整数基底分解法と量子化による音響モデルの圧縮*

○太刀岡 勇気、△安倍 満 (デンソーアイティーラボラトリ)

1　はじめに

音声認識において、深層神経回路網 (Deep Neural Networks; DNN) に基づく音響モデルは、従来のガウス混合分布に基づくモデルの性能を多くの場合上回るが、モデルサイズも大きくなる。これによる必要メモリ量の増加が、特に組み込み向け用途では、問題になることが多い。Xue らは、特異値分解 (Singular Value Decomposition; SVD) を DNN に適用し、パラメータ数を削減する方法を提案した [1]。また [2] では、騒音下音声認識における有効性も確認されている。本報ではさらなるメモリ量の削減を目的として、SVD 後の重み行列の量子化を行う手法 (3 節) を検討する。

一方、画像認識の分野では、重み行列を $\{-1, 0, 1\}$ の 3 値行列と実数行列の積に分解することで、必要なメモリ量・計算量を削減する「整数分解に基づく内積高速化法 (Scalar Product Accelerator by Integer Decomposition; SPADE)」が知られている [3]。本報では、この手法 (4 節) の音声認識における有効性を検証するとともに、両パラメータ削減法の効果を、必要メモリ量・計算量と性能の観点から比較する。

2　SVD による音響モデルの圧縮

DNN の l 層目 1 層を考える。このとき、$l-1$ 層からの入力を $\boldsymbol{x}_I^l$、出力を $\boldsymbol{x}_J^{l+1}$ とする。以降、上添字で層インデクス、下添字で次元を表す。重み行列 $\boldsymbol{A}_{J\times I}^l$ と、入力ベクトル $\boldsymbol{x}_I^l$ の積を取った後に、非線形関数 f を適用することで、以下のように出力ベクトル $\boldsymbol{x}_J^{l+1}$ を得る。

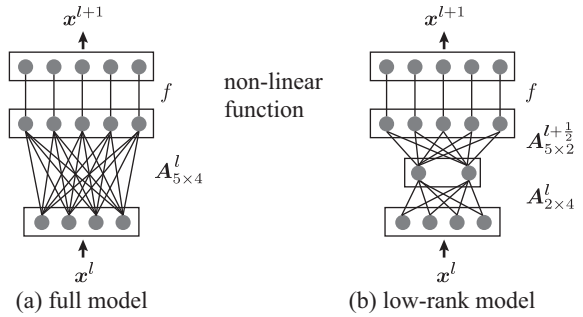
$$\boldsymbol{x}_J^{l+1} = f(\boldsymbol{A}_{J\times I}^l \boldsymbol{x}_I^l + \boldsymbol{b}_J^l) \tag{1}$$

ここで、$\boldsymbol{b}_J^l$ はバイアス項である。重み行列 $\boldsymbol{A}_{J\times I}^l$ を、以下のように SVD で分解する [1]。

$$\boldsymbol{A}_{J\times I}^l = \boldsymbol{U}_{J\times I} \left[\mathrm{diag}\left(\sigma_1, \ldots, \sigma_I\right)\right] \boldsymbol{V}_{I\times I} \tag{2}$$

σ は特異値 $(\sigma_1 \geq \ldots \geq \sigma_I)$ である。行列 $\boldsymbol{U}$ と $\boldsymbol{V}$ は、正規直交化された列ベクトルを持つ。diag() は引数を対角要素に持つ行列である。K 番目 $(K < \frac{IJ}{I+J})$ までの特異値 $\sigma_k (k = \{1, \ldots, K\})$ と、それに対応する特異値ベクトルにより $\boldsymbol{A}^l$ を低ランク近似する。

$$\begin{aligned} \boldsymbol{A}_{J\times I}^l &\approx \boldsymbol{U}_{J\times K} \left[\mathrm{diag}\left(\sigma_1, \ldots, \sigma_K\right)\right] \boldsymbol{V}_{K\times I} \\ &= \boldsymbol{A}_{J\times K}^{l+\frac{1}{2}} \boldsymbol{A}_{K\times I}^l \end{aligned} \tag{3}$$



(a) full model　(b) low-rank model

Fig. 1　DNN compression via low-rank factorization, from (a) $5 \times 4 = 20$ to (b) $5 \times 2 + 2 \times 4 = 18$.

