Received: 23 August 2017 | Revised: 10 December 2017 | Accepted: 11 December 2017

DOI: 10.1002/cnm.2956

WILEY

Generalized backpropagation algorithm for training second-order neural networks

Fenglei Fan | Wenxiang Cong | Ge Wang

Biomedical Imaging Center, BME/CBIS, Rensselaer Polytechnic Institute, Troy, NY, USA

Correspondence
Ge Wang, Biomedical Imaging Center, BME/CBIS, Rensselaer Polytechnic Institute, Troy, New York, USA.
Email: wangg6@rpi.edu

Abstract

The artificial neural network is a popular framework in machine learning. To empower individual neurons, we recently suggested that the current type of neurons could be upgraded to second-order counterparts, in which the linear operation between inputs to a neuron and the associated weights is replaced with a nonlinear quadratic operation. A single second-order neurons already have a strong nonlinear modeling ability, such as implementing basic fuzzy logic operations. In this paper, we develop a general backpropagation algorithm to train the network consisting of second-order neurons. The numerical studies are performed to verify the generalized backpropagation algorithm.

KEYWORDS

artificial neural network, second-order neurons, backpropagation (BP)

1 | INTRODUCTION

In machine learning, artificial neural networks (ANNs) especially deep neural networks (convolutional neural network [CNN]) have been very successful in different applications including multiple areas in biomedical engineering such as medical imaging.[1-3] Usually, a neural network has several layers of neurons. For each neuron, the inner product between an input vector and its weighting vector is nonlinearly processed by an activation function such as Sigmoid or Rectifier.[4-7] The model of the current neuron, by definition, cannot perform nonlinear classification unless a number of neurons are connected to form a network.

To enrich the armory of neural networks at the cellular level, recently, we upgraded inner-product–based (first-order) neurons to quadratic-function–based (second-order) neurons by replacing the inner product with a quadratic function of the input vector. In our feasibility study,[8] a single second-order neuron performed effectively in linearly inseparable classification tasks; for example, basic fuzzy logic operations. That is to say, the second-order neuron has a representation capability superior to that of the first-order neuron.

While either first- or second-order neurons can be interconnected to approximate any functions,[9] we hypothesize that to perform a complicated task of machine learning the network made of second-order neurons could be significantly simpler than that with first-order neurons. To test this hypothesis, the training algorithm, known as backpropagation (BP), for traditional first-order neural networks needs to be extended to handle second-order neural networks, although the principle of BP is essentially the same.

As we know, the key idea behind the training of ANNs is to treat it as an optimization problem.[10] The network learns the features when the optimization is accomplished with respect to the training dataset. In the optimization process, the propagation and BP steps are repeated for gradual refinement until convergence. While the propagation produces an

This work is partially supported by the Clark & Crossan Endowment Fund at Rensselaer Polytechnic Institute, Troy, NY, USA.

error of a loss function, the BP calculates the gradients of the loss function with respect to weights and minimizes the loss function by adjusting the weights. In doing so, the forward and backward steps are performed layer-wise according to the chain rule for differentiation.

In the next section, we formulate a general BP algorithm for training of second-order networks. In the third section, we present several numerical examples to show the feasibility and merits of second-order networks relative to that with first-order neurons. Finally, we discuss relevant issues and conclude the paper.

2 | BACKPROPAGATION THROUGH THE SECOND-ORDER NETWORK

The structure of the second-order neuron is shown in Figure 1, where $w_{0\_r} := b_1$, $w_{0\_g} := b_2$, and $x_0 = 0$. The quadratic transform of the input vector is defined as follows[8]:

$$h(\vec{x}) = \left(\sum_{i=1}^{n} w_{i\_r} x_i + b_r\right)\left(\sum_{i=1}^{n} w_{i\_g} x_i + b_g\right) + \sum_{i=1}^{n} w_{i\_b} x_i^2 + c. \tag{1}$$

Then $h(\vec{x})$ will be further processed by a nonlinear activation function, such as a Sigmoid or ReLU function. That is, in a second-order neuron, the input vector is quadratically, instead of linearly, mapped before being fed into the excitation function. The second-order neuron is a natural extension of the first-order neuron. It is expected that the second-order networks should be more efficient than the first-order counterparts in some important applications, for example, fuzzy logic operations, as explained before.[8]

Multilayer feedforward networks,[11] with at least 3 layers but neither shortcut nor closed loop, are quite common ANNs. Therefore, we choose this architecture as the target for extension of the traditional BP algorithm to the second-order networks.

