Neural Networks 124 (2020) 383–392

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2020 Special Issue

Universal approximation with quadratic deep networks

Fenglei Fan [a], Jinjun Xiong [b], Ge Wang [a,*]

[a] *Department of Biomedical Engineering, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA*
[b] *IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 10598, USA*

ARTICLE INFO

Article history:
Available online 18 January 2020

Keywords:
Deep learning
Quadratic networks
Approximation theory

ABSTRACT

Recently, deep learning has achieved huge successes in many important applications. In our previous studies, we proposed quadratic/second-order neurons and deep quadratic neural networks. In a quadratic neuron, the inner product of a vector of data and the corresponding weights in a conventional neuron is replaced with a quadratic function. The resultant quadratic neuron enjoys an enhanced expressive capability over the conventional neuron. However, how quadratic neurons improve the expressing capability of a deep quadratic network has not been studied up to now, preferably in relation to that of a conventional neural network. Specifically, we ask four basic questions in this paper: (1) for the one-hidden-layer network structure, is there any function that a quadratic network can approximate much more efficiently than a conventional network? (2) for the same multi-layer network structure, is there any function that can be expressed by a quadratic network but cannot be expressed with conventional neurons in the same structure? (3) Does a quadratic network give a new insight into universal approximation? (4) To approximate the same class of functions with the same error bound, could a quantized quadratic network have a lower number of weights than a quantized conventional network? Our main contributions are the four interconnected theorems shedding light upon these four questions and demonstrating the merits of a quadratic network in terms of expressive efficiency, unique capability, compact architecture and computational capacity respectively.

1. Introduction

Over recent years, deep learning (Goodfellow, Bengio, & Courville, 2016) has become the mainstream approach for machine learning. Since AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), increasingly more advanced neural networks (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, & Bengio, 2014; He, Zhang, Ren, & Sun, 2016; Huang, Liu, Van Der Maaten, & Weinberger, 2017; Simonyan & Zisserman, 2014; Szegedy et al., 2015) are being proposed, such as GoogleNet, ResNet, DenseNet, GAN and their variants, to enable practical performance comparable to or beyond what human delivers in computer vision (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016), speech recognition (Dahl, Yu, Deng, & Acero, 2012), language processing (Kumar et al., 2016) game playing (Silver et al., 2017), medical imaging (Hong et al., 2019; Wang, 2016; Zhang & Yu, 2018), and so on. A heuristic understanding of why these deep learning models are so successful is that these models represent knowledge in hierarchy and facilitate high-dimensional non-linear functional fitting. It seems that deeper structures are correlated with greater capacities to approximate more complicated functions.

The representation power of neural networks has been rigorously studied since the eighties. The first result is that a network with a single hidden layer can approximate a continuous function at any accuracy given an infinite number of neurons (Hornik, Stinchcombe, & White, 1989). This means that the network can be extremely wide. Then, a number of papers were published investigating the complexity of approximation. For example, Barron (1993) studied one-hidden-layer feedforward networks using sigmoidal activations, which can approximate a function family, wherein the first moment of the magnitude distribution of the Fourier transform of the function is bounded by a constant, with an accuracy in the order of $\frac{1}{n}$, where n is the number of nodes and Kainen, Kurkova, and Sanguineti (2012) demonstrated that for worst-case error, a generic function with dimensionality d in the Hilbert space can be approximated by a one-hidden-layer neural network with an error of $\epsilon(d)\kappa(n)$, where $\epsilon(d)$ is bounded above by a polynomial in d and $\kappa(n)$ is $n^{-1/m}, m > 0$. Kon et al. show that given that the number of nodes n is no smaller than the number of examples k, the optimal network to approximate a function in reproducing kernel Hilbert space is given by $\sum_i^k c_i G(t_i, x)$, where t_i is the sample input and G is the RBF neuron (Kon & Plaskota, 2000). Please note that the optimality means the minimal error achieved in the worst case over all the possible f. Schmitt borrowed the results based on

* Corresponding author.
E-mail address: wangg6@rpi.edu (G. Wang).

https://doi.org/10.1016/j.neunet.2020.01.007
0893-6080/© 2020 Elsevier Ltd. All rights reserved.

