ISSN 2186-5647

2-26

ニューラルネットワークにおける 非線形活性化関数の提案及びその評価

日大生産工(院)　○榎本　知幸　　日大生産工　角田　和彦
日大生産工　三浦　慎一郎

1. はじめに

画像認識の分野において、ニューラルネットワークを用いた人工知能での認識能力は高く、特に畳み込みニューラルネットワーク（CNN）[1]を用いた画像認識能力は、非常に精度が高い。

そのCNNの認識能力をさらに高める手法の一つに、活性化関数の最適化が挙げられる。CNNにおける畳み込み層の活性化関数として、現在、ReLU関数[2]を始めとした様々な活性化関数が提案され、ネットワークに組み込まれているが、どの活性化関数がどのネットワークにおいて最適であるかは分かっていないことが多い。

本研究では、二つの活性化関数の特徴を合わせた新しい活性化関数を提案し、ネットワークに適用することで他の関数と比較する。さらに、その結果から関数の特徴と学習の向上の関係について明らかにすることを目的としている。

2. ニューラルネットワークのおける活性化関数

現在、画像認識おける CNN に組み込む関数として、ReLU 関数が幅広く使われている。この関数は入力値 v が入力されると、max(0, v)を出力値として返す。ReLU 関数は勾配消失問題を軽減することが出来るという特徴があり、畳み込み層の活性化関数に適用することでシグモイド関数や tanh 関数よりも良い結果が得られている。

しかし、v が 0 以下の時に勾配が無くなることによってユニットが死んでしまうという問題点も挙げられている。

そこで、ReLU関数の負の領域を拡張した、PReLU関数[3]が提案されている。PReLU関数は次のように示される。

$$PReLU(v) = \begin{cases} v & (v > 0) \\ av & (v \leq 0) \end{cases} \quad (1)$$

ただし、パラメータaは逆伝搬により学習し、活性化関数は最適化される。

PReLU 関数は、層ごとに最適な関数に学習することで、ReLU 関数よりも優れた結果を残した例も報告されている。また、負の値をとることで、ユニットが死んでしまう問題点も改善されている。しかし、その一方で負領域が線形で飽和しないため、ノイズの影響を強く受けてしまうといった問題点も指摘されている。

また、ReLU 関数の負領域を非線形的に拡張した ELU 関数[4]も提案されており、次のように示される。

$$ELU(v) = \begin{cases} v & (v > 0) \\ \alpha(e^{v} - 1) & (v \leq 0) \end{cases} \quad (2)$$

しかし、ReLUの持つスパース性は失われており、場合によってはReLUよりも精度が下がってしまうことがある。

3. 新しい活性化関数の提案

PReLU関数は活性化関数に学習するパラメータを増やすことでネットワークの表現力を上げた。また、ELU関数は負領域を非線形に拡張することでReLUの持ついくつかの問題点を解決している。

ここで、これらの二つの関数に注目し、双方の特徴を持った新しい活性化関数を提案する。

文献[5]ではこのような活性化関数が示されている。

$$h(v) = \begin{cases} \frac{1}{2}\left(2 - \frac{1}{1+|\gamma|}\right) & (v > 0) \\ \frac{1}{2}\left(\frac{1}{1+|\gamma|}\right) & (v \leq 0) \end{cases} \quad (3)$$

ただし、$\gamma =$ v/2k, $k > 0$。

この関数は、ネットワーク温度kを用いて、ステップ関数からシグモイド関数を近似する

Proposal of Nonlinear Activation Function in Neural Network and Its Evaluation

Tomoyuki ENOMOTO, Kazuhiko KAKUDA and Shinichiro MIURA

ことができる。また、関数を積分することで、ReLU関数から線形関数まで非線形的に変化させることができる。

ここで、この特徴に注目し、目的の特徴を持った関数を導出していく。

まず、関数の負の領域だけに注目し、2kをパラメータaに置き換え簡略化する。

$$\hat{h}(v) = \frac{a}{a - v} \quad (v < 0) \tag{4}$$

次に、積分時に正の領域を常に線形にするため、入力に関わらず出力を1とする。また、このままではパラメータが指数表記になることや、ゼロ除算が発生してしまう可能性があるので、aを$exp(a)$に置き換える。さらに、積分時に式を簡略化するために二乗する。

$$\tilde{h}(v) = \begin{cases} 1 & (v \geq 0) \\ \left(\frac{e^a}{e^a - v}\right)^2 & (v < 0) \end{cases} \tag{5}$$

ここで、関数を積分することで次式が得られる。

$$\int \tilde{h}(v)dv = \begin{cases} v + C_1 & (v \geq 0) \\ \frac{e^{2a}}{e^a - v} + C_2 & (v < 0) \end{cases} \tag{6}$$

ただし、C_1, C_2は積分定数。

さらにv=0の場合常に出力が0になるようC_1, C_2を調整することで目的の特徴を持った関数を導出する。

$$f(v) = \begin{cases} v & (v \geq 0) \\ \frac{e^a v}{e^a - v} & (v < 0) \end{cases} \tag{7}$$

