画像分類のための Dense Residual Network の提案

武田 敦志[1,a)]

概要

本稿では、ResNet に対して DenseNet のネットワーク構造を導入した画像分類のための CNN である DenseResNet を提案する。DenseResNet では、直接的な特徴量と抽象化された特徴量の両方を考慮した画像分類を行うため、複数の畳み込み層の出力を統合する仕組みを有する。また、ImageNet-1K と CIFAR-100 を用いて DenseResNet の画像分類精度を評価し、DenseResNet が少量のパラメータ数と計算量の増加のみで画像分類性能を改善することを示す。

1. はじめに

畳み込みニューラルネットワーク（CNN）を用いることにより、高い精度で画像データを分類できるようになった。特に、Residual Network（ResNet）は勾配消失問題を解決するために残差構造を導入した高性能の CNN であり、画像分類のための CNN として広く利用されている [1]。ResNet の性能改善を目的とした研究開発も盛んに行われており、残差構造を変更することにより画像分類性能の向上を目指した CNN [7, 12] や、性能を低下させずに計算量の削減を目指した CNN [2, 9] などが提案されている。また、畳み込み層の数だけではなく、特徴量ベクトルの大きさや入力画像の解像度を適切に調節することにより CNN の性能を向上させる方法が提案されている [11]。

ResNet とは異なる構造を持つ CNN として DenseNet が提案されている [4]。DenseNet では、既に計算を行った複数の畳み込み層の出力を結合したものを新たに計算する畳み込み層の入力値とする。これの構造により、DenseNet は勾配消失問題を解決するだけではなく、直接的な特徴量と抽象化された特徴量から必要なものを選択しながら画像分類を行っている考えられている。一方、DenseNet では、複数の畳み込み層の出力を結合したものを新たな畳み込み層の入力値とするため、畳み込み層の出力チャネル数が入力チャネル数によりも大幅に小さい場合がある。入出力のチャンネル数が大幅に異なる畳み込み層では、積算などの演算に必要な時間よりもメモリアクセスに必要な時間が支配的となる [6]。そのため、DenseNet のパラメータ数や理論上の計算量が ResNet よりも小さい場合でも、DenseNet の実際の計算時間は ResNet よりも大きくなることがある。

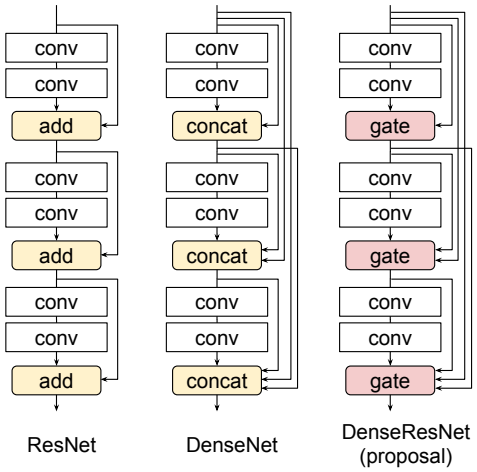

図 1 ResNet [1]・DenseNet [4]・DenseResNet の基本構成

そこで、本稿では、ResNet に対して DenseNet のネットワーク構造を導入した画像分類のための CNN「Dense Residual Network（DenseResNet）」を提案する。図 1 に DenseResNet の基本構成を示す。DenseResNet は、ResNet と同様に 2 層の畳み込み層の出力値にその前の畳み込み層の出力を足し合わせる構造となっているが、DenseNet のように複数の畳み込み層の出力を足し合わせる点が従来の ResNet とは異なる。ただし、複数の畳み込み層の出力を単純に足し合わせた場合、逆誤差伝搬のときに勾配爆発が発生する可能性がある。そこで、前の畳み込み層からの信号強度を調整し、勾配消失問題や勾配爆発問題を解決するために Gate-Module を導入する。Gate-Module は、SENet [3] で提案されている SE-Module と同様に Global Average Pooling を用いて特徴量を集約する構造になっており、この Gate-Module のために必要となるパラメータ数や計算量は非常に少ない。また、Gate-Module を導入することにより、複数の計算済み畳み込み層の出力値から有用な値のみを伝搬させることができると考えられる。

本稿では、まず、DenseResNet の構成を述べ、Gate-Module の計算方法について説明する。また、DenseResNet が ResNet と同様の残差構造になっており、勾配消失問題

