Model Compression Optimization in Neural Network Using Adaptive Differential Evolution

Nagi HANAMOTO* , Naoya IKUSHIMA* , Keiko ONO** and Erina MAKIHARA**

(Received October 14, 2022)

Various deep neural network models have been proposed; however, with the emergence of big data, high computing power is required to train and use such models. A small adequate model should be modeled if a rich computing environment is unavailable. It is challenging to clarify how to build a relatively small, high-performance model. In this paper, we propose a model compression method based on differential evolution. Specifically, the proposed method optimizes not only network structures and but also weights simultaneously by the differential evolution. Experiment results showed that the proposed method appropriately reduced weight parameters after optimization and had similar accuracy as the original model without compression by adjusting the compression rate.

Key words : differential evolution, model compression, Pruning, MNIST

キーワード : 差分進化, モデル圧縮, 枝切り, MNIST

適応的差分進化を用いたNeural Networkにおけるモデル圧縮

花本 凪・ 幾島 直哉・ 小野 景子・ 槇原 絵里奈

1. はじめに

Neural Network（以下，NN）は画像分類や顔認識，テキスト認識などで使用される学習モデルである．これらの技術は医療分野やセキュリティ機器，株取引をはじめとした様々な分野で利用されている．その利用用途は拡大傾向にあり，交通や家電のように生活においてもNNを用いたシステムが普及している．特に自動運転のような交通での利用においては，人間の代わりに判断を行うシステムが増加している．そのため，NNに求められるAccuracyは高いものになっている．また，自動車や家電機器，ウェアラブルデバイスなどに搭載されるシステムでは処理速度が重要視されるため，NNには高い処理速度が必要である．

NNのAccuracy向上を目的として，NNを最適化する手法に関する研究が行われている．NNの最適化にはNNの構造そのものを最適化するトポロジー最適化と，重みやバイアスを最適化するパラメータ最適化が存在する．パラメータ最適化の方法として誤差逆伝搬法と差分進化が挙げられる．誤差逆伝搬法とは，学習の際に発生する誤差を微分を用いた計算によって減

* Graduate School of Science and Engineering, Doshisha University, Kyoto
Telephone:+81-774-65-6930, Fax:+81-774-65-6716, E-mail: hanamoto.nagi@mikilab.doshisha.ac.jp
** Faculty of Science and Engineering, Doshisha University, Kyoto
Telephone:+81-774-65-6930, Fax:+81-774-65-6716, E-mail: kono@mail.doshisha.ac.jp,
emakihar@mail.doshisha.ac.jp

少させることで，NN の最適化を行うアルゴリズムであり，発表された 1970 年代から現在まで使用されている [1]．Accuracy の向上に伴い，NN の最適化で扱うデータ数やデータの情報を有する次元数が増加している．結果，NN の最適化および使用には多くの計算処理が必要である．そのため，誤差逆伝搬法で用いる微分計算には時間が必要であり，データ数および次元数の増加に影響を大きく受ける．一方，差分進化は微分計算を必要とせず，単純な計算によって解の候補を繰り返し改良することで NN の最適化を行うアルゴリズムである．誤差逆伝搬法と比較し，単純かつ高速に動作する差分進化は近年注目されている．

一方, 処理時間や計算量の低減を目的とした手法として, モデル圧縮がある．モデル圧縮とは Deep Learning（以下，DL）に対して処理速度の高速化を目的とした手法である．モデル圧縮の手法として Pruning や Quantize，Distillation が存在する．本研究では直接パラメータを削減する Pruning を用いた手法を適用し，差分進化による最適化への影響を調査する．

2. 差分進化

2.1 概要

差分進化は，進化計算アルゴリズムを応用した，多点探索を行うパラメータ最適化手法である [2]．差分進化は微分が不可能な非線形問題や非凸問題の最適化問題に適用され, 他の手法との差異として, 複雑な計算を必要としないことから高速に動作し，微分が不要であることや操作パラメータの数が少ないことから頑健性を有する [3]．差分進化において設定を必要とするパラメータは世代数やスケーリングファクタ，交叉確率である．世代数は探索の回数を示すものであり，スケーリングファクタは探索の範囲を操作するパラメータである．交叉確率は交叉において，元となる個体の要素を引き継ぐ際に使用するパラメータである．