Specifically, the second-order BP algorithm can be formulated as follows. We assume that the activation function for every neuron is the sigmoid function $\sigma(x) = \dfrac{1}{1 + \exp(-x)}$; the network has m inputs denoted as $\left\{\vec{x_i} \in R^m\right\}$ and n outputs denoted as $\left\{\vec{y_i} \in R^n\right\}$. With $\vec{y'}$ as the final output of the feedforward propagation process, the loss function is expressed as

$$E\left(\vec{w}, \vec{b}, c\right) := \frac{1}{2}\sum_i \left(y_i' - y_i\right)^2 = \frac{1}{2}\sum_i \left(h_{\vec{w}, \vec{b}, c}\left(\vec{x_i}\right) - y_i\right)^2, \tag{2}$$

which is proportional to the Euclidean distance between the vectors $\vec{y'}$ and $\vec{y}$. For a neuron at the lth layer with p input variables, its output o_k^l is an input to the $(l + 1)$th layer and can be computed by

$$o_j^{l+1} = \sigma\left(\text{net}_j\right) = \sigma\left(\left(\sum_{k=1}^{p} w_{kj\_r} o_k^l + b_{j\_r}\right)\left(\sum_{k=1}^{p} w_{kj\_g} o_k^l + b_{j\_g}\right) + \sum_{k=1}^{p} w_{kj\_b}\left(o_k^l\right)^2 + c_j\right). \tag{3}$$

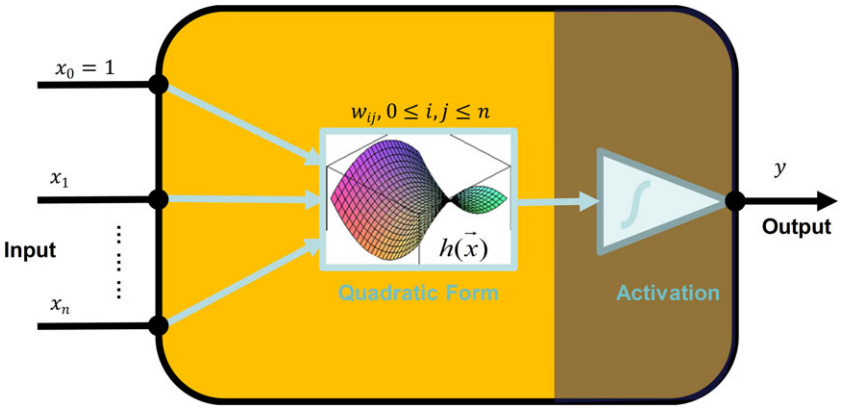

FIGURE 1 The structure of our proposed second-order neuron consisting of a quadratic function and an activation function, both of which are nonlinear

where the weights w_{kj_r}, w_{kj_g}, b_{j_r}, b_{j_g}, and c_j are associated with the kth neuron in the $(l-1)$th layer and the jth neuron in the lth layer. The value of net_j for activation by the neuron j is a quadratic operation of $\overrightarrow{o^l}$, the outputs of p neurons at the previous layer. For the neurons in the input layer, the inputs are our original data.

The goal of training the network is to adjust parameters for a minimized E. The optimal parameters can be found using the gradient descent method starting from initial values. The derivative of the sigmoid function has a simple form:

$$\frac{d\sigma}{dz}(z) = \sigma(z)(1-\sigma(z)). \tag{4}$$

First of all, we separate the contribution of individual training sample:

$$\begin{cases} \dfrac{\partial E}{\partial \overrightarrow{w}} = \sum\limits_i \dfrac{\partial E_i}{\partial \overrightarrow{w}} \\ \dfrac{\partial E}{\partial \overrightarrow{b}} = \sum\limits_i \dfrac{\partial E_i}{\partial \overrightarrow{b}} \\ \dfrac{\partial E}{\partial c} = \sum\limits_i \dfrac{\partial E_i}{\partial c} \end{cases}. \tag{5}$$

