Vol.2017-SLP-119 No.18
2017/12/22

テンソルトレイン分解による End-to-End 自動音声認識モデルの圧縮

森 巧磨[1,a)]　Andros Tjandra[1]　Sakriani Sakti[1]　中村 哲[1]

概要：大規模な音声コーパスが作成され，自動音声認識（ASR）を学習する統計学習法として大規模なニューラルネットワークで使用されるようになった．また，簡単で強力な方法として，殆どのモジュールを単一のモデルとして学習する End-to-End 自動音声認識の研究が行われている．ニューラルネットワークを用いた音声認識は，たくさんのパラメータを有し，新しいデータの学習および予測のために多くの計算資源を必要とする．パラメータが増える原因の一つとして，時系列タスクをモデリングし，様々な複雑な問題を解析する手法としてリカレントニューラルネットワーク（RNN）を使うことがあげられる．音声認識は音声要約，自動コールセンター，音声翻訳などの多くのアプリケーションの重要なコンポーネントである．したがって，多くの場面でこのモデルを使えるようにするためには，メモリを削減して，軽量なモデルとする必要がある．本稿では，リカレントネットワークの中間層をテンソル表現し，それらを効率的に分解するテンソルトレイン分解により，パラメータ数を大幅に削減する代替 RNN モデルを提案する．我々は，Libri Speech において非圧縮ゲーティッドリカレントユニット（GRU）モデルとテンソルトレイン分解により圧縮した GRU モデルを比較評価し，パラメータの数を大幅に削減しながら性能を維持することを示した．

キーワード：テンソルトレイン分解，モデル圧縮，End-to-End モデル，音声認識

1. はじめに

大規模な音声コーパスが作成され，自動音声認識（ASR）を学習する統計学習法として大規模なニューラルネットワークが使用されるようになった [1]．また，簡単で強力な方法として，殆どのモジュールを単一のモデルとして学習する End-to-End 自動音声認識の研究が行われている [2], [3], [4]．これらの高性能モデルは，入力音声を認識するために高性能なサーバを必要とする．音声認識は様々な使用環境があり，モバイルデータ接続が利用できない場合や，通信が遅く，信頼性が低い場合でも，ユーザーのやりとりを可能にしながら，待ち時間を短縮する必要がある．これらの主な問題は，デバイスのディスク，メモリ，および計算量の制約である．ニューラルネットワークの動作回数はモデルパラメータの数に比例するため，モデルを圧縮することは，ディスク使用量とメモリ消費量を削減する観点から不可欠である．本稿では，テンソルトレイン分解を用いて，できるだけ小さなモデルから性能を導出することを検討する．テンソルトレイン分解とは，高次元テンソルを低次元テンソルに圧縮する手法であり，空間/時間計算量ともに次元数を指数に持たないため，高次元テンソルであっても高速に計算することができる．本稿では，End-to-End 自動音声認識の GRU 層の重みの圧縮に用いる．本稿では，テンソルトレイン分解 End-to-End 自動音声認識に焦点を当てているが，この提案は，手書き認識や機械翻訳などの他の分野の難しいタスクにも適用できる．第 2 章では，ニューラルネットワークを圧縮するための関連研究を紹介する．第 3 章では，テンソルトレイン分解による GRU の圧縮の説明をする．提案した方法の有効性は第 4 章と第 5 章で検証する．最後に，第 6 章にて結果をまとめる．

2. 先行研究

ニューラルネットワークのモデル圧縮に関する研究はいくつかのアプローチで行われている．先行研究では，ニューラルネットワークパラメータに重大な冗長性があることが示されている．例えば，Daniel ら [5] は，ニューラルネットワークがいくつかのパラメータの値を考慮して再構成できることを示している．Wang ら [6] および Rohit ら [7] は

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] mori.takuma.mi1@is.naist.jp