差分進化では JADE や jDE，IDE などの複数の方法が存在する．各方法の差異はパラメータの設定方法や子個体の生成方法にある．本研究では一般的な適応的差分進化である jDE を用いる．

2.2 アルゴリズム

差分進化のアルゴリズムは，次の 4 つの工程で構成される．はじめに NP 個の個体を生成する初期化が行われ，この個体集団は設定された範囲に一様に分布するベクトルで構成される．その後は全ての個体に対して突然変異，交叉，選択を繰り返し行うことで探索点の更新を行う．

突然変異の段階では，対象の個体 $\boldsymbol{x}_i$（$i = 1, 2, ..., NP$）に対して，与えられた生成方法によって変異個体 $\boldsymbol{v}_i$ が生成される．変異個体の生成にはスケーリングファクタである $F \in [0, 1]$，および $\boldsymbol{x}_i$ を除いた個体の中から $\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}, \boldsymbol{x}_{r3}$ の 3 つを選択し，使用する．変異個体 $\boldsymbol{v}_i$ の生成式を式（1）に示す．

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}). \tag{1}$$

交叉では，更新の対象とする個体および突然変異で生成した変異個体を用いて子個体となるベクトル $\boldsymbol{u}_i$ を生成する [4]．子個体の生成には次元数 D を最大とした自然数からランダムに選択した $j_{rand} \in [1, 2, ..., D]$，および交叉確率 CR（$0 \leq CR \leq 1$）を使用する．子個体を生成する式を式（2）に示す．また．突然変異と交叉による DE の進化操作を 2 次元で表現したものを Fig. 1 に示す．

$$\boldsymbol{u}_{i,j} = \begin{cases} \boldsymbol{v}_{i,j}, & rand_{i,j} \leq CR \text{ or } j = j_{rand} \\ \boldsymbol{x}_{i,j}, & \text{otherwise.} \end{cases} \tag{2}$$

選択では，変異の対象となる個体 $\boldsymbol{x}_i$ と交叉によって生成された $\boldsymbol{u}_i$ を比較する．2 つの良い個体を採用し，もう一方を削除することで 1 個体に対する更新を行う．以上の操作を全ての個体に対して行い，探索点の更新を行う．

2.3 jDE

差分進化で操作するスケーリングファクタ，および交叉確率は大域的な探索を行う際に大きな影響を与える．これらのパラメータを設定する際に異なるアプローチを設けることで自己適応を行う手法が適応的差分進化である．

最も一般的な手法として，2006 年に Brest らによって提案された jDE が挙げられる [5]．jDE で使用する

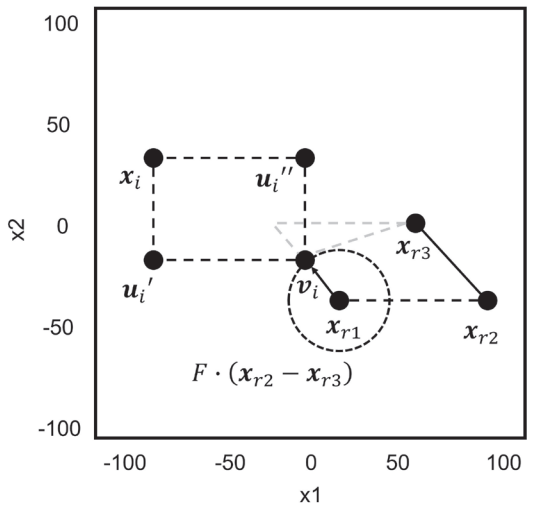


Fig. 1. Evolutionary operations of differential evolution

子個体の生成方法は DE/rand/1 である．jDE では各個体がそれぞれのスケーリングファクタおよび交叉確率を有し，最適化の際に制御パラメータを適応的に変化させることで調整し最適化を行う．初期値としてスケーリングファクタは 0.1，交叉確率は 0.9 に設定する．変化を行う確率として $\tau 1, \tau 2$，スケーリングファクタの上限値，下限値として F_u, F_l を使用し，式（3）および式（4）に子個体のパラメータ算出式を示す．