The partial derivatives of the involved weights and biased with respect to the loss function can be computed by the chain rule:

$$\begin{cases} \dfrac{\partial E_i}{\partial w_{kj\_r}} = \dfrac{\partial E_i}{\partial o_j^l}\dfrac{\partial o_j^l}{\partial \text{net}_j}\dfrac{\partial \text{net}_j}{\partial w_{kj\_r}} \\ \dfrac{\partial E_i}{\partial w_{kj\_g}} = \dfrac{\partial E_i}{\partial o_j^l}\dfrac{\partial o_j^l}{\partial \text{net}_j^l}\dfrac{\partial \text{net}_j^l}{w_{kj\_g}} \\ \dfrac{\partial E_i}{w_{kj\_b}} = \dfrac{\partial E_i}{\partial o_j^l}\dfrac{\partial o_j^l}{\partial \text{net}_j^l}\dfrac{\partial \text{net}_j^l}{w_{kj\_b}} \\ \dfrac{\partial E_i}{\partial b_{j\_r}} = \dfrac{\partial E_i}{\partial o_j^l}\dfrac{\partial o_j^l}{\partial \text{net}_j^l}\dfrac{\partial \text{net}_j^l}{\partial b_{j\_r}} \\ \dfrac{\partial E_i}{\partial b_{j\_g}} = \dfrac{\partial E_i}{\partial o_j^l}\dfrac{\partial o_j^l}{\partial \text{net}_j^l}\dfrac{\partial \text{net}_j^l}{\partial b_{j\_g}} \\ \dfrac{\partial E_i}{\partial c_j} = \dfrac{\partial E_i}{\partial o_j^l}\dfrac{\partial o_j^l}{\partial \text{net}_j^l} \end{cases}. \tag{6}$$

If j is the output node, that is, the l^{th} layer is actually the output layer, then $o_j^l = h_{\underset{w,b,c}{\rightarrow\rightarrow}}(\overrightarrow{x_i})$,

$$\frac{\partial E_i}{\partial o_j^l} = o_j^l - y_i. \tag{7}$$

If j is not the output node, then the change of o_j^l will affect all the neurons connected to it in the $(l+1)$th layer. Assuming L is the set of indices for such neurons, the first factor becomes

$$\begin{aligned} \frac{\partial E_i}{\partial o_j^l} &= \sum_{v\in L}\left(\frac{\partial E_i}{\partial \text{net}_v^{l+1}}\frac{\partial \text{net}_v^{l+1}}{\partial o_j^l}\right) \\ &= \sum_{v\in L}\left(\frac{\partial E_i}{\partial o_v^{l+1}}\frac{\partial o_v^{l+1}}{\partial \text{net}_v^{l+1}}\frac{\partial \text{net}_v^{l+1}}{\partial o_j^l}\right). \end{aligned} \tag{8}$$

For the second-order neurons, we have

$$\frac{\partial \text{net}_v^{l+1}}{\partial o_j^l} = w_{jv\_r}\left(\sum_{j=1}^{n} w_{jv\_g}\, o_j^l + b_{j\_g}\right) + w_{jv\_g}\left(\sum_{j=1}^{n} w_{jv\_r}\, o_j^l + b_{j\_r}\right) + 2\sum_{j=1}^{n} w_{jv\_b}\, o_j^l, \tag{9}$$

where n is the number of the neurons in the lth layer. One comment is that Equations 7 to 9 together allow to calculate arbitrary partial derivatives $\frac{\partial E_i}{\partial o_j^l}$ because applying these partial differentiation steps repeatedly will finally reach the input layer.

Furthermore, the second factors in Equation 6 and the factors in Equation 8 are calculated as

$$\frac{\partial o_j^l}{\partial \text{net}_j^l} = \sigma\left(\text{net}_j^l\right)\left(1-\sigma\left(\text{net}_j^l\right)\right) = o_j^l\left(1-o_j^l\right). \tag{10}$$

Finally, the third factor in Equation 6 is computed as

$$\begin{cases} \dfrac{\partial \text{net}_j^l}{\partial w_{kj\_r}} = o_k^{l-1}\left(\sum\limits_{i=1}^{m} w_{ij\_g}\, o_i^l + b_{j\_r}\right) \\ \dfrac{\partial \text{net}_j^l}{\partial w_{kj\_g}} = o_k^{l-1}\left(\sum\limits_{i=1}^{m} w_{ij\_r}\, o_i^l + b_{j\_g}\right) \\ \dfrac{\partial \text{net}_j^l}{\partial w_{kj\_b}} = \left(o_k^{l-1}\right)^2 \\ \dfrac{\partial \text{net}_j^l}{\partial b_{j\_r}} = \sum\limits_{i=1}^{m} w_{ij\_g}\, o_i^l + b_{j\_g} \\ \dfrac{\partial \text{net}_j^l}{\partial b_{j\_g}} = \sum\limits_{i=1}^{m} w_{ij\_r}\, o_i^l + b_{j\_r} \end{cases}, \tag{11}$$

where m is the number of the neurons in the lth layer.