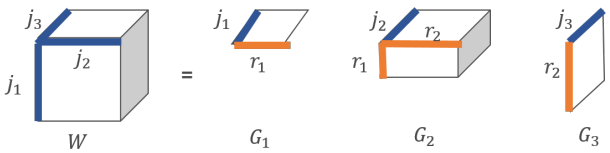


図 1 テンソルトレインフォーマットによる低階テンソル表現

音響モデルを圧縮するために特異値分解（SVD）とベクトル量子化の組み合わせた． Hinton ら [8] は，統計的な温度で複数のモデルの出力の加重平均を調整しながら，アンサンブル学習を単一のモデルに圧縮する方法を提案した．Hinton らはこのような複数のモデルや高性能なモデルの出力分布を学習のターゲットとし，単一の小さなモデルを学習させ，高いパフォーマンスを得ることを知識蒸留と呼んでいる [8]．また，Bulo ら [9] は，多くのドロップアウトパターンを作り，異なる問題に対して強いモデルを作り，これを知識蒸留し，単一の高性能なモデルを作った．機械翻訳の分野では，Kim ら [10] は時系列学習のための Sequence-to-Sequence 注意モデルの知識蒸留を提案し，教師ネットワークの系列出力分布や出力文字列を学習する方法などを提案した．また，Oseledets ら [11] は次元の呪いを回避する手法として，高ランクテンソル行列を低ランクテンソルに近似する手法であるテンソルトレイン分解を提案し，Andros ら [12] はテンソルトレイン分解が音楽分析のために RNN や GRU の重み行列を圧縮できることを示した．我々は，テンソルトレイン分解が CTC ベースの End-to-End 自動音声認識という難しい系列タスクにも用いることができることを示す．

3. テンソルトレイン分解による GRU の圧縮

本章では，テンソルトレイン分解による GRU の重み行列の圧縮 [12] について解説する．$k \in \{1, .., d\}$ と，k 次元の次元インデックス $j_k \in \{1, .., n_k\}$ の値を用いて表せる，d 次元配列（テンソル）を $\mathcal{W}$ とする．$\mathcal{W}$ の要素が以下の式で行列 $G_k[j_k]$ によって表せるときテンソルトレインフォーマットと呼び，以下の式で定式化される [11]．

$$\mathcal{W}(j_1, j_2, .., j_d) = G_1[j_1] \cdot G_2[j_2] \cdot .. \cdot G_d[j_d]. \qquad (1)$$

例えば，$d = 2$ のときのテンソルフォーマット適用を図示すると，図 1 のように表すことができ，低ランク近似するときに必要となる $r_k \in \{1, .., n_k\}$ がテンソルトレインランクである．

GRU アーキテクチャでは，入力から隠れ層，隠れ層から出力が 1 つずつ，リセットゲートを制御する重みが入力と出力で 2 つ，更新ゲートを制御する重みが入力と出力で 2 つの合計 6 つの重み行列で表すことができる．以下に，入力から隠れ層，隠れ層から出力の重みの圧縮について説明するが，他の 4 つの重みも同様の手法で圧縮することができる．

入力から隠れ層の重みを行列 $W_{xh} \in \mathbb{R}^{M \times M}$ と定義し，隠れ層から出力の重みを行列 $W_{hh} \in \mathbb{R}^{M \times M}$ と定義する．今回，GRU に対する入力次元と出力次元が同じであるため，どちらの重みも $\mathbb{R}^{M \times M}$ に帰属する．本手法では，この (W_{xh}, W_{hh}) を圧縮する．