Vapnik–Chervonenkis dimension to prove that to approximate any polynomials, the network size needs to grow $\mathcal{O}((logk)^{1/4})$, where k is the degree of the polynomial (Schmitt, 2000). With the emergence of deep neural networks, studies have been performed on theoretical benefits of deep models over shallow ones (Bianchini & Scarselli, 2014; Cohen, Sharir, & Shashua, 2016; Eldan & Shamir, 2016; Kurkova & Sanguineti, 2017; Liang & Srikant, 2017; Lin & Jegelka, 2018; Lin, Tegmark, & Rolnick, 2017; Lu, Pu, Wang, Hu, & Wang, 2017; Mhaskar & Poggio, 2016; Szymanski & McCane, 2014; Telgarsky, 2016). One way (Eldan & Shamir, 2016; Telgarsky, 2016) is to construct a special kind of functions that are easy to be approximated by deep networks but hard by shallow networks. It has been reported in Lu et al. (2017) that a fully-connected network with ReLU activation can approximate any Lebesgue integrable function in the L_1-norm sense, provided a sufficient depth and at most $d+4$ neurons in each layer, where d is the number of inputs. Through a similar analysis, it was reported in Lin and Jegelka (2018) that ResNet with one single neuron per layer is a universal approximator. Moreover, it was demonstrated in Eldan and Shamir (2016) that a special class of functions is hard to be approximated by a conventional network with a single hidden layer unless an exponential number of neurons are used. Bianchini and Scarselli (2014) utilized a topological measure to characterize the complexity of functions that are realized by neural networks, and proved that deep networks can represent functions of much higher complexity than the shallow counterparts. Kurkova and Sanguineti (2017) showed that unless with sufficiently many network units (more than any polynomial of the logarithm of the size of the domain), a good approximation cannot be achieved with a shallow perceptron network for almost any uniformly chosen function on a specially constructed domain.

In our previous studies (Fan, Cong, & Wang, 2017, 2018; Fan, Shan, & Wang, 2019; Fan & Wang, 2018), we proposed quadratic neurons and deep quadratic networks. In a quadratic neuron, the inner product of an input vector and the corresponding weights in a conventional neuron is replaced with a quadratic function. The resultant quadratic neuron enjoys an enhanced expressive capability over the conventional neuron (Here conventional neurons refer to neurons that perform activation of an inner-product, and conventional networks refer to networks completely comprising conventional neurons). Actually, a quadratic neuron can be viewed as a fuzzy logic gate, and a deep quadratic network is nothing but a deep fuzzy logic system (Fan & Wang, 2018). Furthermore, we successfully designed a quadratic autoencoder for a real-world low-dose CT problem (Fan et al., 2019). Note that high-order neurons (Giles & Maxwell, 1987; Minsky & Papert, 1969) were taken into account in the early stage of artificial intelligence, but they are not connected to deep networks and suffer from a combinatorial explosion with the number of parameters due to high order terms. In contrast, our quadratic neuron uses a limited number of parameters (tripled that of a conventional neuron) and performs a cost-effective high-order operation in the context of deep learning. For quadratic deep networks, we already developed a general backpropagation algorithm (Fan et al., 2017), enabling the network training process.

However, how quadratic neurons improve the expressing capability of a deep quadratic network has not been theoretically studied up to now, preferably in relation to that of a conventional neural network. In this paper, we ask four basic questions regarding the expressive capability of a quadratic network: (1) for the one-hidden-layer network structure, is there any function that a quadratic network can approximate much more efficiently than a conventional network? Here, the efficiency means less neurons. (2) for the same multi-layer network structure, is there any function that can be expressed by a quadratic network but cannot be expressed with conventional neurons in the same structure? (3) Does a quadratic network give a new insight and a new method for universal approximation? (4) To approximate the same class of functions with the same error bound, is a quadratic network able to enjoy a lower number of weights than a conventional network? If the answers to these questions are favorable, quadratic networks should be significantly more powerful in many machine learning tasks.

In this paper, our contributions are to present four theorems addressing the above questions respectively and positively, thereby establishing the intrinsic advantages of quadratic networks over conventional networks. More specifically, these theorems characterize the merits of a quadratic network in terms of expressive efficiency, unique representation, compact architecture and computational capacity. We answer the first question with the first theorem, given the network with only one hidden layer and admissible activation function, there exists a function that a quadratic network can approximate it with a polynomial number of neurons but a conventional network can only do the same level approximation with an exponential number of neurons. Regarding the second question and the second theorem below, with the ReLU activation function, any continuous radial function can be approximated by a quadratic network in a structure of no more than four neurons in each layer but the function cannot be approximated by a conventional network of the same structure (Lu et al., 2017), while with the third theorem below we provide a new insight and a new method for a universal presentation from the perspective of the Algebraic Fundamental Theorem. Without introducing complex numbers, a univariate polynomial of degree n can be uniquely factorized into a product of quadratic terms instead of first-order terms. Since a ReLU quadratic network can represent any univariate polynomial in a *unique and global* manner, and by the Weierstrass theorem and the Kolmogorov theorem that multivariate functions can be represented through summation and composition of univariate functions, we can approximate any multivariate function with a well-structured ReLU quadratic neural network, justifying the universal approximation power of the quadratic network. To our best knowledge, our quadratic network is the first-of-the-kind global universal approximator. Finally, with the fourth theorem below, we attempt to address the theoretical foundation for quantization of a neural network. To approximate the same class of functions with the same error bound, a quantized quadratic network demands a much lower number of weights than a quantized conventional network. The saving is almost in the order of $\mathcal{O}\left(\lambda\left(log^{\frac{1}{\lambda-1}+1}(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$, $\lambda \geq 2$, $d, n \geq 1$. Compared to the previous studies that theoretically explored the properties of quantized networks, our result adds unique insights into this aspect.