図 1(b) のように、$\boldsymbol{x}^l$ に対して $\boldsymbol{A}^l$ を乗じたのちに、$\boldsymbol{A}^{l+\frac{1}{2}}$ を乗ずることで、図 1(a) に比べて、全体の計算量・メモリ量を削減することができる。この方法では、ネットワークの内部で、各層が 2 層に分かれるだけなので、デコード時の実装を変える必要がないという利点もある。

3　重み行列の量子化

量子化のアルゴリズムは SVD に限らないので、記号を変えて説明する。$\boldsymbol{A} \approx \boldsymbol{W}' \cdot \boldsymbol{W}$ のように分解できたとすると、分解後の重み行列 $\boldsymbol{W}$ と $\boldsymbol{W}'$ のパラメータを量子化によりメモリ量を削減できる。また固定小数点化すれば、計算量も削減できる。まず行列 $\boldsymbol{W}$ の要素の最大の絶対値 $W^{\max} = |\boldsymbol{W}|_{\max}$ を算出し、$-W^{\max}$ と $W^{\max}$ の間を D 量子化する。その際に、以下の SPADE と同様に、0 を入れた量子化を行う。また以下で見るように、重み行列の要素の値はほぼ正負対称な分布となるので、正負対称に量子化を行う。すなわち、0 から $W^{\max}$ の区間を $(D/2-1)$ 分割し、$-W^{\max}$ から 0 の区間を $D/2$ 分割することで、量子化された行列 $[\boldsymbol{W}]_q$ を得る。その後、$\boldsymbol{W}' = \boldsymbol{A}[\boldsymbol{W}]_q^{-1}$ のように、$\boldsymbol{W}'$ を更新し、同様の手順で $\boldsymbol{W}'$ も量子化する。

4　SPADE による音響モデルの圧縮

SPADE [3] でも、SVD と同じく重み行列 $\boldsymbol{A}$ を低ランク近似する。その際に $\boldsymbol{A}$ を、図 2 のように、実数行列 $\boldsymbol{C}_{J\times K} \in \mathbb{R}$ と 3 値行列 $\boldsymbol{M}_{K\times I} \in \{-1, 0, 1\}$ の積に分解する。重み行列 $\boldsymbol{A}$ は 0 付近の値をとることが多いため、$\{-1, 1\}$ の 2 値とする [4] よりも、0 を加え

*Acoustic model compression by integer bases decomposition and quantization. by TACHIOKA, Yuuki and AMBAI, Mitsuru (Denso IT Laboratory)

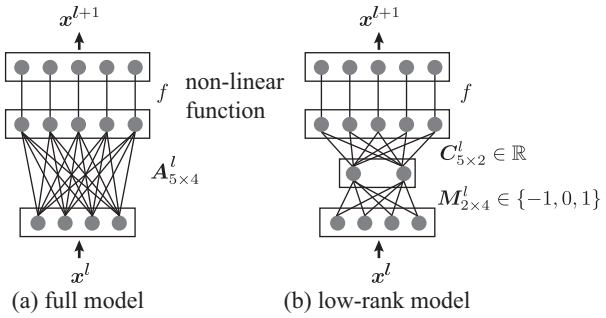


Fig. 2　DNN compression by SPADE.

Table 1　Computational costs of a layer-wise propagation.

	# Additions	# Multiplications
Full	$I \times J$	$I \times J$
SVD	$K(I+J)$	$K(I+J)$
SPADE	$\leq K(I+J)$	$K \times J$

Table 2　Required memory [byte] of weight matrices at each layer where B[byte] is the size of each element of weight matrices.