From Equations 5 to 11, all the partial derivatives can be computed to implement the gradient-based optimization. Typically, the steepest descent algorithm in the following form is used:

$$\chi^{(n)} = \chi^{(n-1)} - \eta \cdot \frac{\partial E}{\partial \chi^{(n-1)}}, \tag{12}$$

where $\chi^{(n)}$ is a generic parameter in the nth iteration, and η is the step size. The optimization flowchart is in Figure 2.

3 | NUMERICAL RESULTS

To evaluate the power of second-order networks, we started with testing the network architecture with 2 hidden layers. For illustration, the BP process can be more clearly illustrated with the double hidden layer network than with a single hidden layer network. Pilot study suggests that the computational power of the double hidden layer network is sufficiently strong.

For visualization, the training datasets contained vectors of only 2 elements for the first 2 examples. The color map "cool" in MATLAB was used to show the functional values. The symbols "o" and "+" are labels for 2 classes.

3.1 | "Telling-Two-Spirals-Apart" problem

The "Telling-Two-Spirals-Apart" problem is a classical neural network benchmark test proposed by Wieland.[12] There are 194 instances on the 2 spiral arms, with 32 points per turn plus an end point in the center for each spiral. The training points were generated with the following equations.[13]

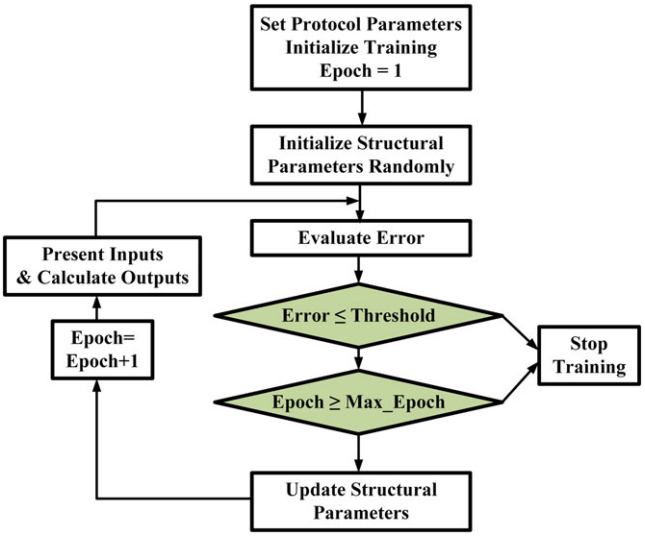

FIGURE 2 Flowchart of the general backpropagation algorithm to update structural parameters (weights including offsets)

$$x^{2n-1} = r_n \sin\alpha_n + 0.5 = 1 - x^{2n} \tag{13}$$

$$x^{2n} = 0.5 - r_n \sin\alpha_n = 1 - x^{2n-1} \tag{14}$$

$$y^{2n-1} = r_n \cos\alpha_n + 0.5 = 1 - y^{2n} \tag{15}$$

$$y^{2n} = 0.5 - r_n \cos\alpha_n = 1 - y^{2n-1} \tag{16}$$

$$b^{2n-1} = 1 \tag{17}$$

$$b^{2n} = 0 \tag{18}$$

$$r_n = 0.4\left(\frac{105-n}{104}\right) \tag{19}$$

$$\alpha_n = \frac{\pi(n-1)}{16} \tag{20}$$

Lang and Witbrock reported that the standard BP network with forward connections cannot classify such spirals. Then they made a 2-5-5-5-1 network architecture with shortcut to solve this problem successfully. They trained the network with the BP algorithm and *QuickProp*.[14]

In our experiment, however, we constructed the second-order network of the 2-20-20-1 configuration without any shortcut and trained the network with our generalized BP algorithm in a divide-and-conquer fashion (step by step, from the outer turn toward inner turn). The result is color-mapped in Figure 3, showing the superior inherent power of the second-order network.

3.2 | Separating concentric rings

Another type of natural patterns is concentric rings. As a test, we generated 4 concentric rings, which were respectively assigned to 2 classes. There are 60 instances per ring in total of 240 instances. With randomly selected initial weights and