There are prior papers related to but different from our contributions (Andoni, Panigrahy, Valiant, & Zhang, 2014; Blondel et al., 2016; Du & Lee, 2018; Krotov & Hopfield, 2018; Liang & Srikant, 2017; Livni et al., 2014). Motivated by a need for more powerful activation, Livni et al. (2014) and Krotov and Hopfield (2018) proposed to use the quadratic activation: $\sigma(z) = z^2$ and rectified polynomials in the neuron respectively. Despite somewhat misleading in their name, networks with quadratic activation or rectified polynomials and our proposed networks that consist of quadratic neurons have fundamental differences. At the cellular level, a neuron with quadratic activation is still characterized with a linear decision boundary, while our quadratic neuron allows a quadratic decision boundary. In Livni et al. (2014), the authors demonstrated that networks with quadratic activation are as expressive as networks with threshold activation, and constant-depth networks with quadratic activation can learn in polynomial time. In contrast, our work goes further showing that the expressibility of the quadratic network is superior to that of the conventional networks; for example, a single quadratic neuron can implement the XOR gate, and a quadratic network of finite width and depth can represent a finite-order polynomial up to any desirable accuracy. In Du and Lee (2018), Du et al. showed that over-parameterization and weight decay are instrumental to the optimization aided by quadratic activation. Andoni et al. (2014) reported how a neural network can provably learn a low-degree polynomial with gradient descent search from scratch, with an emphasis on the effectiveness of the gradient descent method. Liang and Srikant (2017) present that $O(log\frac{p}{\epsilon})$ layers of binary units and $O(plog\frac{p}{\epsilon})$ ReLU units can approximate $f(x) = \sum_{i=0}^{p} a_i x^i$ with closeness of ϵ. In contrast, our Theorem 3 is based on the Algebraic Fundamental Theorem to provide an exact representation of any finite-order polynomial. Blondel et al. (2016) are on factorization machine (FM) dedicated to combine high order features, clearly different from the polynomial factorization we propose to perform using a quadratic network.

2. Preliminaries

Quadratic/second-order neuron: The n-input function of a quadratic/second-order neuron before being nonlinearly processed is expressed as:

$$\begin{aligned} h(\mathbf{x}) &= (\sum_{i=1}^{n} w_{ir}x_i + b_r)(\sum_{i=1}^{n} w_{ig}x_i + b_g) + \sum_{i=1}^{n} w_{ib}x_i^2 + c \\ &= (\mathbf{w}_r\mathbf{x}^T + b_r)(\mathbf{w}_g\mathbf{x}^T + b_g) + \mathbf{w}_b(\mathbf{x}^2)^T + c, \end{aligned} \tag{1}$$

where $\mathbf{x}$ denotes the input vector, $\mathbf{w_r}, \mathbf{w_g}, \mathbf{w_b}$ are vectors of the same dimension with $\mathbf{x}$ and b_r, b_g, c are adjustable biases. Our quadratic function definition only utilizes $3n$ parameters, which is more compact than the general second-order representation requiring $\frac{n(n+1)}{2}$ parameters. While our quadratic neuron design is unique, other papers on quadratic neurons are also in the later literature; for example, Tsapanos, Tefas, Nikolaidis, and Pitas (0000) proposed a type of neurons with paraboloid decision boundaries. It is underlined that the emphasis of our work is not only on quadratic neurons individually but also deep quadratic networks in general.

One-hidden-layer networks: The generic function by a one-hidden-layer conventional network is as follows:

$$f_1(\mathbf{x}) = \sum_{l=1}^{k} t^l \sigma_l(\mathbf{w}^l\mathbf{x} + b^l), \tag{2}$$

where l refers to the lth neuron, and σ_l is the activation function. In contrast, the generic function represented by a one-hidden-layer quadratic network is:

$$f_2(\mathbf{x}) = \sum_{l=1}^{k} t^l \sigma_l[(\mathbf{w}_r^l\mathbf{x}^T + b_r^l)(\mathbf{w}_g^l\mathbf{x}^T + b_g^l) + \mathbf{w}_b^l(\mathbf{x}^2)^T + c^l]. \tag{3}$$

In our Theorem 1, we will compare the representation capability of a quadratic network and that of a conventional network assuming that both networks are of one hidden layer.