	Required memory [byte]
Full	$B \times I \times J$
SVD	$B \times K(I+J)$
SPADE	$B \times K(\frac{1}{4B}I + J)$

た 3 値としたほうがよい。これによって、$\boldsymbol{Mx}$ のベクトル行列積の計算が符号反転と加算のみででき乗算が必要なくなるため、必要な計算量が減少する。さらに、$\boldsymbol{M}$ の各要素は 2 ビットで表せるため、メモリ量も削減できるという利点もある。表 1 と 2 に、SVD と SPADE でそれぞれ必要な計算量とメモリ量をまとめた。目的の分解

$$\boldsymbol{A}^l_{J\times I} \approx \boldsymbol{C}^l_{J\times K}\boldsymbol{M}^l_{K\times I} \tag{4}$$

を解くためには、目的関数 $\mathcal{J} = ||\boldsymbol{A} - \boldsymbol{CM}||^2_F$ を最小化すればよい。以下、層毎にこれを行うため、自明な行列の添字は可読性向上のため省略する。しかしながら $\mathcal{J}$ の直接的な最小化は困難なため、これをランク 1 近似により、逐次的に最適化する。ランク 1 近似では、$\boldsymbol{C}$ の k 列目を $\boldsymbol{C}(:,k)$、$\boldsymbol{M}$ の k 行目を $\boldsymbol{M}(k,:)$ として、目的関数

$$\mathcal{J}^k = ||\boldsymbol{R} - \boldsymbol{C}(:,k)\boldsymbol{M}(k,:)||^2_F \tag{5}$$

を k ごとに最小化することになる。ここで $\boldsymbol{R} \in \mathbb{R}^{J\times I}$ は、初期値を $\boldsymbol{A}$ とする残差行列である。残差を更新しながら、$\boldsymbol{C}(:,k)$ と $\boldsymbol{M}(k,:)$ を交互に最適化する。まず $\boldsymbol{M}(k,:)$ を固定すると、$\boldsymbol{C}(:,k)$ は

$$\boldsymbol{C}(:,k) = \left[\frac{\boldsymbol{M}(k,:)}{\boldsymbol{M}(k,:)\boldsymbol{M}(k,:)^\top}\boldsymbol{R}^\top\right]^\top \tag{6}$$

Table 3　Setup for the automatic speech recognition systems.

Sampling frequency	16 kHz
Window length/shift	25 ms/10 ms
Features	0–22th FBANKs + Δ + ΔΔ
HMM states	2,500 shared triphone states
DNN nodes per layer	1024 nodes
DNN layer size	7 layers
Vocabulary size	5,000

のように最小 2 乗法で求められる。$\top$ は転置である。つぎに $\boldsymbol{C}(:,k)$ を固定すると、$\boldsymbol{M}$ は $\{-1,0,1\}$ のいずれかのため、$i(=\{1,\ldots,I\})$ 列について 3 通り、すべての列に対して $3\times I$ 通りの総当たりで、

$$\boldsymbol{M}(k,i) = \underset{\alpha\in\{-1,0,1\}}{\arg\min}\ ||\boldsymbol{R}(:,i) - \alpha\boldsymbol{C}(:,k)||^2_2 \tag{7}$$

のように、最小値をとる $\boldsymbol{M}$ を求められる。$\boldsymbol{R}$ を現時点での残差 $\boldsymbol{R} \leftarrow \boldsymbol{R} - \boldsymbol{CM}$ で更新し、k に 1 を加えて式 (6) に戻り同様に更新する。最後に、このようにして得られた行列のうち、実数値行列 $\boldsymbol{C}$ を対象としてファインチューニングを行った[1][1, 2]。