[1] 東北学院大学教養学部情報科学科
[a)] takeda@cs.tohoku-gakuin.ac.jp

や勾配爆発問題を解決していることを示す。さらに、画像分類のためのデータセットである ImageNet-1K [8] と CIFAR-100 [5] を用いた実験結果を示し、DenseResNet と従来手法の画像分類性能を比較する。これらの実験結果より、DenseResNet の画像分類性能が従来手法よりも優れていることを示す。

2. Dense Residual Network

図 1 に従来手法である ResNet と DenseNet、及び、提案手法である DenseResNet の基本構成を示す。ResNet は ResBlock と呼ばれる残差構造をつなげる構成となっており、i 番目の ResBlock で実行される計算処理は

$$\mathbf{x}_i = f_i(\mathbf{x}_{i-1}) + \mathbf{x}_{i-1} \tag{1}$$

と定義される。ここで、$\mathbf{x_i}$ は i 番目の ResBlock の出力であり、$f_i(\cdot)$ は i 番目の ResBlock に含まれる畳み込み層や活性化関数によって実行される計算を示す。ここで、1 番目から i 番目までの ResBlock に含まれる畳み込み層によって実行される計算を $F_i(\cdot)$ とすると、1 番目から i 番目の ResBlock によって実行される計算処理は

$$\mathbf{x}_i = F_i(\mathbf{x}_0) + \mathbf{x}_0 \tag{2}$$

となる。ここで、$\mathbf{x_0}$ は 1 番目の ResBlock への入力である。(2) 式より、ResNet では、ResBlock の数が多くなったとしても、入力値に畳み込み層の計算結果を足し合わせたものが ResBlock の出力値となることを示している。また、逆誤差伝搬を行う場合、出力側の誤差情報が入力側に直接伝播することも示している。この仕組により、ResNet は勾配消失や勾配爆発の問題を回避していると考えられる。

Dense Residual Network（DenseResNet）は、上記の ResNet の仕組みに対して DenseNet の考えを導入したものである。ResNet と同様に、DenseResNet は DenseResBlock をつなげる構成となっており、i 番目の DenseResBlock で実行される計算処理は

$$\mathbf{x}_i = \mathbf{v}_i \odot f_i(\mathbf{x}_{i-1}) + \sum_{k=i-N}^{i-1} \mathbf{w}_{ik} \odot \mathbf{x}_k \tag{3}$$

とする。ここで、$\mathbf{v}_i$ はこの DenseResBlock の出力への重み行列であり、$\mathbf{w}_{ik}$ は前の DenseResBlock の出力への重み行列である。また、N は入力として受け取る前の DenseResBlock の出力の数であり、$\odot$ は要素ごとの積算であり、$\sum_k \mathbf{w}_{ik} = J$（$J$ はすべての要素が 1 の行列）となる。

従来の ResNet と同様に、DenseResNet においても 1 番目から i 番目までの DenseResBlock によって実行される計算は式 (2) となる。つまり、DenseResBlock の数が多くなったとしても、入力値に畳み込み層の計算結果を足し合わせたものが DenseResNet の出力値となる。そのため、誤差伝搬では出力側の誤差情報が入力側に直接伝搬し、勾配消失や勾配爆発の問題を回避できる。

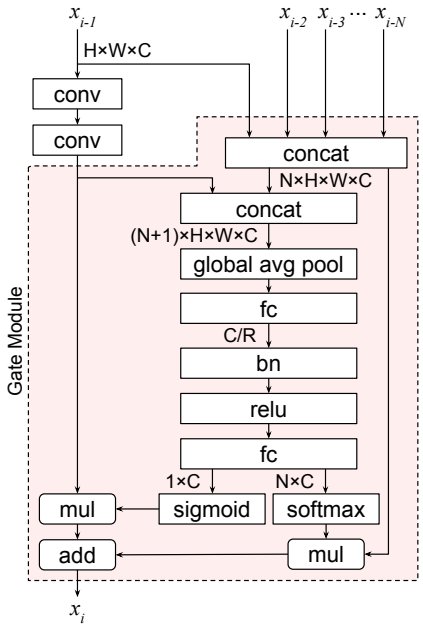

図 2 DenseResBlock の構成

Proof. 式 (3) より、1 番目の DenseResBlock の出力 $\mathbf{x_1}$ が式 (2) となるのは自明である。また、$i-1$ 番目の DenseResBlock の出力 $\mathbf{x_{i-1}}$ が式 (2) となるとき、i 番目の DenseResBlock で実行される計算処理は

$$\begin{aligned}\mathbf{x}_i &= \mathbf{v}_i \odot f_i(F_{i-1}(\mathbf{x}_0) + \mathbf{x}_0) \\ &\quad + \sum_{k=i-N}^{i-1} \mathbf{w}_{ik} \odot (F_k(\mathbf{x}_0) + \mathbf{x}_0)\end{aligned} \tag{4}$$