L-Lipschitz function: A L-Lipschitz function f from $\mathbb{R}^n$ to $\mathbb{R}$ is defined by the following property:

$$|f(\mathbf{x}) - f(\mathbf{y})| \le L\|\mathbf{x} - \mathbf{y}\|$$

Radial Function: A radial function only depends on the norm of its input vector, generically denoted as $f(\|\mathbf{x}\|)$. The functions mentioned in Theorems 1 and 2 are all radial functions.

Euclidean unit-volume ball: In a d-dimensional space, let R_d be the radius of a Euclidean ball such that the ball has the unit volume. Euclidean ball is used to define density function μ later. Here, the main reason why the μ is defined on the Euclidean ball is because the function f being constructed is a radial function, and the Fourier transform of f is also radial. Thus, it is appropriate to investigate f in the Euclidean ball, which simplifies the analysis and helps focusing on the essence of the problem. The extension of considering other domains (such as cube) will be considered for future work.

Bernstein polynomial: $l_{n,m} = C_n^m x^m(1-x)^{n-m}, 0 \le m \le n$. The nth Bernstein polynomial of the arbitrary continuous function $f(x)$ in (0, 1) is defined as

$$B_n(x) = \sum_{m=0}^{n} f(\frac{m}{n}) l_{n,m}(x)$$

Previously, the Bernstein Polynomials are used to prove the Weierstrass theorem.

Admissible function: An activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is called admissible if:

(1) σ is measurable

(2) there are $K, \kappa > 0$, $|\sigma(x)| \le K(1 + |x|^\kappa)$ for all $x \in \mathbb{R}$

(3) given any L−Lipschitz function $f : \mathbb{R} \to \mathbb{R}$ which is constant outside a bounded interval $[-R, R]$ and any $\delta > 0$, there are scalars $a \in \mathbb{R}$ and $(\alpha_i, \beta_i, \gamma_i)_{i=1,2,\dots,w}$ such that

$$|f(x) - [a + \sum_{i=1}^{w} \alpha_i \sigma(\beta_i x - \gamma_i)]| \le \delta,$$

where $w \le c_\sigma \frac{RL}{\delta}$ and c_σ is a constant.

The standard activation functions such as ReLU and sigmoid satisfy the above three properties. Detailed proofs are out of the scope of this paper, please refer to Blondel et al. (2016) and Eldan and Shamir (2016).

Function space $\mathcal{F}_n^d$: The Sobolev space $\mathcal{W}^{n,\infty}([0, 1]^d)$ with $n = 1, 2, \dots$ is defined on $[0, 1]^d$, lying in L_∞ space together with their weak derivatives of up to order n. The function space $\mathcal{F}_n^d$ is made of any function $f \in \mathcal{W}^{n,\infty}([0, 1]^d)$ and

$$\max_{\mathbf{n}:|\mathbf{n}| \le n} ess \sup_{\mathbf{x} \in [0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})| \le 1,$$

where $|\mathbf{n}| = \sum_{k=1}^{d} n_k$.

3. Four theorems

First, we present four theorems, and then give their proofs respectively.

Theorem 1. *For an admissible activation function $\sigma(\cdot)$ and for some universal constants $c > 0, C > 0, C' > 0, c_\sigma > 0$, there exist a probability measure μ and a radial function $\tilde{g}: \mathbb{R}^d \to \mathbb{R}$, where $d > C$, that is bounded on [-2,2] and supported on $\|\mathbf{x}\| \le C'\sqrt{d}$ satisfying:*

1. $\tilde{g}$ can be approximated by a single-hidden-layer quadratic network with $C'c_\sigma d^{3.75}$ neurons, which is denoted as f_2.

2. For every function f_1 expressed by a single-hidden-layer conventional network with at most $c \times e^{c \times d}$ neurons, we have:

$$E_{\mathbf{x}\sim\mu}(f_1(\mathbf{x}) - f_2(\mathbf{x}))^2 \ge \delta$$

for some positive constant δ, where $E_{\mathbf{x}\sim\mu}(f_1(\mathbf{x}) - f_2(\mathbf{x}))^2$ is the distance between $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ regarding the measure μ.