$$F_{i,g+1} = \begin{cases} F_l + rand_1 \cdot F_u, & \text{with probability } \tau 1 \\ F_{i,g}, & \text{otherwise} \end{cases} \tag{3}$$

$$CR_{i,g+1} = \begin{cases} rand_2, & \text{with probability } \tau 2 \\ CR_{i,g}, & \text{otherwise} \end{cases} \tag{4}$$

g：更新世代

$\tau 1, \tau 2 = 0.1$

$F_l = 0, F_u = 1$ or $F_l = 0.1, F_u = 0.9$

$rand_1, rand_2$：[0,1] の範囲で一様分布に従う数値

2.4 Neuroevolution

Neuroevolution は，遺伝的アルゴリズムを NN の学習に使用する機械学習手法である．遺伝的アルゴリズムとは，生物の進化の仕組みを模倣した解探索手法である [6]．差分進化は遺伝的アルゴリズムの１つであり，差分進化における突然変異，選択の工程は生物進化の原理に則ったものといえる．NN は人間の脳をモチーフとした学習モデルであり，ノードをエッジで繋ぐように構成される．ノードの数値を足し合わせる際に乗算する値が重みであり，重みを適切な値にすることが NN の最適化である．差分進化を NN の学習で利用するにあたり，最適化を行うパラメータを有したベクトルが初期化の段階で生成される個体となる．これらの個体を更新することで学習を行う．次に子個体が生成され，選択を行う．差分進化の選択において採用対象となる個体は，適応度が高いものである．Neuroevolution では適応度として，Accuracy や cross-entropy などを用いる．

Neuroevolution は重みのほかに構造についても最適化の研究がされており，近年注目を集めている [7]．

3. モデル圧縮

3.1 概要

モデル圧縮とは DL に対して行われる処理であり，計算量を低減することを目的としている．モデル圧縮には複数の手法が存在する．NVIDIA とスタンフォード大学によって提案されたものとしてプルーニングや量子化，ウェイトシェアリングなどが挙げられる [8]．これらの手法は全て DL における重みの削減を目的としており，その過程においてメモリ量やメモリの帯域，演算量を少なくすることが可能となる．また，重みの削減によって生じる Accuracy の低下に対する対策や，Accuracy の低下を抑える削減手法の研究が行われている．

3.2 プルーニング

プルーニングはモデル圧縮手法の１つであり，決定木の学習で使用されていたものである．モデル圧縮におけるプルーニングでは DL の重みを直接取り除き，計算自体を行わないように再構築する手法である [8]．削除する重みの選択方法としてランダムに重みを選択する方法や，重みが閾値以下のもの全てを対象とする方法が存在する．重みを削除した後は再学習を行い，残った重みを適切な値に設定することで Accuracy の低下を抑えることが可能である．プルーニングは単純

でありながらも多くの重みを削減可能である特徴があり，DL のみならず中間層を持たない NN にも導入可能であると考えられる．

4. 先行研究

4.1 Neuroevolution に関する先行研究

Neuroevolution の例として，Deep Neuroevolution（以下, DNE）が挙げられる．DNE とは, Deep Neural Network（以下，DNN）のパラメータを進化計算を用いて更新する手法である [9]．Salimans らは DNN の構造を初期化段階で定義し，DNN のパラメータを最適化する手法の提案を行った [10]．DNN では，パラメータおよびトポロジーについて最適化を行う．DNN は通常の NN よりも大量のパラメータを有し，DNE ではこれらを十分に最適化することが可能である．