5　実験

5.1　実験条件

第 4 回 CHiME チャレンジ [5] において、提案手法の有効性を確認した。これは、発話が「ウォールストリート・ジャーナル」から採られている中語彙の騒音下音声認識タスクである。実データ (「real」) およびシミュレーションデータ (「simu」) の 2 種のデータがある。それぞれは、「バス」、「カフェ」、「歩行者天国」および「街頭」の 4 環境からなる。以下に示す単語誤り率 (Word Error Rate; WER) は、4 環境の平均 WER である。学習セットは、real と simu それぞれで、4 および 83 話者による 1,600 と 7,138 発話からなる。開発 (dt) および評価 (et) セットは、real と simu ともに、4 話者によるそれぞれ 1,640 と 1,320 発話からなる。本報では、騒音抑圧を行わない 1ch トラックの音声で評価する。音響特徴量は、0 次から 22 次のフィルターバンク (FBANK) 特徴量とその Δ と ΔΔ 特徴量を使っている。

音響モデルは、学習セットにより学習し、パラメータを開発セットの WER により調整した。表 3 に音声認識の設定を示す。DNN 学習には、Kaldi ツールキットの「nnet1」を使い、7 層の制約付きボルツマンマシンから始めて、各隠れ層にシグモイド活性化関数 f を持つ DNN を構築した。開発セットのクロスエントロピーの減少分が閾値以下であった場合には、

[1] オリジナルの SPADE では、重み行列の再学習 (ファインチューニング) はしない。

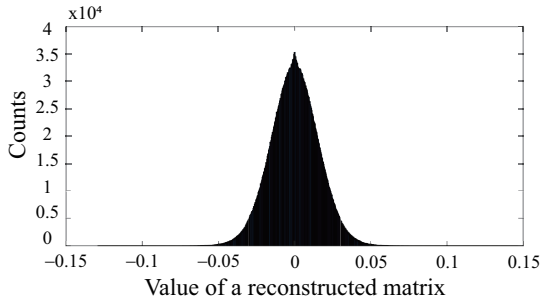


Fig. 3 Histogram of SVD errors ($\boldsymbol{A}^{l}_{J\times I} - \boldsymbol{A}^{l+\frac{1}{2}}_{J\times K}\boldsymbol{A}^{l}_{K\times I}$).

Table 4 WER[%] comparison between full and SVD models. ($\tau = \{0.5, 0.25, 0.1\}$)

	model	dt		et	
	size [MB]	simu	real	simu	real
Full	115.02	17.66	16.21	25.99	30.03
SVD(0.5)	68.75	17.67	16.29	26.18	30.28
SVD(0.25)	33.35	18.21	16.79	26.55	30.59
SVD(0.1)	14.93	21.76	21.14	30.96	36.28

学習率を初期の学習率 (0.008) から低減していく方法で学習した。9 隣接フレームの特徴量をまとめて入力した。

SVD でのパラメータ K は特異値の上位 τ が残るような数に設定した。すなわち $\sum_{i=1}^{K}\sigma_i / \sum_{i=1}^{I}\sigma_i \geq \tau$ を満たす最小の K とした。ここでは τ は 0.5、0.25、0.1 の 3 水準とした。SPADE では SVD とモデルサイズを揃えて、実験を行った。

5.2 SVD でのパラメータ数と認識性能の比較

SVD による低ランク近似 ($\tau = 0.5$) 後の行列を合成した行列 ($\boldsymbol{A}^{l+\frac{1}{2}}\boldsymbol{A}^{l}$) と元の重み行列 $\boldsymbol{A}^{l}$ との要素ごとの差異を、図 3 に示す。このように誤差は約 0.05 以下であり、よく近似できている。SVD を行った場合の元のモデルとの認識性能の差異を、表 4 に示す。$\tau = 0.5$ では最大 0.2%程度であまり差異がない。$\tau = 0.25$ までは 0.5%程度の差異で小さいが、$\tau = 0.1$ になると顕著に性能が低下していることが分かる。