Theorem 2. *For any compactly supported, radial, continuous function $f: \mathbb{R}^d \to \mathbb{R}$ and any $\epsilon > 0$, there exists a function g that can be implemented by a ReLU-activated quadratic network with at most four neurons in each layer, such that:*

$$\sup_{\mathbf{x} \in \mathbb{R}^d} |f(\mathbf{x}) - g(\mathbf{x})| \le \epsilon.$$

Theorem 3. *With ReLU as activation function, the quadratic network is a global universal approximator.*

Theorem 4. *For any $f \in \mathcal{F}_d^n$ and any $\epsilon \in (0, 1)$, there is a ReLU quadratic network with λ distinct weights that can approximate f with ϵ, satisfying:*
(i) the depth is $\mathcal{O}\left(log(log(\frac{1}{\epsilon})) + log(\frac{1}{\epsilon})\right)$
(ii) the number of weights is $\mathcal{O}\left(log(log(\frac{1}{\epsilon}))(\frac{1}{\epsilon})^{\frac{d}{n}} + log(\frac{1}{\epsilon})(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$.

3.1. Theorem 1

Key idea for proving theorem 1: The form of functions represented by a single-hidden-layer conventional network is $f_1(\mathbf{x}) = \sum_{l=1}^{k} t^l\sigma_l(\mathbf{w}^l\mathbf{x} + b^l)$. It is observed that the distribution of the Fourier transform of $f_1(\mathbf{x})$ is supported on a finite collections of lines. The support covered by the finite lines is sparse in the Fourier space, especially for a high dimension and high frequency regions, unless an exponential number of lines are involved. Thus, a possible target function to be constructed should have major components at high-frequencies. A suitable candidate has been constructed in Eldan and Shamir (2016):

$$\tilde{g}(\mathbf{x}) = \sum_{i=1}^{N} \epsilon_i g_i(\|\mathbf{x}\|),$$

where $\epsilon_i \in \{-1, 1\}$, N is a polynomial function of d, $g_i(\|\mathbf{x}\|) = \mathbf{1}\{\|\mathbf{x}\| \in \Delta_i\}$ are radial indicator functions over disconnected intervals. According to the definition, $\tilde{g}$ is well bounded. Although the constructed $\tilde{g}$ is hard to approximate by a conventional network, it is easy to approximate by a quadratic network, because $\tilde{g}$ is a radial function. Consequently, it is feasible for a single-hidden-layer quadratic network to approximate the radial function with a polynomial number of neurons. Note that $\tilde{g}(\mathbf{x})$ is discontinuous, hence it cannot be perfectly expressed by a neural network with continuous activation functions. Therefore, let us use a probability measure μ for network quality assessment. With μ, the distance between $\tilde{g}$ and f represented by a network is characterized as $E_{\mathbf{x}\sim\mu}(f(\mathbf{x})-\tilde{g}(\mathbf{x}))^2$. In particular, $\mu = \phi^2$, where ϕ is the inverse Fourier transform of the indicator function defined on the Euclidean ball: $\mathbf{1}\{x \in B\}$. Therefore, $E_{\mathbf{x}\sim\mu}(f(\mathbf{x}) - \tilde{g}(\mathbf{x}))^2 = \|\widehat{f\phi} - \widehat{\tilde{g}\phi}\|^2_{L_2}$, where $\widehat{f\phi}$, $\widehat{\tilde{g}\phi}$ are the frequency forms of $f\phi$ and $\tilde{g}\phi$. The physical meaning of $E_{\mathbf{x}\sim\mu}(f(\mathbf{x}) - \tilde{g}(\mathbf{x}))^2$ is to measure the closeness of $\tilde{g}$ and f in frequency domain within Euclidean ball B.

Please note that c and c_σ are different constants. In Lemma 1, as long as the number of neurons is no more than ce^{cd}, the distance between f_1 and $\tilde{g}$ in the sense of $L_2(\mu)$ will be greater than a constant. In Lemma 2, c_σ is a fixed constant after the activation function σ is pre-specified.

The contribution of our work: In our Theorem 1, we demonstrate the utility of a quadratic network in approximating the constructed radial function $\tilde{g}$, which is straightforward but interesting and non-trivial. Furthermore, our result holds for all the admissible activation functions, while the proof in Eldan and Shamir (2016) is only for the ReLU activation function. Proposition 1 in Eldan and Shamir (2016) demonstrates that $\tilde{g}$ cannot be well approximated by a single-hidden-layer conventional network with a polynomial number of neurons. We put this proposition here as Lemma 1 for completeness. We would like to emphasize that although we put Lemma 1 and other Lemmas on an equal footing, Lemma 1 is more important than other lemmas. Actually, our Theorem 1 is merely a corollary of Proposition 1 in Eldan and Shamir (2016) when ReLU activation is used. Again, our contribution is the proof that the one-hidden-layer quadratic network using an admissible activation function can approximate $\tilde{g}$.

