計算コストの小さい深層学習モデルアーキテクチャの構築

中 西 健

東京大学理学系研究科

1 はじめに

近年は画像分類・画像認識・音声処理・自然言語処理をはじめとするあらゆる分野で深層学習が目覚ましい成果を挙げている。また，GPU を初めとした計算機能力の向上により非常に大きなニューラルネットワークを高速に学習させることが可能となった。一方，スマートフォンや組み込み機器のような利用できる計算リソースが限られた環境において実用的な応答速度で学習済モデルによる推論をしたい場合はモデル自体のメモリコスト・計算コストを抑えることが求められる。そこで，本課題ではメモリコスト・計算コストの小さい深層学習モデルアーキテクチャを構築する手法を提案し，検証した。

2 目的

計算コストはモデルのネットワークの結合数に比例するため，既存研究でははじめからネットワークの結合が素なモデルを作って学習させたり，モデルの学習済パラメータに対して低ランク近似やスパース化を行うことによって学習パラメータ数を削減したりする手法が取られていた [1–10]。本課題では，学習パラメータ数を削減する新しい手法を提案・検証した。従来の学習パラメータ数削減手法では，「学習→パラメータ削減→再学習」という手順を必要とするものが多かった。提案する手法では，バッチ正規化の出力の分散を減少させる作用を持つコスト関数を追加し，出力の分散がゼロになるものを随時消去していくアルゴリズムを組み込む。これによって，一度の学習中に不要なチャネルを消していき，パラメータを計算機で効率的に計算できる形で随時削減していくことができる手法を提案する。

3 手法

まず，バッチ正規化層について説明する。バッチ正規化層は多くのニューラルネットワークで用いられており，例えば，ResNet[11] を構成する ResBlock の中にも図 1 のように用いられている。

バッチ正規化層に入力されるミニバッチ数を N，チャネル数を C とし，ミニバッチの n 番目のデータの c 番目のチャネルの i 番目の要素を

$$x_{c,i}^{(n)}$$

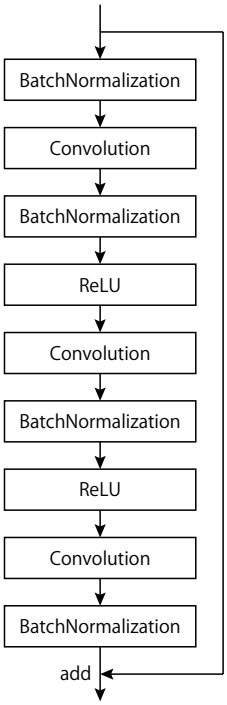

図 1: ResBlock の一例。BatchNoemalization はバッチ正規化層，Convolution は畳み込み層，ReLU は活性化関数のひとつであり，$\mathrm{ReLU}(x) := \max(0, x)$ である。ResNet は主にこのような ResBlock が積み重なってできている。

と表す。バッチ正規化層ではまずデータのチャネルごとに平均

$$\mu_c := \frac{1}{NC} \sum_{n=1}^{N} \sum_{c=1}^{C} x_{c,i}^{(n)}$$

と分散

$$\sigma_c^2 := \frac{1}{NC} \sum_{n=1}^{N} \sum_{c=1}^{C} (x_{c,i}^{(n)} - \mu_c)^2$$

を計算し，各チャネルの平均が 0，分散が 1 となるように次のように正規化する。(ϵ は十分小さな値とする。)

$$\hat{x}_{c,i}^{(n)} := \frac{x_{c,i}^{(n)} - \mu_c}{\sqrt{\sigma_c^2 - \epsilon}}$$

その後，各チャネルごとに独立な学習可能パラメータ γ_c, β_c を用いて，次のように出力 y の要素 $y_{c,i}^{(n)}$ を計算する。

$$y_{c,i}^{(n)} = \gamma_c \hat{x}_{c,i}^{(n)} + \beta_c \tag{1}$$

今回の手法では，コスト関数に γ_c をゼロに近づけるような L1 正則関数を付け加えた。さらに，学習の途中から，更新により γ_c の値の符号が変化したものを随時ゼロに固定した。これにより，$\gamma_c = 0$ となるチャネルでは 1 が

$$y_{c,i}^{(n)} = \beta_c$$

となって，$\hat{x}_{c,i}^{(n)}$ に依存しなくなり，直前の畳み込み層でこのチャネルを出力する必要が無くなる。これによって，直前の畳み込み層の出力チャネル数を減らすことが可能となる。さらに，このバッチ正規化層のあとに活性化関数→畳み込み層と続く場合は，$\gamma_c = 0$ となるチャネルの出力を一つにまとめることができる。これによって，直後の畳み込み層の入力チャネル数も減らすことが可能となる。