5.3 SVD(量子化) でのパラメータ数と認識性能の比較

SVD($\tau = 0.5$) 後のモデルを対象として、量子化を行った。量子化 SVD(qSVD) の結果を、表 5 に示す。$\boldsymbol{A}^{l+\frac{1}{2}}$ と $\boldsymbol{A}^{l}$ の量子化分割数 D をそれぞれカッコ内に示している。ハイフンは量子化なしを表している。はじめの 3 行は、$\boldsymbol{A}^{l}$ のみを量子化した場合である。8 量子化 (3 ビット) では性能が顕著に低いが、16 量子化 (4 ビット) では 1%程度の性能低下、32 量子化 (5 ビット) では量子化前と同程度の性能となっている。モデルサイズは 6 割程度になっていることから、量子化がサイズ縮減に有効であることが分かる。実装の観点からは 4 ビット量子化の方が望ましいので、$\boldsymbol{A}^{l}$ には 16 量子化を採用して、$\boldsymbol{A}^{l+\frac{1}{2}}$ も量子化する。結果を 4 行目以降に示す。4 行目の結果から、4 ビット-4 ビットの量子化では性能が低いことが分かる。そのため、5〜8 行目までは、実用的な 8 ビット-4 ビットまでモデルサイズを大きくして実験した。このように、上層の量子化を 128(7 ビット) 以上にすれば、どちらも量子化しても性能の低下は 1.5%程度で許容できる範囲である。8 ビット-4 ビット量子化では、モデルサイズは 23%、元のモデルからでは 14%に削減できている。最下段に下層のパラメータをリッチに量子化した場合を示す。モデルサイズはほぼ同じなのにも関わらず、性能が大幅に低い。SPADE と同様に、上層を詳細にモデル化する必要があることが分かる。分布の形状に合わせて不当間隔で量子化を行うなどの工夫をすれば、さらに量子化数を低減することもできると思われる。

Table 5 WER[%] of quantized SVD models. The target was SVD(0.5) model.

	model	dt		et	
	size [MB]	simu	real	simu	real
SVD(0.5)	68.75	17.67	16.29	26.18	30.28
qSVD(-,8)	39.12	33.51	31.62	44.07	51.45
qSVD(-,16)	40.14	18.60	17.17	27.32	31.40
qSVD(-,32)	41.17	17.53	16.40	26.05	30.04
qSVD(16,16)	11.61	44.39	44.73	58.72	65.19
qSVD(32,16)	12.63	20.03	18.78	30.06	34.22
qSVD(64,16)	13.65	19.70	17.82	28.39	32.53
qSVD(128,16)	14.67	19.12	17.33	27.72	31.72
qSVD(256,16)	15.68	18.72	17.17	27.33	31.73
qSVD(16,256)	15.70	53.47	52.65	67.25	73.67

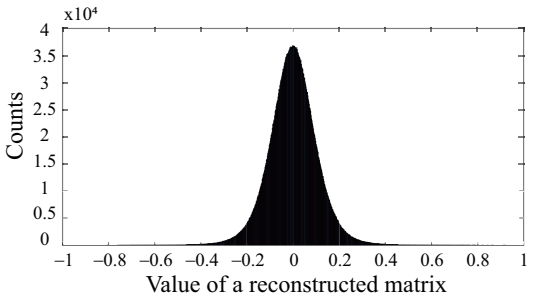


Fig. 4 Histogram of SPADE errors ($\boldsymbol{A}^{l}_{J\times I} - \boldsymbol{C}^{l}_{J\times K}\boldsymbol{M}^{l}_{K\times I}$).

5.4 SPADE でのパラメータ数と認識性能の比較 (ファインチューニングなし)

SPADE による低ランク近似後の行列を合成した行列 ($\boldsymbol{CM}$) と元の重み行列との要素ごとの差異を、図 4 に示す。SVD とパラメータ数は同じでも、分布の形状は図 3 と似ているものの、三値分解の影響で誤差の分散が大きい。認識性能を、表 6 に示す。$\tau = 0.5$ のモ

Table 6　WER[%] comparison between full and SPADE models without fine tuning.

	model size [MB]	dt simu	dt real	et simu	et real
Full	115.02	17.66	16.21	25.99	30.03
SPADE(0.5)	38.09	20.51	18.97	29.45	33.95
SPADE(0.25)	19.32	28.24	25.57	38.13	43.88
SPADE(0.1)	9.56	79.71	81.09	85.64	89.22

Table 7　WER[%] of SPADE with fine tuning models.