Lemma 1. *There are universal constants $c, C > 0$ such that for every $d > C$, there is a probability measure μ on $\mathbb{R}^d$ such that for any $\alpha > C$ and $N \geq C\alpha^{1.5}d^2$, there exists a function $\tilde{g}(\mathbf{x}) = \sum_{i=1}^{N} \epsilon_i g_i(\mathbf{x})$, where $\epsilon_i \in \{-1, 1\}, i = 1, 2, \ldots, N$, such that for any function of the form: $f_1(\mathbf{x}) = \sum_{l=1}^{k} t^l\sigma_l(\mathbf{w}^l\mathbf{x} + b^l)$ with σ_l as admissible function and $k \leq ce^{cd}$, we have:*

$$\|f_1 - \tilde{g}\|_{L_2(\mu)} \geq \delta/\alpha,$$

where $\delta > 0$ is a universal constant.

Universal constants means they are independent from the dimension d. To illustrate $\tilde{g}$ is expressible with a quadratic network, we know from Lemma 12 of Eldan and Shamir (2016) that a continuous Lipschitz function g can approximate $\tilde{g}$, what is remained for us to do is to use a quadratic network with a polynomial number of neurons to approximate g.

Lemma 2. *Given an admissible activation function σ, there is a constant $c_\sigma \geq 1$ (depending on σ and other parameters) such that for any L-Lipschitz function $g : \mathbb{R} \to \mathbb{R}$, which is constant outside a bounded interval $[r, R]$ ($r \geq 1$) and any δ, there exist scalars a, $\{\alpha_i, \beta_i, \gamma_i\}$, $i = 1, \ldots, w$, $w \leq c_\sigma \frac{(R^2-r^2)L}{4\sqrt{r}\delta}$ with which we have:*

$$h(\|\mathbf{x}\|^2) = a + \sum_{i=1}^{w} \alpha_i\sigma(\beta_i\|\mathbf{x}\|^2 - \gamma_i)$$

satisfies:

$$\sup_{\|\mathbf{x}\|\in[r,R]} |g(\|\mathbf{x}\|) - h(\|\mathbf{x}\|^2)| < \delta.$$

Proof. $g : \mathbb{R} \to \mathbb{R}$ is an L-Lipschitz function and supported on $[r, R]$, $r > 0$. Let us define $s(x) = g(\sqrt{x})$, which is of $\frac{L}{2\sqrt{r}}$-Lipschitz and supported on $[r^2, R^2]$. By the property of an admissible function, we construct a function $h(t) = a + \sum_{i=1}^{w} \alpha_i\sigma(\beta_i t - \gamma_i)$, which satisfies the following condition:

$$\sup_{l\in[r^2,R^2]} |s(l) - h(l)| < \delta,$$

where $w = c_\sigma \frac{(R^2-r^2)L}{4\sqrt{r}\delta}$. Next, we have:

$$\begin{aligned} &\sup_{\|\mathbf{x}\|\in[r,R]} |g(\|\mathbf{x}\|) - h(\|\mathbf{x}\|^2)| \\ = &\sup_{\|\mathbf{x}\|\in[r,R]} |s(\|\mathbf{x}\|^2) - h(\|\mathbf{x}\|^2)| \\ = &\sup_{l\in[r^2,R^2]} |s(l) - h(l)| < \delta \end{aligned}$$

Lemma 3. *There are a universal constant $C > 0$ and $\delta \in (0, 1)$, for $d \geq C$ and any choice of $\epsilon_i \in \{-1, 1\}, i = 1, 2, \ldots, N$, there exists a function f_2 expressed by a single-hidden-layer quadratic network of a width of at most $c_\sigma \frac{3N\alpha^{1.5}d^{1.75}}{4\delta}$ and with the range $[-2,+2]$ such that*

$$\|f_2 - \tilde{g}\|_{L_2(\mu)} \leq \frac{\sqrt{3}}{\alpha d^{\frac{1}{4}}} + \delta.$$

Proof. In Lemma 2, we make the following substitutions: $R = 2\alpha\sqrt{d}$, $r = \alpha\sqrt{d}$, $L = N$, $h = g$. Notably, α can be greater than 1, and $r = \alpha\sqrt{d} > 1$ satisfying the condition of Lemma 2. Thus, $g(\|\mathbf{x}\|)$ is expressible by a single-hidden-layer quadratic network with at most $c_\sigma \frac{3N\alpha^{1.5}d^{1.75}}{4\delta}$ neurons. Coupled with Lemma 12 of Eldan and Shamir (2016), Lemma 3 is immediately obtained.