ここで、1 番目から i 番目までの DenseResBlock の畳み込み層で実行される計算処理を $F_i(\cdot)$ としてまとめると

$$\begin{aligned}\mathbf{x}_i &= F_i(\mathbf{x}_0) + \sum_{k=i-N}^{i-1} \mathbf{w}_{ik} \odot \mathbf{x}_0 \\ &= F_i(\mathbf{x}_0) + \mathbf{x}_0\end{aligned} \tag{5}$$

となる。以上より、1 番目から i 番目までの DenseResBlock によって実行される計算処理は式 (2) となる。 □

図 2 に DenseResBlock の構成を示す。DenseResBlock では、DenseResBlock の出力に対して適切な重みをかけるために Gate-Module を導入している。Gate-Module では、その DenseResBlock の畳み込み層の出力と前の DenseResBlock の出力 $\mathbf{x}_i$ から、この DenseResBlock の出力への重み $\mathbf{v}_i$ と前の DenseResBlock の出力への重み $\mathbf{w}_{ik}$ を求める。ただし、$\mathbf{x}_i$ と同じ大きさの重み行列 $\mathbf{v}_i$ や $\mathbf{w}_{ik}$ を求める場合、この大きさの行列に対する畳み込み計算が必要となるため、この計算に必要となる計算量とメモリ消費量が大幅に増加する。そこで、SENet [3] で提案されている SE-Module と同様に、Gate-Module では Global Average Pooling を用いて $\mathbf{x}_i$ を集約した特徴量ベクトルから重み行

列 $\mathbf{v}_i$ や $\mathbf{w}_{ik}$ を求めることにより計算量とメモリ消費量を削減する。また、Gate-Module では、重み行列 $\mathbf{v}_i$ への活性化関数としては sigmoid を用い、重み行列 $\mathbf{w}_{ik}$ への活性化関数としては softmax を用いる。これにより、$\sum_k \mathbf{w}_{ik} = J$ となるように正規化を行っている。

3. 実装と実験

3.1 実装

DenseResNet の性能を評価するため、深層学習フレームワークである PyTorch を用いて DenseResNet を実装した。DenseResNet の基本的な構造は ResNet [1] と同じであるが、DenseResNet は複数の DenseResBlock の出力を加算するために Gate-Module を有する点が異なる。この DenseResNet の実装を用いて、標準的な画像分類データセットある ImageNet-1K [8] と CIFAR-100 [5] における DenseResNet の画像分類精度を検証した。さらに、Gate-Module の有効性を検証するため、従来手法である MobileNet-v2 [9] に対して Gate-Module を導入した DenseMobileNet-v2 の画像分類精度を検証した。

ImageNet-1K は 120 万枚の学習画像と 5 万枚のテスト画像からなる大規模画像データセットであり、それぞれの画像は 1000 個のカテゴリに分類されている。ImageNet-1K の学習処理では、従来手法 [1, 4, 10] と同様に解像度変更・切り取り・反転などの標準的なデータ拡張を行った。また、バッチサイズは 64 とし、パラメータの更新は Nesterov SGD とし、学習係数は Cosine Annealing に従って変化させた。ResNet-34 と DenseResNet-34 の学習では、入力画像は 160x160 ピクセル、初期学習係数は 0.025、モーメンタムは 0.9、正則化係数は 1e-4 とした。一方、MobileNet-v2 と DenseMobileNet-v2 の学習では、入力画像は 224x224 ピクセル、初期学習係数は 0.0125、モーメンタムは 0.9、正則化係数は 4e-5 とした。ImageNet-1K の分類精度のテストでは、画像の短辺の長さを 256 ピクセルとなるように縮小した画像から中央部分を切り取りとった 224x224 ピクセルの画像をテスト用の入力画像（single crop test）とした。