この活性化関数は、$a \rightarrow$ -∞の極限でReLU関数に収束し、$a \rightarrow$ +∞の極限で傾き 1 の線形関数に収束する（図1）。
また、パラメータaが 0 付近の時、ELU 関数に似た出力を得ることが出来る(図２)。

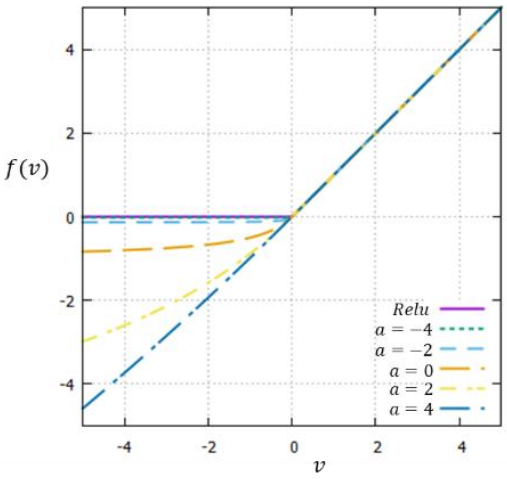

Fig 1. 本研究の活性化関数

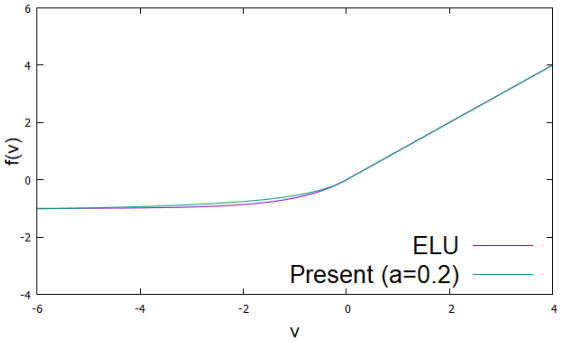

Fig 2. ELU 関数と本研究の関数

また、活性化関数におけるaによる偏微分は次のように示される。

$$\frac{\partial f(v)}{\partial a} = \begin{cases} 0 & (v \geq 0) \\ -\frac{e^a v^2}{(e^a - v)^2} & (v < 0) \end{cases} \tag{8}$$

誤差関数 E によりパラメータa_iは、次のように更新される。

$$\Delta a_i := \varepsilon \frac{\partial \mathrm{E}}{\partial a_i} \tag{9}$$

$$\frac{\partial E}{\partial a_i} = \sum_{v_i} \frac{\partial E}{\partial f(v_i)} \frac{\partial f(v_i)}{\partial a_i} \tag{10}$$

ただし、ϵは学習率、$\sum v_i$はa_iに対応する全てのv_iにおける合計を表す。

4. 数値計算と比較

ReLU関数、PReLU関数、ELU関数と本研究の活性化関数をCNNの畳み込み層に組み込み、一般物体データセットCifar-10を用いてそれぞれ学習させた。また、PReLU関数と本研究の活性化関数はチャネルごとにパラメータaを所有している。

次に、実験に用いるCNNの構造を図3に示す。このCNNはDrop out、Max pooling層、全結合層を除くと9つのレイヤーを有しており、それぞれのレイヤーは畳み込み層、バッチ正規化層、活性化関数で構成されている。入力する画像はランダムに反転、明るさ、コントラストを調整することで訓練画像のデータの水増しを行っている。

また、表1に各層でのカーネルのサイズ、チャネル数、パラメータの合計数を示す。

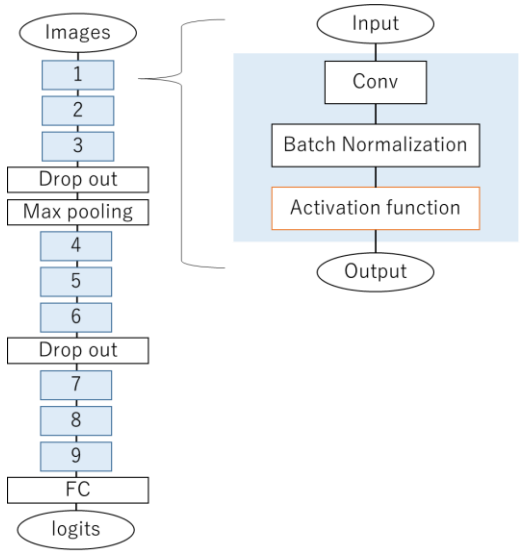

Fig 3. ネットワークの構造

表 1. ネットワークのパラメータ数

	kernel size	channel	parameter
1	1	64	64
2	3	64	576
3	5	64	1600
4	1	128	128
5	3	128	1152
6	5	128	3200
7	1	256	256
8	3	256	2304
9	5	256	6400
10(FC)		256	256
SUM			15936