Proof of Theorem 1. By the combination of Lemmas 1 and 3, the proof for Theorem 1 is straightforward. In Lemma 1, by choosing

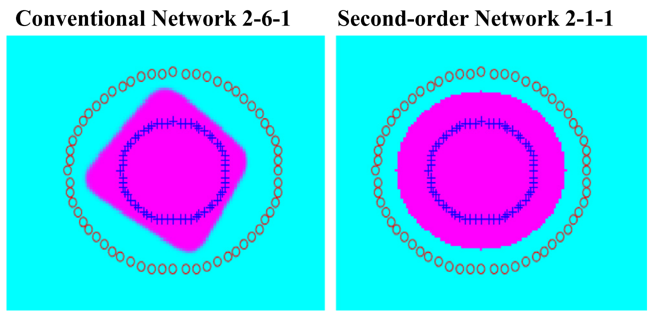


Fig. 1. Classification of two concentric rings with conventional and quadratic networks ($d = 2$). To succeed in the classification, a conventional network requires at least 6 neurons (Left), and a quadratic network only takes one neuron.

$\alpha = C$, $N = C\alpha^{1.5}d^2$, we have

$\|f_1 - \tilde{g}\|_{L_2(\mu)} \geq \delta_1$

Let $\delta \leq \frac{\delta_1}{2} - \frac{\sqrt{3}}{Cd^{\frac{1}{4}}}$, which demands d is very large so that $\delta > 0$, to approximate $\tilde{g}$ we need the number of quadratic neurons being at most

$$c_\sigma \frac{3N\alpha^{1.5}d^{1.75}}{4\delta} = c_\sigma \frac{3C^{2.5}d^{3.75}}{4\delta} \leq C'c_\sigma d^{3.75}$$

(where C' is a universal constant depending on the universal constants C, δ)

such that

$$\|f_2 - \tilde{g}\|_{L_2(\mu)} \leq \frac{\sqrt{3}}{Cd^{\frac{1}{4}}} + \delta \leq \frac{\delta_1}{2}.$$

Therefore, we have $\|f_1 - f_2\|_{L_2(\mu)} \geq \frac{\delta_1}{2}$. The proof is completed.

Classification example: The exponential difference between the conventional and quadratic networks in our Theorem 1 is due to the dimension d. To demonstrate this, we constructed an example for separation of two concentric rings. In this example, there are 60 instances in each of the two rings representing two classes. With only one quadratic neuron in a single hidden layer, the rings were totally separated, while at least six conventional neurons are required to complete the same task, as shown in Fig. 1.

3.2. *Theorem 2*

Key idea for proving theorem 2: It was proved in Lu et al. (2017) that an n-input function that is not constant along any direction cannot be well approximated by a conventional network with no more than $n - 1$ neurons in each layer. However, for such a radially defined function, it is feasible for a quadratic network to approximate the function with width $= 4$, which breaks the lower width bound $n + 4$ given in Lu et al. (2017). The trick for the approximation by a deep conventional network is adapted for analysis of a deep quadratic network so that the function can be approximated in a "quadratic" way via composition layer by layer. To compute a radial function, we need to find the norm and then evaluate the function approximation by the norm. With a quadratic neuron, the norm can be easily found. Heuristically speaking, a quadratic network with no more than $n - 1$ neurons in each layer could approximate a radial function very well, even if the function is not constant along any direction. Therefore our theorem positively shows that it is more natural and more effective to represent functional non-linearity with a quadratic network. Finally, since a quadratic network with width

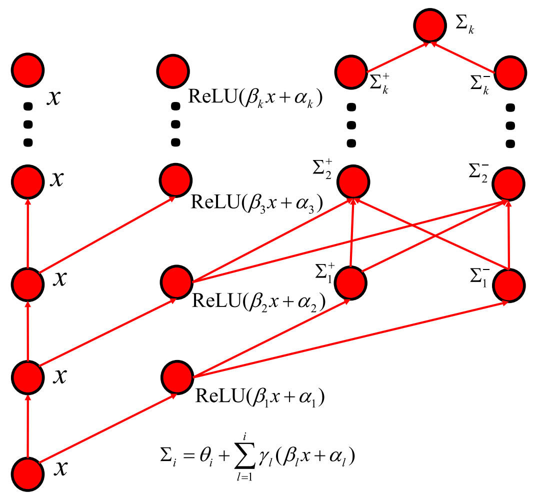

Σ_k
x
$\text{ReLU}(\beta_k x + \alpha_k)$
Σ_k^+
Σ_k^-
x
Σ_2^+
Σ_2^-
$\text{ReLU}(\beta_3 x + \alpha_3)$
x
Σ_1^+
Σ_1^-
$\text{ReLU}(\beta_2 x + \alpha_2)$
x
$\text{ReLU}(\beta_1 x + \alpha_1)$
x
$\Sigma_i = \theta_i + \sum_{l=1}^{i} \gamma_l(\beta_l x + \alpha_l)$

Fig. 2. Illustration of a conventional network structure that allows to implement any continuous univariate piecewise linear function.

at most 4 can approximate any radial function and radial function family can approximate any continuous function (Park & Sandberg, 1991), a quadratic network can serve as a width-bounded universal approximator.