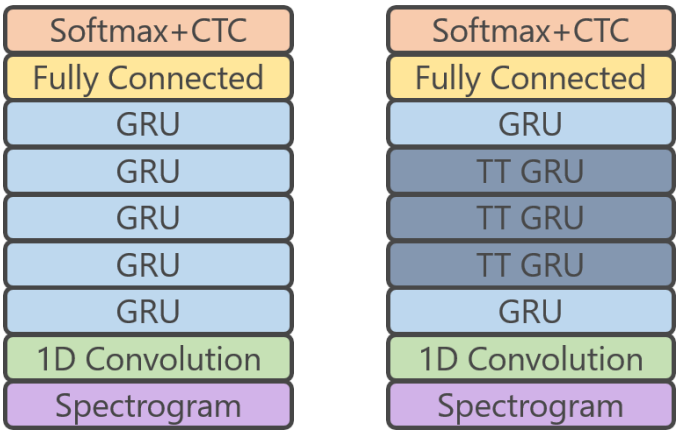
まず，行列の形状 M を $\prod_{k=1}^{d} m_k$ に分解する． 次に，モデルのテンソルトレインランク $\{r_k\}_{k=0}^{d}$ を決定し，W_{xh} をテンソル $\mathcal{W}_{xh}$ と W_{hh} をテンソル $\mathcal{W}_{hh}$ と置き換える．テンソル $\mathcal{W}_{xh}$ は，$\forall k \in \{1, .., d\}$，$\mathcal{G}_k^{xh} \in \mathbb{R}^{m_k \times m_k \times r_{k-1} \times r_k}$ である時，テンソルトレインコア $\{\mathcal{G}_k^{xh}\}_{k=1}^{d}$ で表すことができ，テンソル $\mathcal{W}_{hh}$ は，$\forall k \in \{1, .., d\}$，$\mathcal{G}_k^{hh} \in \mathbb{R}^{m_k \times m_k \times r_{k-1} \times r_k}$ である時，テンソルトレインコア $\{\mathcal{G}_k^{hh}\}_{k=1}^{d}$ で表すことができる（図 1）．また，W_{xh} および W_{hh} のテンソルコアによって定義される行 $p \in \{1, .., M\}$ を全単射する関数 $\mathbf{f}_i^x$ および $\mathbf{f}_i^h$ を定義する．これらを用いて，RNN の出力 h_t は以下のように書き表せる．

$$\begin{aligned}
a_t^{xh}(p) &= \sum_{j_1, .., j_d} \mathcal{W}_{xh}(\mathbf{f}_i^x(p), [j_1, .., j_d]) \cdot \mathcal{X}_t\,(j_1, .., j_d) \\
a_t^{hh}(p) &= \sum_{j_1, .., j_d} \mathcal{W}_{hh}(\mathbf{f}_i^h(p), [j_1, .., j_d]) \cdot \mathcal{H}_{t-1}\,(j_1, .., j_d) \\
a_t^{xh} &= \left[a_t^{xh}(1), .., a_t^{xh}(M)\right] \\
a_t^{hh} &= \left[a_t^{hh}(1), .., a_t^{hh}(M)\right] \\
h_t &= f(a_t^{xh} + a_t^{hh} + b_h),
\end{aligned}$$

$\mathcal{X}$ は入力 x_t のテンソル表現であり，$\mathcal{H}_{t-1}$ は以前の隠れ状態 h_{t-1} のテンソル表現である．以上が GRU の入力から隠れ層，隠れ層から出力の重みの圧縮であり，GRU で圧縮するには，リセットゲートと更新ゲート内の行列を同様の方法で圧縮することで達成できる．これをテンソルトレイン GRU（TTGRU）とする．

4. 実験

この章では，TTGRU を評価し，ベースラインである非圧縮 GRU と比較する．我々は，このモデルを評価するタスクとして，英語音声コーパスである LibriSpeech コーパス [10] を使用した．この実験では，LibriSpeech コーパスのいくつかのサブセットが使用され，使用されるコーパスは表 1 に示されている．トレーニングデータに train-clean-1000 サブセット，検証データに dev-clean サブセット，テストデータにの test-clean サブセットを用いた．この音声認識タスクでは，16kHz でサンプリングされた発話が入力として与えられ，正しいラベルである文字列誤り率で評価される．TTGRU のモデルパラメータを最適化するために Adam アルゴリズム，ベースラインの GRU には SGD アルゴリズムを使用した．以降，GRU は GRU-HF と表



（a）GRU モデル　　（b）TTGRU モデル

図 2　モデル設計

し，TTGRU は TTGRU-HF-R で表す．ここで，F は隠れユニットの数であり．R はテンソルトレインランクを表す．例えば，TTGRU-H4x8x6x8-R3 は，テンソルトレインフォーマットのテンソルトレインランク 3 で 4x8x6x8 の隠れユニットを有する TTGRU を意味する．GRU のパラメータの設定は Deep Speech2[4] の論文に基づいており，TTGRU のパラメータの設定は検証セットのパフォーマンスに基づいて決定され，図 2 のような設計となっている．