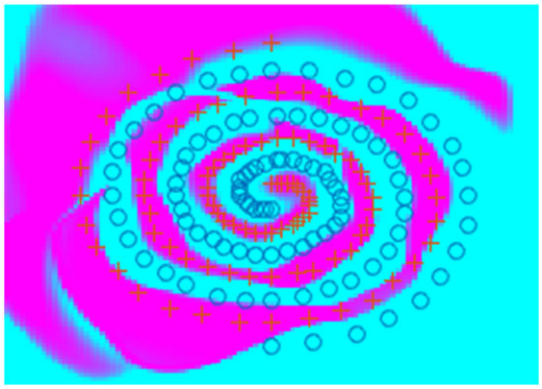

FIGURE 3 Classification of 2 spirals with the second-order network of the 2-20-20-1 configuration

biases using the *rands* function in MATLAB, the second-order network of a 2-3-2-1 configuration was trained, perfectly separating the 2 classes of instances in no more than 1000 iterations, as shown in Figure 4. In contrast, we also trained the first-order networks. Even if we increased the complexity of the first-order network to a 2-5-8-1 configuration, the circles could still not be classified well. It is seen in Figure 5 that the purple ring is uneven with a defect zone at the bottom of the purple ring. Furthermore, to compare 2 networks statistically, we independently repeated the training procedure by 10 times and counted the total number of correct predictions each time. For the second-order network of the 2-3-2-1 configuration, the correct outcome after each training session remained 240 of 240 testing cases, while the counterparts from the first-order network of the 2-5-8-1 configuration were 239, 240, 218, 223, 240, 220, 239, 240, 240, and 234. The simpler structure of the second-order network means a higher computation efficiency than the first-order counterpart.

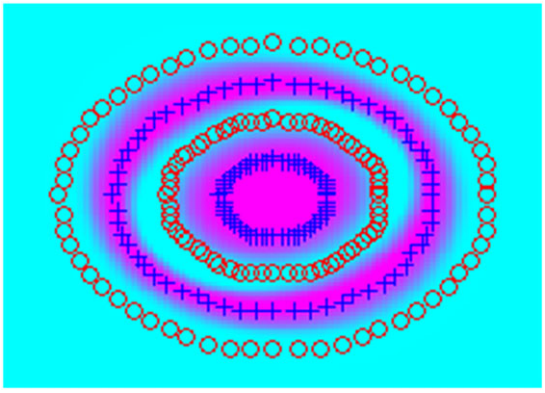

FIGURE 4 Perfect classification of concentric rings by the second-order network of the 2-3-2-1 configuration

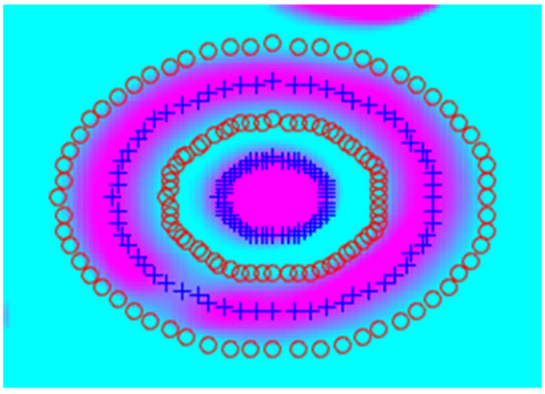

FIGURE 5 Imperfect classification of concentric rings by the first-order network of the 2-5-8-1 configuration

3.3 | Sorting CT and MRI Images

Computed tomography (CT) and magnetic resonance imaging (MRI) have many clinical applications. While both CT and MRI tomographically depict the structure and function of the patient, they have relative merits and drawbacks. CT enjoys high spatial resolution, little geometric distortion, fast speed, and cost-effectiveness. MRI is featured by high contrast resolution, rich functional information, pulse programming flexibility, and no radiation risk.

However, neither CT nor MRI is perfect. CT uses ionizing radiation that is potentially harmful to the patient if the radiation dose is not well controlled. Also, CT is not good at revealing subtle features of soft tissues. The recent development of photon-counting energy-discriminating CT promises to improve contrast resolution, but this spectral CT technology is still under development. As far as MRI is concerned, the major problems are also multiple. First, an MRI scanner is significantly more expensive than a CT scanner. Also, MRI suffers from poor spatial resolution when its scanning speed is not too slow. Moreover, there is substantial geometric distortion in MRI images, due to various magnetic interferences. Recently, it has been reported that some contrast agents used for MRI may be potentially harmful and have been stopped in Europe.

Our group proposed the conceptual design of a simultaneous CT-MRI scanner to integrate advantages of CT and MRI and correct their respective shortcomings for cardiac studies, cancer imaging, and other applications.[15] CT-MRI is the next step after PET-CT and PET-MRI and eventually would lead to a PET-CT-MRI scanner, which we call *Omnitomography* for super-radiomics.