4.2 CNN に対するプルーニングの先行研究

2015 年にスタンフォード大学と NVIDIA の研究者らによって発表された論文では，ニューラルネットワークに対してプルーニングを行う研究が行われた [8]．ニューラルネットワークを学習した後，定めた閾値以下の重みに対してプルーニング処理を行う．プルーニングによってパラメータを削除したのちに再学習を行うことで，低下した正確度の向上を図る．結果，AlexNet で 88.9 %，VGG-16 で 92.3 %の重みを削減することに成功した．

5. 提案するモデル圧縮手法

5.1 概要および手法の手順

本手法は，差分進化を用いた NN 最適化において，最適化の途中に不要と判断した重みパラメータを削除することでモデル圧縮を行うものである．本研究で使用する jDE では，複数個体の差分を用いて変異および子個体の生成を行う．したがって，不要とされる NN の重みに該当するベクトル要素を全ての個体において 0 にすることで重みの削除を行う．全個体において 0 であるベクトル要素は，差分が存在しないことから交叉および選択による更新が行われず，適応度の計算および評価において影響力を持たない．

本手法における重みの削除は 2 つの操作によって行われる．それぞれを操作 1, 操作 2 とし，以下に提案手法を取り入れた NN の最適化手順を示す．また，jDE におけるハイパーパラメータである世代数を G とし，G 以下の整数である g_1 および g_2 をそれぞれ操作 1, 操作 2 を行う世代数とする．

1. 初期個体の生成を行う．
2. 第 1〜$g_1 - 1$ 世代において学習データを使用し，個体の更新を行う．
3. 第 g_1 世代において操作 1 による重みの削減を行い，削減された重みを 0 に設定する．
4. 第 $g_1 + 1$〜$g_2 - 1$ 世代において削減されていない重みの更新を行う．
5. 第 g_2 世代において操作 2 による重みの削減を行い，削減された重みを 0 に設定する．
6. 第 $g_2 + 1$〜G 世代において削減されていない重みの更新を行う．

操作 1 は学習データを用いた削減を行い，操作 2 は削減前の重みを参考に削減を行う．よって，操作 1 は学習データを収集してから行う必要がある．また，操作 2 を行う際に重みが十分に学習されていない状態では，最適化後に大きな影響力を持つ重みを削減してしまう可能性が高くなる．したがって，十分に学習を進めた後に操作 2 を行う必要がある．

5.2 未使用の重みに着目した重み削減操作

操作 1 では，学習における未使用の重みに着目した削減操作を行う．ここでの未使用の重みとは，入力値が常に 0 であるノードの計算に使用される重みを指す．入力値が 0 であるため，適応度に対する影響力を持たないことから，十分な更新が行われない．その後，学習データでは 0 であった入力がテストデータで他の値を持っていた場合に Accuracy は低下し，最適化に余地がある NN となる．よって，未使用の重みは 0 に設定し，削減する必要がある．誤差逆伝搬法ではこれらのような重みに対し学習データから受ける影響が小さいと判断し，0 に近づける処理を行う．差分進化では初期値が乱数で設定され，それらの差分を用いて求められた更新の候補は 0 に近づくとは限らないため，特別な操作が必要である．本操作では，未使用の重み

の判別を学習データから行う．操作としては，次のとおりである．はじめに初期化から削減を行う世代までに使用した学習データを蓄積する．次にそれらの学習データにおいて，常に 0 である入力に該当する重みを判別する．最後に，判別した重みを全ての個体において 0 に設定する．

5.3 最良個体を用いたプルーニング操作

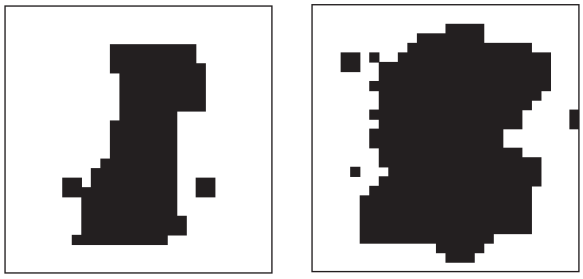
差分進化を用いた最適化を行うとき，最適解に最も近い個体を最良個体と表す．本操作では，プルーニングを行う際に最良個体を参考に削減の対象となる重みを選択する．