CIFAR-100 は 5 万枚の学習画像と 1 万枚のテスト画像からなる小規模画像データセットであり、それぞれ 32x32 ピクセルの解像度の画像が 100 個のカテゴリに分類されている。CIFAR-100 の学習処理においても、従来手法 [1, 4] と同様に切り取り・反転の標準的なデータ拡張を行った。また、パラメータの更新は Nesterov SGD で行い、初期学習係数は 0.05、モーメンタムは 0.9、正則化係数は 5e-4、バッチサイズは 64 とし、学習係数は Cosine Annealing に従って変化させた。

3.2 実験結果

図 3 に ImageNet-1K の学習を行ったときのテストデータに対する分類エラー率の変化を示し、表 1 に最終的な画像分類精度の比較を示す。この実験では、DenseResNet-34 は従来手法である ResNet-34 よりも高い画像分類精度を達成し、ハイパーパラメータである N を 6 に設定した DenseResNet-34 の画像分類のエラー率は 23.34%となった。一方、Gate-Module の導入によるパラメータ数と計算量の増加は限定的である。N を 6 に設定した DenseResNet-34 の場合、Gate-Module 導入によるパラメータ増加数は約 2.20M であり、計算量の増加量は約 16.0M flops であった。また、DenseMobileNet-v2 も MobileNet-v2 よりも高い画像分類精度を示したが、このときのパラメータ数と計算量の増加量は限定的であった。以上の実験結果は、Gate-Module を導入することにより、少量のパラメータ数と計算量の増加のみで画像分類性能を改善することを示している。

表 1 ImageNet-1K のテストデータに対する分類エラー率の比較

model	# params	flops	top-1 err
ResNet-34 (baseline)	21.80 M	3.68 G	24.99 %
DenseResNet-34 N=2	22.75 M	3.69 G	23.80 %
DenseResNet-34 N=4	23.38 M	3.69 G	23.60 %
DenseResNet-34 N=6	24.00 M	3.70 G	**23.34** %
DenseResNet-34 N=8	24.60 M	3.70 G	23.43 %
MovileNet-v2 (baseline)	3.50 M	315 M	27.70 %
DenseMovileNet-v2 N=4	4.08 M	320 M	**26.60** %

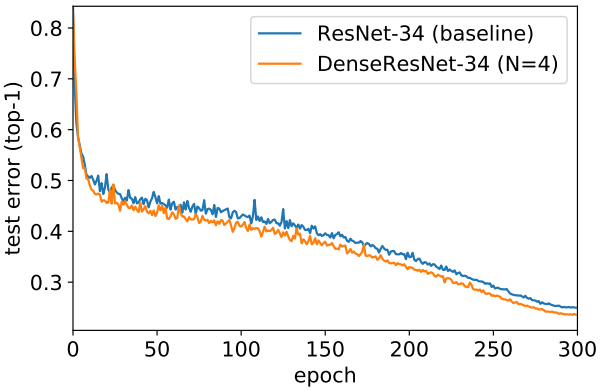

図 3 ImageNet-1K のテストデータに対する分類エラー率の変化

図 4 に、ImageNet-1K のテストデータの分類を行ったときの DenseResNet-34 の重み行列 $\mathbf{w}_{ik}$ の要素の平均値を示す。直前の DenseResBlock の出力に対する重みが小さくなっている部分が複数箇所あり、出力に最も近い DenseResBlock では 3 個前の DenseResBlock の出力に対する重みが最も大きくなった。この実験結果は、DenseResNet が ResNet とは異なる動作をしていることを示しており、この変化により画像分類性能が改善したと考えられる。

図 5 に CIFAR-100 の学習を行ったときのテストデータに対する分類エラー率の変化を示し、表 2 に最終的な画像分類精度の比較を示す。この実験においても、DenseResNet-101 は従来手法である ResNet-101 よりも高い画像分類精度を達成し、ハイパーパラメータである N を 4 に設定した