結果を次の章に示す．

表 1　用いたデータセットの概要

サブセット	時間	話者数	発話時間/人 (分)
train-clean-100	100.6	251	25
dev-clean	5h	40	10
test-clean	5h	40	10

5. 実験結果

表 2　train-clean-100 による学習の結果

Model	Param	Comp	Val CER	Test CER
GRU-H1510	13M	100	20.03%	20.62%
TTGRU				
H4x8x6x8-R3	11k	0.08	27.57%	27.21%
H4x8x6x8-R5	22k	0.16	23.76%	23.40%
H4x8x6x8-R7	37k	0.27	22.68%	23.73%

実験の結果と GRU のパラメータがどの程度圧縮されているかを表 2 に示した．TTGRU はランク 5 やランク 7 では圧縮される前のモデルに近い性能を引き出すことに成功した．ベースラインモデルは，1510 の隠れユニットを持つ GRU である．我々の提案するモデルは，4x8x6x8 の出力形状とテンソルトレインランク（3,5,7）の TTGRU である．音声入力はスペクトログラムに変換して正規化したもたのである．評価関数には単語誤り率 (CER) を用いた．表 2 は, ベースラインおよび提案されたモデルに関する我々の実験の結果である．表は，これらのモデルが検証セットとテストセットが同様の性能を有することを示している．我々の提案したモデルは，変換した GRU 層に限れば約 99%削減し，モデル全体では約 60%削減しているが，パラメータの数を減らすと同時に性能を維持することができた．

6. まとめ

本稿では，CTC ベースの End-To-End 自動音声認識という難しいタスクでテンソルトレイン分解が有効であることを示した．テンソルトレインフォーマットを使用して，複数の低ランクテンソルをによって GRU レイヤー内の重み行列を表現した．TTGRU End-to-End 自動音声認識は，モデルの性能と精度を維持しながらパラメータの数を大幅に圧縮した．我々は，読み上げコーパスでこれを評価し，難しいタスクでも精度を失うことなく元のモデルと比較して RNN パラメータを大幅に削減できた．

謝辞　本研究の一部は JSPS 科研費 JP17H06101 および JP17K00237 の助成を受けたものである．

参考文献

[1] Deng, L., Yu, D. et al.: Deep learning: methods and applications, *Foundations and Trends® in Signal Processing*, Vol. 7, No. 3–4, pp. 197–387 (2014).
[2] Graves, A. and Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772 (2014).
[3] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. et al.: Deep speech: Scaling up end-to-end speech recognition, *arXiv preprint arXiv:1412.5567* (2014).
[4] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G. et al.: Deep speech 2: End-to-end speech recognition in english and mandarin, *International Conference on Machine Learning*, pp. 173–182 (2016).
[5] Denil, M., Shakibi, B., Dinh, L., de Freitas, N. et al.: Predicting parameters in deep learning, *Advances in Neural Information Processing Systems*, pp. 2148–2156 (2013).
[6] Wang, Y., Li, J. and Gong, Y.: Small-footprint high-performance deep neural network-based speech recognition using split-VQ, *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, IEEE, pp. 4984–4988 (2015).
[7] Prabhavalkar, R., Alsharif, O., Bruguier, A. and McGraw, L.: On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition, *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, IEEE, pp. 5970–5974 (2016).
[8] Hinton, G., Vinyals, O. and Dean, J.: Distilling

the knowledge in a neural network, *arXiv preprint arXiv:1503.02531* (2015).

[9] Bulò, S. R., Porzi, L. and Kontschieder, P.: Dropout distillation, *International Conference on Machine Learning*, pp. 99–107 (2016).

[10] Kim, Y. and Rush, A. M.: Sequence-Level Knowledge Distillation, (online), available from ⟨http://arxiv.org/abs/1606.07947⟩ (2016).

[11] Oseledets, I. V.: Tensor-train decomposition, *SIAM Journal on Scientific Computing*, Vol. 33, No. 5, pp. 2295–2317 (2011).

[12] Tjandra, A., Sakti, S. and Nakamura, S.: Compressing Recurrent Neural Network with Tensor Train, *arXiv preprint arXiv:1705.08052* (2017).