To perform image reconstruction for our intended CT-MRI scanner, we proposed an algorithmic framework emphasizing structural coupling between CT and MRI images.[16] In the same spirit, there are algorithms available in the literature for such estimation, which are either atlas-based[17] or feature learning–based.[18] In our joint CT-MRI reconstruction process, we utilize mutual information between CT and MRI data, among other priors, to improve CT-MRI reconstruction quality. However, our proposed utilization of mutual information has been limited to the case of a single joint CT-MRI scan. Given the great successes of deep learning, we are motivated to develop a dual mechanism for estimation of CT images from MRI images and vice versa in the machine learning framework.

Now, we are interested in collecting big CT data and big MRI data, which may or may not be matched. One convenient way to collect CT and MRI images automatically is to search, download, and archive CT and MRI images from Internet. As an initial step of this effort, we need to be able to sort CT and MRI images. Here, we develop a preliminary second-order network for this purpose.

We downloaded 15 CT images and 15 MRI images from the Internet, extracted image patches of 28*28 pixels from these images, and then normalized them with the maximum pixel value in the corresponding image. For classification, we connected second-order neurons to form a second-order CNN, which consists of a convolutional layer, a pooling layer, and 2 fully connected layers, as shown in Figure 6. The features of the images were extracted with ten 5×5 filters with stride = 1 and padding = 0. The mean-pooling function was used in the pooling layer. We adopted the "linear" function in the output layer and the "tanh" function in the other layers. If the output is positive, then the input image is judged as CT. Otherwise, the input image is MRI. Then, with weights and biases randomly initialized using the "normrnd" function in MATLAB, the network was trained using our generalized BP algorithm. After 10 iterations, the CNN can classify CT and MRI images successfully.

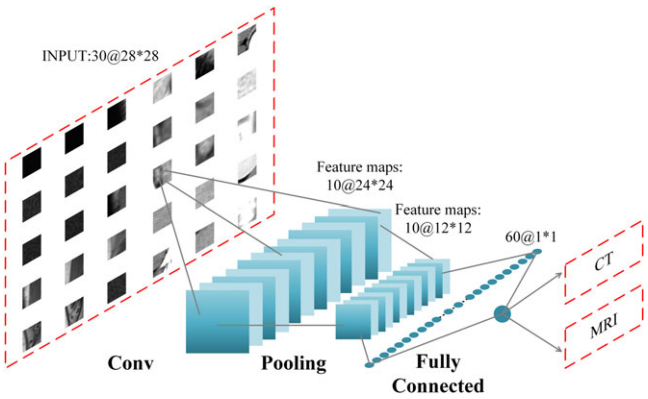

FIGURE 6 The structure of the second-order convolutional neural network trained to sort computed tomography (CT) and magnetic resonance imaging (MRI) images for the construction of a big dataset of CT and MRI images

Finally, we compared the performance of the second CNN with that of the first CNN in the same architectures. One has 10 initial feature maps and 60 subsequent feature maps (10-60 configuration), and the other has 5 initial feature maps and 70 subsequent feature maps (5-70 configuration). The number of iterations was set to 10. We trained the networks 20 times using randomly initialized weights and biases and then recorded the number of images correctly labeled. With the 10 to 60 configuration, the average success rate was 72.17% for the second-order CNN, which is higher than 60.00% for the first-order CNN. With the 5 to 70 configuration, the success rates were 76.50% and 63.83% for the second- and first-order networks, respectively.

4 | DISCUSSIONS AND CONCLUSION

In the literature, high-order neurons were considered in the early years of artificial intelligence.[19] What was called "a high-order neural network" is actually a computational unit without any hidden layers,[20] which is fundamentally different from what we have studied in this paper. Although those high-order neurons are more general than the first-order or linear neurons,[21,22] real applications of such neurons are generally restricted by the combinatorial explosion of the involved parameters. Thus, it is desirable to reduce the number of parameters needed to define a high-order neuron.[23-25] Compared with these prior arts, our proposed second neuron[8] is a unique design that extracts second-order features with a minimum number of parameters; ie, the number of parameters is only 3n where n is the number of inputs to the second-order neuron. Mathematically speaking, any neuron in the machine learning context is nothing but a composite function. The key is how to design such a composite function to balance the structural complexity, expressing power, and computational efficiency. Based on our pilot data, our second-order neuron with 3n parameters seems in a sweet spot. Moreover, we have proposed to construct deep networks with our second-order neurons for deep learning and developed a generalized BP algorithm for this purpose, which is the first step along this direction.