一般的にプルーニングで削減する次元を選択する方法として，NN の評価に影響しない重みを選択することが望ましい．そこで本手法では，計算の際に結果に対して影響が少ない順に重みを並び替え，指定した割合の重みに対して削減を行う．はじめに，操作を行う世代において最良個体に注目する．次に最良個体の各重みについて絶対値を用いて並び替える．その後一定の割合の重みを設定し，削減の対象とする．また，削減の方法は操作 1 と同様である．

6. 提案手法における有用性の検証実験

6.1 実験概要

本実験では，手書き文字データである MNIST を使用し，重み削減が Accuracy に与える影響について実験を行う．MNIST の画像はサイズが一定であるため，各画像ごとに数字の背景にあたる部分が存在する．入力値は MNIST の各画素から得られる数値であり，重みは入力値が分類結果に与える影響の大きさを示す数値であることから，背景部分に関係する重みは Accuracy に影響しないと考えられる．よって，これらの重みは Accuracy を保ったまま削減することが可能である．また，MNIST の画像に書かれる数字は大きさや位置がデータごとに異なるため，複数のデータから削減を行う重みを決定する．操作 1 において，使用するデータ数によって削減の対象となる次元数は大きく変化する．Fig. 2 は手書き数字の 1 に該当する画像に限定し，異なるデータ数における入力値が存在する領域範囲を比較したものである．Fig. 2 において各画像の黒い領域が入力値が存在する箇所である．比較からわかるようにデータ数が多くなると入力が存在する領域は増加し，削減の対象は少なくなる．データ数を少なくすることで削減数を増やすことが可能となるが，Accuracy に影響が大きい箇所を削減する可能性が高くなる．



(a) データ数 200　　(b) データ数 2000

Fig. 2. Range of input images with input values

6.2 検証条件

使用する NN は入力層 1 つ，中間層 2 つ，出力層 1 つで構成され，中間層にはシグモイド層とソフトマックス層を用いる．MNIST は 28 × 28 の画像データであるから，NN の入力層は 784 となる．また，ノード数 10 のシグモイド層と入力層からのノード数 784 より，ネットワーク全体では 7840（=784 × 10）の重みが存在する．7840 の重みに加えて，シグモイド層に入力されるバイアスの数 10 を加えた 7850 が今回最適化を行う個体の要素数である．個体数に 500 個，世代数に 1000 を設定し，世代あたり 200 のバッヂ数を用いて jDE によるパラメータの最適化を行う．重み削減の条件として操作 1 は第 10 世代に行い，削減に使用する学習データの数を 2000 とする．また，操作 2 は第 500 世代に削減を行う．

6.3 結果および考察

通常の NN と操作 1 によって重みを削減した NN を比較した結果を Fig. 3 に，操作 1 および操作 2 によって重みを削減した NN を比較した結果を Fig. 4 に示す．また，Table 1 に各手法において削減した重みの数と最終の Accuracy を示す．