このパラメータ数削減方法のメリットは，チャネルごとに消すか消さないかを決めることができることと，新たに不要となったチャネルを判断して無効化できることである。畳み込み関数内で使われるパラメータは多次元配列の状態で格納されているが，これをただスパース化するだけでは，(よほどスパースにならない限り) 実際の計算機上において，計算速度やパラメータの保存に必要なメモリ量は変化しないと考えられる。一方，このパラメータ数削減方法では，多次元配列の入出力のチャネル方向の幅自体を縮小させることができるため，実際の計算機上での計算速度を向上させ，パラメータの保存に必要なメモリ量を削減させられると考えられる。

4　実験条件

実験には代表的なニューラルネットワークである ResNet50[1] を元にして，コスト関数にバッチ正規化層の γ_c をゼロに近づけるような L1 正則関数を追加したものを用いた。比較対象としては，元の ResNet50 を用いた。このニューラルネットワークのパラメータ数は 23,573,642 であった。学習とテストに用いるデータセットは，それぞれ CIFAR10 の学習データセット (画像 50,000 枚，10 クラス) とテストデータセット (画像 10,000 枚，10 クラス) を用いた。バッチサイズは 125 に設定し，500 エポック学習させた。パラメータの更新アルゴリズムには MomentumSGD を用い，学習率の初期値を 0.05 に設定し，学習率を 50 エポックごとに半分にした。パラメータの荷重減衰パラメータを 0.0005 に設定した。提案手法では，コスト関数に追加した L1 正則関数の重みとして 10 通りの値 (0.001, 0.0008, 0.0006, 0.0005, 0.0004, 0.0003, 0.0002, 0.0001, 0.00005, 0) を設定し，さらに，250 エポック終了時以降では，更新により γ_c の値の符号が変化したものを随時ゼロに固定した。

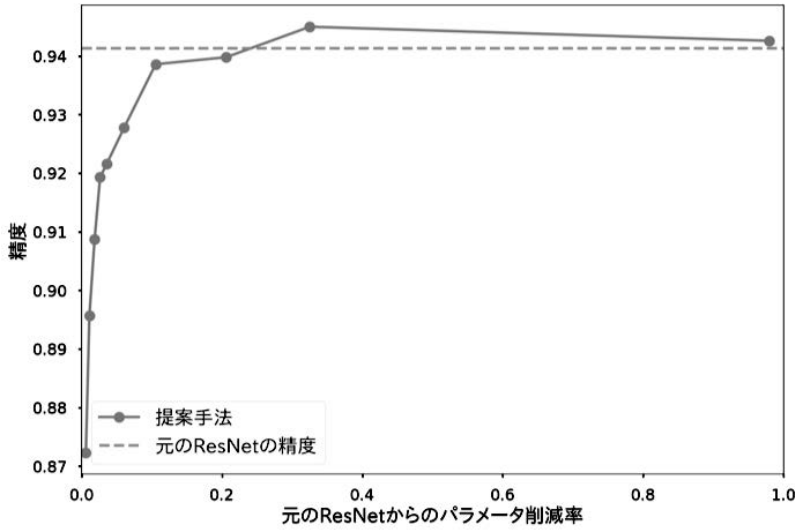

図 2: 実験結果を比較したグラフ。横軸は元の ResNet50 のパラメータ数を基準としたときの学習済みモデルのパラメータ削減率を示している。縦軸は学習済みモデルの Cifar10 のテストデータにおける分類精度を示している。実線は，提案手法の結果を表しており，点線は，元の ResNet50 における分類精度を表している。ただし，各実験はインターン期間中に 1 度ずつしか行うことができなかったため，エラーバーは描けていない。本来は各実験を複数回行ったあとでなければ正確なことは言えないが，この図からは，予測精度をほとんど減少させずにモデル内のパラメータを 9 割程度削減できたことがわかる。

5 結果

画像データセット cifar10 の 10 クラス分類タスクについて，提案した深層学習モデルを用いて行った実験結果を図 2 に示した。ただし，各実験はインターン期間中に 1 度ずつしか行うことができなかったため，エラーバーは描けていない。本来は各実験を複数回行ったあとでなければ正確なことは言えないが，図 2 を見ると，予測精度をほとんど減少させずにモデル内のパラメータを約 90%削減できたことがわかる。

[1] `https://github.com/mitmul/chainer-cifar10` を参考にさせていただいた。

6　考察

今回の実験の結果から言えることとしては，ResNet50 は cifar10 の画像データセットの分類タスクに関しては過剰にパラメータ数があるということである。そして，提案したパラメータ削減アルゴリズムは，このようなニューラルネットワークの，実際の計算機上での計算速度向上やパラメータの保存に必要なメモリ量削減に有効な形でパラメータ数を削減できるということがわかる。さらに，このアルゴリズムは既存のニューラルネットワークに容易に組み込むことができるため，スマートフォンや組み込み機器のような，利用できる計算リソースが限られた環境での利用を想定した深層学習モデルの作成に非常に役に立つと考えられる。

参考文献

[1] Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257*, 2017.

[2] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. 2015. *arXiv preprint arXiv:1506.02626*, 2015.

[3] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

[4] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[5] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[6] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[7] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[8] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[9] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 138–145, 2017.

[10] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.