ただし、PReLU 関数及び本研究の活性化関数ではパラメータ数がチャネル数だけ増加する。ミニバッチサイズは 100、学習アルゴリズムは Adam[6]、epoch 数は 200 とした。また、それぞれの関数の訓練データにおける損失の推移を図 4 に示す。次に、それぞれの関数の検証データにおける損失の推移を図 5 に示す。次に、ネットワーク中間の第 5 レイヤーにおける PReLU 関数と本研究の活性化関数のパラメータaの値の推移を図 6 、図 7 にそれぞれ示す。また、訓練後の最終的なネットワークでのテスト結果と、1step ごとの計算時間を表 2 に示す。最後に、計算開始から 3 時間経過した時点での訓練データでの損失を表 3 に示す。

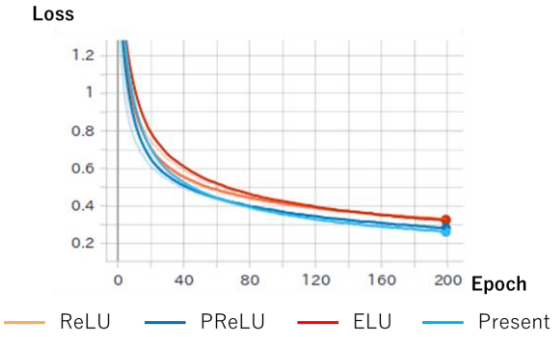

Fig 4. 訓練データでの損失

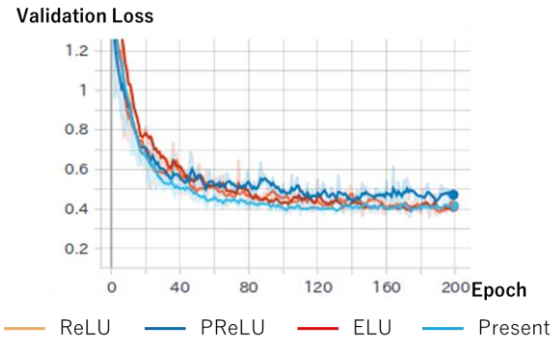

Fig 5. 検証データでの損失

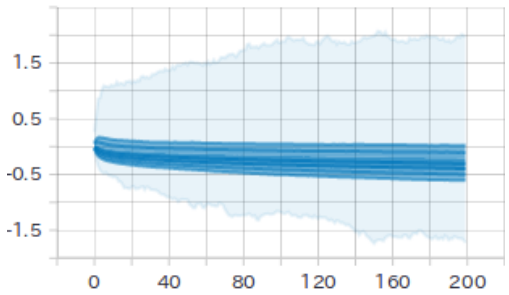

Fig 6. PReLU 関数のパラメータaの推移

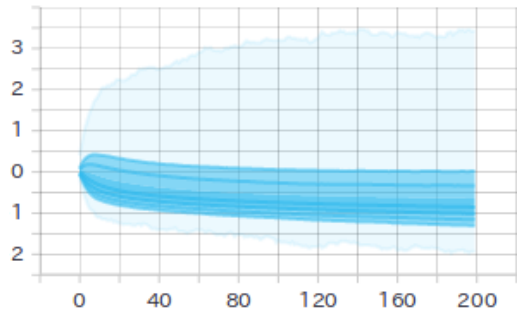

Fig 7. 本研究の関数のパラメータaの推移

表2. 損失、正解率と計算時間

	ReLU	ELU	PReLU	Present
Loss	0.355	0.316	0.280	0.258
Test_Loss	0.394	0.414	0.455	0.429
Accuracy(%)	88.7	88.8	90.0	90.9
Test_Acc(%)	87.5	87.6	86.9	88.0
Time(ms)	21	22	26	30

表3. 計算開始から3時間時点での損失

	ReLU	ELU	PReLU	Present
Loss	0.351	0.347	0.314	0.308

5. おわりに

本研究では、PReLU関数とELU関数が持つそれぞれの特徴に注目し、それを引き継いだ新しい活性化関数を提案した。今回の数値計算ではパラメータを導入したことによってネットワークの表現力をPReLUと同等かそれ以上に高めつつ、未知のデータに対しても強い特性が得られた。また、計算時間は今回比較した関数の中では最も長かったが、単位時間ごとの学習効率においてはむしろ良い結果が得られた。

今後はさらに複雑なネットワークでも学習を行い、活性化関数についてより詳しく調べていきたい。

参考文献

1) Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, Proc. of the IEEE,(1998), pp.2278-2324.
2) V. Nair and G. E. Hinton, “Rectified Linear Units Improverestricted Boltzmann Machines”, In ICML, (2010), pp.807-814.
3) K.He, X.Zhang, S.Ren, J.Sun. “Delving Deep into Rectifiers”, arXiv:1502.01852v1 [cs.CV] 6 Feb 2015.
4) D.Clevert, T.Unterthiner, S.Hochreiter, “Fast And Accurate Deep Network Learning by Exponential Linear Units (ELUs)”, arXiv:1511.07289v5 [cs.LG] 22 Feb 2016. 1) Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, Proc. of the IEEE,(1998), pp.2278-2324.
5) 角田和彦, “ニューラルネットの特性関数とその近似関数”, 情報処理学会第 67 回全国大会, (2005), pp.239-240.
6) Diederik P. Kingma, Jimmy Lei Ba “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”, arXiv:1412.6980v9 [cs.LG] 30 Jan 2017