理数を図 7 を示す。RaNNC については、パイプライン並列のマイクロバッチ数 M を 2 または 32 とした処理数を調べた。プロセス数は 8 とし、バッチサイズは 1 プロセスあたり 8（合計 64）とした。また RaNNC におけるパイプライン並列で、$M = 32$ の場合、checkpointing を用いた上で、マイクロバッチあたりのバッチサイズを 8 としている。

図に示すように、モデルパラレルの並列数が 2 ないしは 4 において RaNNC が優位であった。モデルパラレルの分割数が 8 の際、RaNNC では GPU の使用率の低下が顕著であり、Megatron-LM が優位という結果が得られた。

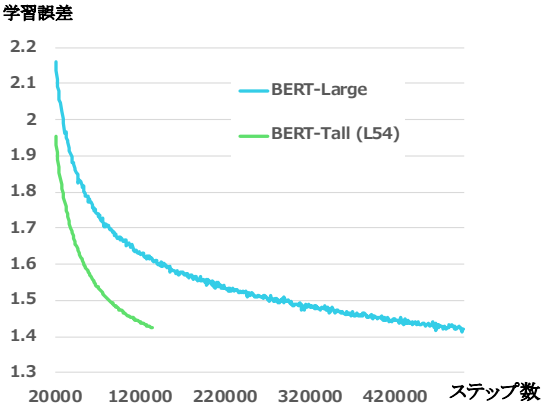

図 6　学習誤差の推移

[*4] https://github.com/huggingface/transformers

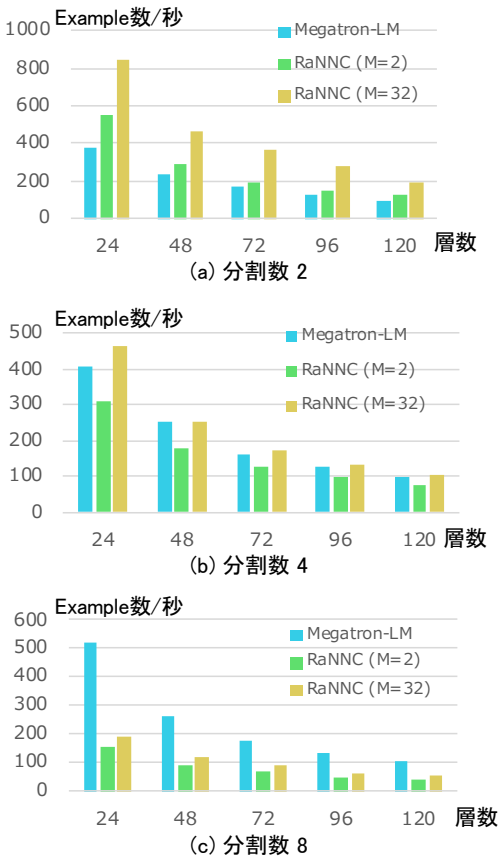

図 7　BERT におけるスループット

6　今年度の進捗状況と今後の展望

モデルパラレル・データパラレルのハイブリッド並列を可能とし、BERT-Large の 5 倍以上の規模のネットワークで最大 480 枚の GPU を用いた分散学習が実現できた点は、大きな進捗があったと言える。また速度面においても、モデルパラレルのために大幅な改変を要する既存研究と比較して、ニューラルネットワークの定義を改変することなく、実際的な条件においてより高速な学習が実現できたことも、顕著な成果と考えられる。

一方、RaNNC の一般公開を予定していたが、大規模化 BERT における数値精度と収束性の問題などの解決に注力したため、多様なネットワークでの検証に課題を残し、実現には至らなかった。モデルパラレル・データパラレルのハイブリッド並列のフレームワークとして、基本的な機能はすでに実現されたものと考えられるので、すでに採択されている次年度の課題期間において、安定性の改善と、ソフトウェアの一般公開、論文投稿を進めていく。

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）

なし

国際会議発表　（査読なし）

なし

国内会議発表　（査読なし）

なし

その他（特許，プレス発表，著書等）

なし

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, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pp. 4171–4186 (2019).

[2] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P. J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (2019).

[3] 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).

[4] Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J. and Catanzaro, B.: Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism (2019).

[5] 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).

[6] 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).

[7] 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).

[8] 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).

[9] 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).

[10] Griewank, A. and Walther, A.: Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation, *ACM Transactions on Mathematical Software*, Vol. 26, No. 1, pp. 19–45 (2000).

[11] Chen, T., Xu, B., Zhang, C. and Guestrin, C.: Training Deep Nets with Sublinear Memory Cost (2016).

[12] Harlap, A., Narayanan, D., Phanishayee, A., Seshadri, V., Devanur, N., Ganger, G. and Gibbons, P.: PipeDream: Fast and Efficient Pipeline Parallel DNN Training (2018).