	model size [MB]	dt simu	dt real	et simu	et real
SPADE(0.5)	38.09	17.77	16.55	26.55	30.44
SPADE(0.25)	19.32	18.33	17.02	26.87	30.74
SPADE(0.1)	9.56	19.29	17.86	27.94	31.91

デルサイズであっても、パラメータ数が同じ SVD に比べ、性能が 3〜4%程度低下している。これは、三値分解で表現力が低下しているのに加え、SPADE の最適化でのランク 1 近似が必ずしも最適な分解を達成できていないことに起因していると考えられる。$\tau = 0.1$ のモデルサイズでは、SVD と比べても、認識性能が著しく低い。

5.5　SPADE でのパラメータ数と認識性能の比較 (ファインチューニングあり)

これに対して、SPADE による分解後に実数行列 $\boldsymbol{C}$ のみをファインチューニングした場合の結果を、表 7 に示す。これによれば $\tau = 0.5, 0.25$ の場合に 0.5%以下の低下に抑えられ、$\tau = 0.1$ の場合にもモデルサイズを元の 8.3%に削減しながら 1〜2%程度の認識率低下に抑えられている。

5.6　モデルサイズと認識性能のまとめ

各手法でのモデルサイズと WER の比較を、図 5 に示す。まず青の SVD はパラメータ数を削減しても性能低下が緩やかに起こることが分かる。緑の SVD 後の量子化も有効で性能を落とすことなく、パラメータ数を削減できている。これに対して、赤の SPADE はパラメータ数の削減に伴い急激に性能が低下する。ただし、ファインチューニングを行うことで、SVD よりも性能がよく、量子化 SVD と同程度の性能が得られた。

6　まとめ

DNN 音響モデルのモデルサイズ縮減を目的として、SVD 後の行列の量子化と、SPADE による三値行列と実数行列の積への分解の 2 通りの方法を検討した。結果として、両手法により SVD 後のモデルから 1/5 程度のモデルサイズ (元のモデルの 1/10 程度) で、元

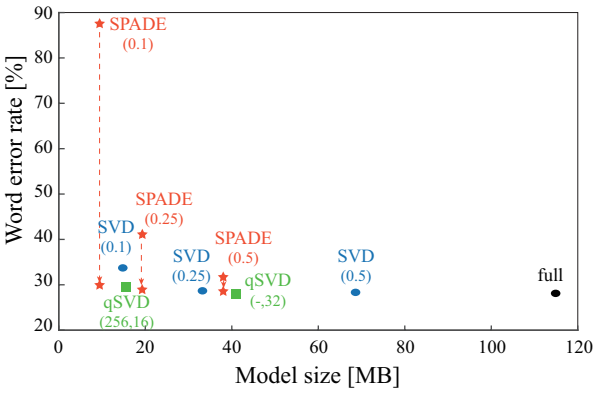


Fig. 5　Relationship between model size and average WER[%] (et).

のモデルと比べて 1〜2%程度の認識性能低下に抑えられた。メモリ量については両手法同等である。計算量については、量子化 SVD では固定小数点化が必要だが、SPADE の実装はそれに比べれば易しいことから SPADE による圧縮が有用であることが分かった。

参考文献

[1] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," Proceedings of INTERSPEECH, pp.2365–2369 (2013).

[2] 太刀岡勇気，渡部晋治，ルルージョナトン，ハーシージョン，"低ランク DNN 音響モデルの騒音下音声認識での評価と系列の識別学習," 情報処理学会論文誌，**57**，1080–1088 (2016).

[3] M. Ambai and I. Sato, "SPADE: Scalar product accelerator by integer decomposition," Proceedings of ECCV, pp.267–281 (2014).

[4] S. Hare, A. Saffari, and P.H. Torr, "Efficient online structured output learning for keypoint-based object tracking," Proceedings of CVPR, pp.1894–1901 (2012).

[5] E. Vincent, S. Watanabe, A.A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," Computer Speech and Language, **46**, 535–557 (2016).