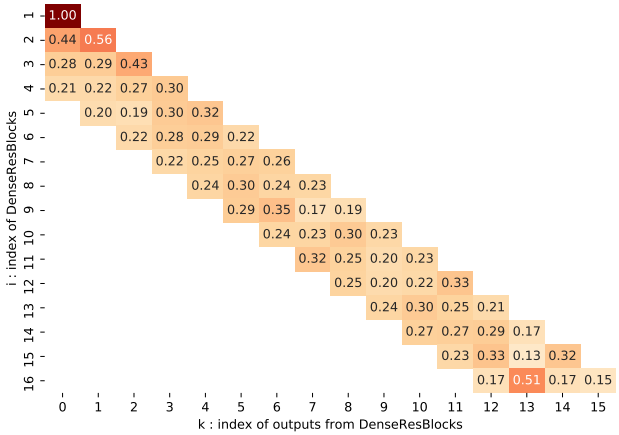

図 4 DenseResNet34 (N=4) の重み行列 $\mathbf{w}_{ik}$ の要素の平均

表 2 CIFAR-100 のテストデータに対する分類エラー率の比較

model	# params	flops	top-1 err
ResNet-110 (baseline)	1.74 M	258 M	24.65 %
DenseResNet-110 N=2	2.03 M	261 M	23.03 %
DenseResNet-110 N=4	2.23 M	263 M	**22.62** %
DenseResNet-110 N=6	2.43 M	265 M	23.29 %
DenseResNet-110 N=8	2.62 M	267 M	22.95 %

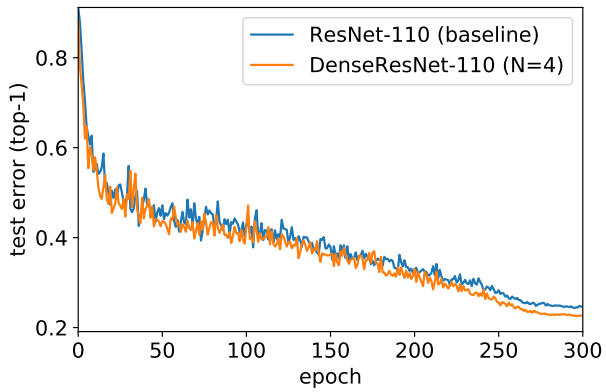

図 5 CIFAR-100 のテストデータに対する分類エラー率の変化

DenseResNet-101 の画像分類のエラー率は 22.62%となった。この実験結果は、小規模データセットである CIFAR-100 においても、DenseResNet は少量のパラメータ数と計算量の増加のみで画像分類性能を改善することを示している。

4. まとめ

本稿では、ResNet に対して DenseNet のネットワーク構造を導入した画像分類のための CNN である DenseResNet を提案した。DenseResNet では、直接的な特徴量と抽象化された特徴量から必要なものを選択しながら画像を分類するため、それぞれの DenseResBlock において以前に計算を行った複数の DenseResBlock の出力を統合する仕組みを導入している。また、新たに導入した Gate-Module では、複数の DenseResBlock からの出力の信号強度を調整することで、勾配消失問題や勾配爆発問題を回避している。本稿では、画像分類データセットである ImageNet-1K と CIFAR-100 を用いて DenseResNet の画像分類精度を検証した。実験結果より、従来手法である ResNet に対して、DenseResNet が少量のパラメータ数の増加のみで画像分類性能を改善していることを確認した。

参考文献

[1] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778 (2016).

[2] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. et al.: Searching for MobileNetv3, *Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV)*, pp. 1314–1324 (2019).

[3] Hu, J., Shen, L. and Sun, G.: Squeeze-and-Excitation Networks, *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141 (2018).

[4] Huang, G., Liu, Z., Weinberger, K. Q. and van der Maaten, L.: Densely connected convolutional networks, *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269 (2017).

[5] Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images, *Tech Report* (2009).

[6] Lee, Y., Hwang, J.-w., Lee, S., Bae, Y. and Park, J.: An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection, *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2019).

[7] Real, E., Aggarwal, A., Huang, Y. and Le, Q. V.: Regularized Evolution for Image Classifier Architecture Search, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 1, pp. 4780–4789 (2019).

[8] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al.: Imagenet large scale visual recognition challenge, *International Journal of Computer Vision*, Vol. 115, No. 3, pp. 211–252 (2015).

[9] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.: MobileNetV2: Inverted Residuals and Linear Bottlenecks, *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520 (2018).

[10] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A.: Going Deeper with Convolutions, *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9 (2015).

[11] Tan, M. and Le, Q. V.: Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6105–6114 (2019).

[12] Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K.: Aggregated Residual Transformations for Deep Neural Networks, *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995 (2017).