Our successes in the "Telling-Two-Spirals-Apart" and other tasks have demonstrated that our generalized BP algorithm works well for training the second-order multilayer forward network. In principle, the BP principle can be adapted for any second-order networks. Nevertheless, it is still desirable to refine the BP algorithm for better efficiency and reliability, using techniques such as *Quickprop*,[14] learning rate adaption,[26] and other methods in previous studies.[27-30]

Based on our experience so far with second-order networks, it seems that the second-order network could be considerably simpler in the network topology, despite that individual neurons are more complicated. Preliminary studies show that in certain applications, the overall "after-training" performance of the second-order network would be better than the first-order counterpart in terms of accuracy and speed for a given computational complexity; for example, in the case of separating concentric rings, the first-order network is inherently handicapped. Recurrent neural networks[27] consisting

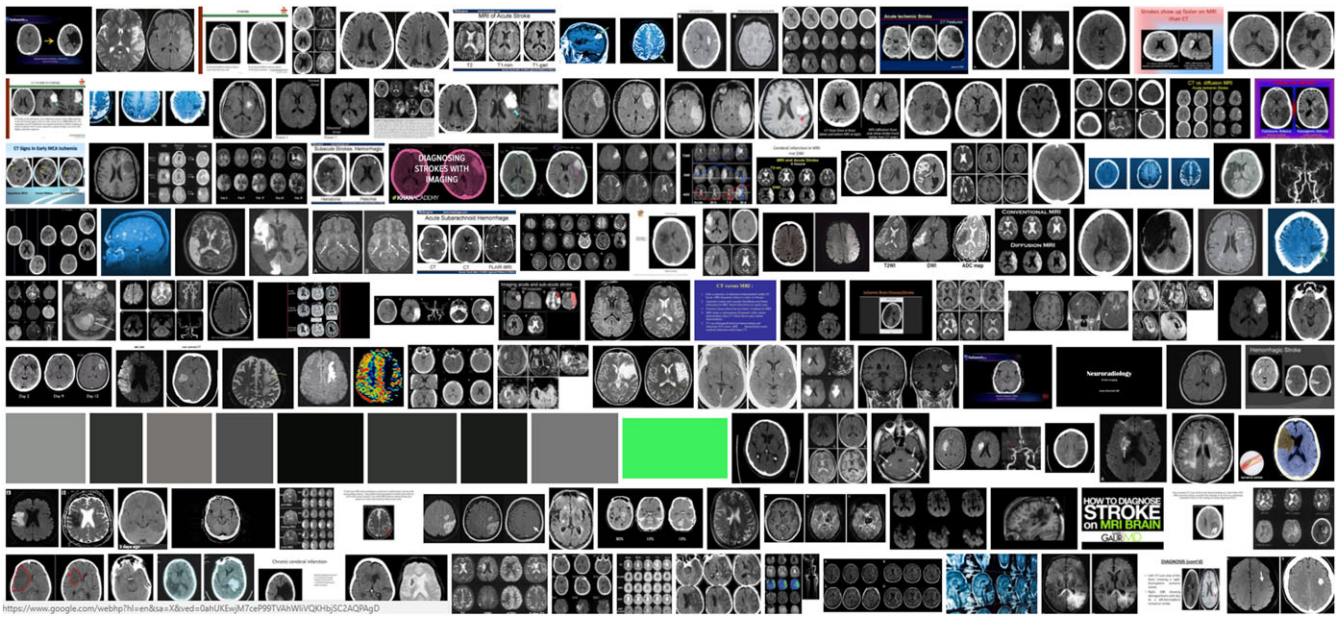

FIGURE 7 Exemplary CT/MRI images, which could be merged into a dedicated database for medical imaging research

of first neurons were demonstrated to perform well for approximation of nonlinear functions. Recurrent neural networks with second-order neurons should be more flexible.

As a biomedical imaging–related example, we have shown that a second-order network can sort CT and MRI images from the Internet. Next, we will enhance this network to not only differentiate between CT and MRI images but also reject those images that are of neither CT- nor MRI-type. This will lead to a big dataset consisting of numerous CT and MRI images from the Internet, as Figure 7 shown. Potentially, these images can be registered to some standard CT and MRI atlases such as the visible human CT and MRI volumes. Then this huge atlas should be able to help improve prediction from CT images to MRI images, and vice versa, and facilitate joint CT-MRI reconstruction as well.

In conclusion, the general BP algorithm has been formulated and tested for the second-order network in this pilot study. Numerical examples show the BP algorithm works well in some biomedical engineering relevant cases. Much more efforts are needed to refine the optimization method and design the second-order networks, and find killer applications in the real world.