結果として，操作 1 を行ったことで Accuracy が低

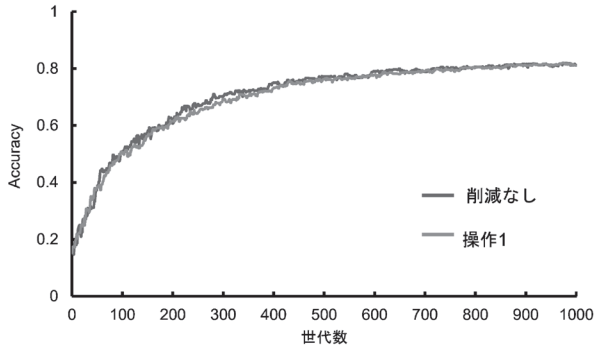


Fig. 3. Comparative experimental results of accracy with operation 1

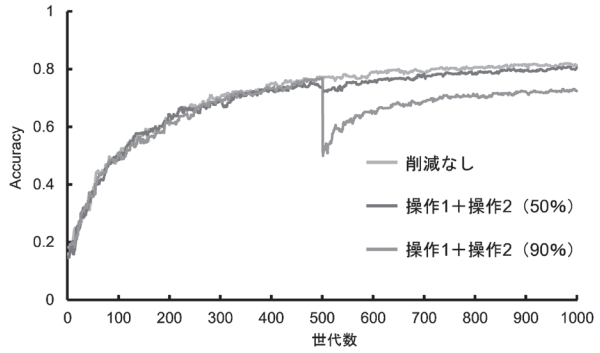


Fig. 4. Comparative experimental results of accracy with operation 2

下する事実は得られなかった．今回削減の対象となった重みは，画像データにおける背景部分のように，値を持たない入力に関係する重みである．これらの重みに対して，差分進化を用いた最適化の途中で削減が可能であるといえる．

操作 2 では操作 1 を用いた削減に加え，残りの 50 %および 90 %の重みの削減を行った．削減を行った世代においては Accuracy が低下しているが，その後再び向上している．このことから最良個体を用いることで，差分進化においてもプルーニングによるモデル圧縮が可能であると考えられる．また，削減数が増加すると Accuracy が低下することから，使用するネットワークやデータによって適切な値の設定が必要である．

7. おわりに

本研究では，差分進化を用いた NN の最適化におけるモデル圧縮手法として，2 つの操作によって削減する重みを選択し，進化操作の特性を用いて削減を行う手法を提案した．そして，実際にモデル圧縮を行い，Accuracy を確認することで有用性の検証を行った．検証では半数以上の重みを削減することに成功し，削減を理由とした Accuracy の低下は確認できなかった．

以上より，提案手法を用いることによって，差分進化においても影響力が少ない重みの最適化や，プルーニングを用いたモデル圧縮が可能である．また，削減する重みの選択方法を改良することでより多くの重みの削減が可能となると考えられる．

Table 1. Number of parameters reduced and accuracy in each operation

操作	重み削減数	Accuracy
削減なし	0	0.812
操作 1	3147	0.820
操作 1+操作 2（50 %）	5530	0.805
操作 1+操作 2（90 %）	7382	0.724

本研究は JSPS 科研費 21K12097 の助成を受けたものです．

参考文献

1) D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-propagating Errors”, *Nature*, vol. 323, no. 6088, pp. 533–536, （1986）.

2) R. Storn and K. Price, “Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”, *Journal of Global Optimization*, vol. 11, pp. 341–359, 01 （1997）.

3) 野津亮，鍔本純也，生方誠希，本田克宏，“差分進化における探索点群の広がりと アルゴリズムの切り替え”，第 35 回ファジィシステムシンポジウム，pp. 78-83,（2019）.

4) K. Price, R. Storn J. Lampinen, “Differential Evolution: A Practical Approaeh Global to Optimizationtion”, *Springer*,（2005）.

5) J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, “Dynamic Optimization Using Self-adaptive Differential Evolution”, *in IEEE Congress on Evolutionary Computation*, pp. 415–422, （2009）.

6) J.H.Holland, “Adaptation in Natural and Artficial Systems”, *The Univercity of Michigan Press*,（1975）.

7) J.Gauci, K. O. Stanley, “A Case Study on the Critical Role of Geometric Regularity in Machine Learn-

ing", *Twenty-Third AAAI Conference on Artificial Intelligence*, 628-633,（2008）.

8) S. Han, J. Pool, J. Tran, W.J.Dally, "Learning both Weights and Connections for Efficient Neural Networks" ,*Neural Information Processing Systems*,（2015）.

9) F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep Neuroevolution Genetic Algorithms are a Computive Alternative for Training Deep Neural Networks for Reinforcement Learning", *arXiv preprint*, 1712.06567（2017）.

10) T. Salimans, J. Ho, X. Chen, S. Sidor and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning", *arXiv preprint*,1703.03864（2017）.