ORCID

Fenglei Fan http://orcid.org/0000-0003-3691-5141
Ge Wang http://orcid.org/0000-0002-2656-7705

REFERENCES

1. Pereira S, Pinto A, Alves V, Silva CA. Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Trans Med Imaging*. 2016;35(5):1240-1251.
2. Shin HC, Roth HR, Gao M, et al. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Trans Med Imaging*. 2016;35(5):1285-1298.
3. Anthimopoulos M, Christodoulidis S, Ebner L, Christe A, Mougiakakou S. Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE Trans Med Imaging*. 2016;35(5):1207-1216.
4. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Netw*. 2015;61:85-117.
5. Goodfellow I, Bengio Y, Courville A. *Deep learning*. Cambridge: MIT Press; 2016.
6. Shen D, Wu G, Zhang D, Suzuki K, Wang F, Yan P. Machine learning in medical imaging. *Comput Med Imaging Graph*. 2015;41:1-2.
7. de Jesús Rubio J. USNFIS: uniform stable neuro fuzzy inference system. *Neurocomputing*. 2017;262:57-66.
8. Fan F, Cong W, Wang G. A new type of neurons for machine learning. *Int J Numer Meth Biomed Eng*. 2017; https://doi.org/10.1002/cnm.2956
9. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Netw*. 1989;2(5):359-366.
10. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Cognitive modeling, vol. 5, 1988.
11. Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks. *Chemom Intell Lab Syst*. 1997;39:43-62.
12. Lang KJ, Witbrock MJ. Learning to tell two spirals apart. In: *Proceedings of the 1988 connectionist models summer school*. San Mateo, CA: Morgan Kaufmann; 1989:52-59.
13. Zhou Z, Chen S, Chen Z. FANNC: a fast adaptive neural network classifier. *Knowl Inf Syst*. 2000;2(1):115-129.
14. Fahlman SE. An empirical study of learning speed in back-propagation networks, 1988.
15. Wang G, Kalra M, Murugan V, et al. Simultaneous CT-MRI—next chapter of multimodality imaging. *Med Phys*. 2015;42(10):5879-5889.
16. Xi Y, Zhao J, Bennett JR, et al. Simultaneous CT-MRI reconstruction for constrained imaging geometries using structural coupling and compressive sensing. *IEEE Trans Biomed Eng*. 2016;63(6):1301-1309.
17. Catana C, van der Kouwe A, Benner T, et al. Toward implementing an MRI-based PET attenuation-correction method for neurologic studies on the MR-PET brain prototype. *J Nucl Med*. 2010;51(9):1431-1438.
18. Huynh T, Gao Y, Kang J, et al. Estimating CT image from MRI data using structured random forest and auto-context model. *IEEE Trans Med Imaging*. 2016;35:174-183.
19. Minsky ML, Papert S. *Perceptrons*. Cambridge: MIT Press; 1969.
20. Giles CL, Maxwell T. Learning, invariance, and generalization in high-order neural networks. *Appl Optics*. 1987;26(23):4972-4978.
21. Wang Z, Fang JA, Liu X. "Global stability of stochastic high-order neural networks with discrete and distributed delays," chaos. *Solitons Fractals*. 2008;36(2):388-396.

22. Lu Z, Shieh LS, Chen G, Coleman NP. Adaptive feedback linearization control of chaotic systems via recurrent high-order neural networks. *Inform Sci*. 2006;176(16):2337-2354.

23. Kowalczyk A, Ferra HL. Developing higher-order networks with empirically selected units. *IEEE Trans Neural Netw*. 1994;5(5):698-711.

24. Kohonen T. *Self-organisation and Associative Memory*. Berlin: Springer-Verlag; 1989.

25. Specht TDF. Probabilistic neural networks and polynomial adaline as complementary techniques for classification. *IEEE Trans Neural Netw*. 1990;1:111-121.

26. Li Y, Fu Y, Li H, et al. The improved training algorithm of back propagation neural network with self-adaptive learning rate. 2009 International Conference on Computational Intelligence and Natural Computing, 2009.

27. Lipton ZC, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019, 2015.

28. Spratling MW, Hayes G. Learning synaptic clusters for nonlinear dendritic processing. *Neural Process Lett*. 2000;11(1):17-27.

29. Huk M. Backpropagation generalized delta rule for the selective attention sigma-if artificial neural network. *Int J Appl Math Comput Sci*. 2012;22(2):449-459.

30. Huk M. Notes on the generalized backpropagation algorithm for contextual neural networks with conditional aggregation functions. *J Intell Fuzzy Syst*. 2017;32(2):1365-1376.

How to cite this article: Fan F, Cong W, Wang G. Generalized backpropagation algorithm for training second-order neural networks. *Int J Numer Meth Biomed Engng*. 2018;34:e2956. https://doi.org/10.1002/cnm.2956