Proof of Theorem 2. As per structure outlined in Fig. 2, a conventional ReLU network with four neurons in each layer can implement any function of the form:

$$[0, +\infty] \to \mathbb{R} : x \to \theta + \sum_{l=1}^{k} \gamma_l ReLU(\alpha_l x + \beta_l),$$

which is actually any continuous piecewise linear function defined on non-negative real number domain. Because function $f : \mathbb{R}^d \to \mathbb{R}$ is radial and continuous, without loss of generality, let $h(\|\mathbf{x}\|^2) = f(\mathbf{x})$, while h is function: $[0, +\infty] \to \mathbb{R}$. It implies that for any $\epsilon > 0$, there will be a continuous piecewise linear function $g : [0, +\infty] \to \mathbb{R}$, and such that

$$\sup_{[0,+\infty]} |h(s) - g(s)| < \epsilon.$$

g can be implemented by a conventional ReLU network as shown in Fig. 2. Therefore, the function $\mathbf{x} \to g(\|\mathbf{x}\|_2^2) : \mathbb{R}^d \to \mathbb{R}$ can be implemented with a quadratic ReLU network of the same structure by calculating norm of input. Then we will obtain:

$$\begin{aligned}
&\sup_{\mathbf{x}\to\mathbb{R}^d} |f(\mathbf{x}) - g(\|\mathbf{x}\|_2^2)| \\
&= \sup_{x\to\mathbb{R}^d} |h(\|\mathbf{x}\|_2^2) - g(\|\mathbf{x}\|_2^2)| \\
&= \sup_{s\in[0,+\infty]} |h(s) - g(s)| \\
&< \epsilon
\end{aligned}$$

Lemma 4 (*Park & Sandberg, 1991*). *Let $K : \mathbb{R}^r \to \mathbb{R}$ be an integrable bounded function such that K is continuous almost everywhere and $\int_{\mathbb{R}^r} K(x)\,dx \neq 0$. Then the functional family $S_K : \sum_{i=1}^{M} w_i K(\frac{\|x - z_i\|}{\sigma})$, ($M$ is a positive integer, $\sigma > 0$ and $w_i \in \mathbb{R}$) is dense in $L^p(\mathbb{R}^r)$.*

Proof. Please see Park and Sandberg (Theorem 1 1991).

Corollary 1. *With ReLU activation function, residual quadratic networks with at most four neurons in one layer is universal approximator.*

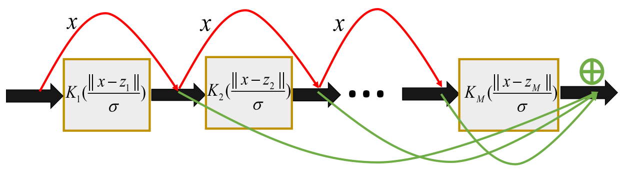

x
x
x
$K_1(\frac{\|x-z_1\|}{\sigma})$
$K_2(\frac{\|x-z_2\|}{\sigma})$
$K_M(\frac{\|x-z_M\|}{\sigma})$

Fig. 3. Quadratic ReLU network with no more than four neurons using shortcuts in each layer is universal approximator. The red shortcuts are used for passing the input and the green shortcuts for aggregating the outputs of different modules.

Proof. As Fig. 3 shown, because Lemma 4 makes sure that the quadratic ReLU network can approximate function $K(\frac{\|x-z_i\|}{\sigma}), i = 1, 2....$ We use residual connections (red lines) to make the network modules that represent different $K(\frac{\|x-z_i\|}{\sigma}), i = 1, 2...$ and we use another groups of residual connections (green lines) to aggregate the outputs of these modules, such that quadratic ReLU network can represent any function from the family S_K at any accuracy. By triangle relationship, we conclude that residual quadratic ReLU network with no more than four neurons in each layer is universal approximator.

Compared with the results in Lin and Jegelka (2018) and Lu et al. (2017), our Corollary 2 presents another special width-bounded universal approximator, which is slimmer than that of Lu et al. (2017) and has sparser shortcuts than that of Lin and Jegelka (2018). The upper limit of the width required by our width-bounded universal approximator is 4, while the counterpart in Lu et al. (2017) is $n + 4$ given an $n-$input. Although only one neuron is demanded in each layer in Lin and Jegelka (2018), much denser residual shortcuts are employed as the trade-off. In our work, only sparser residual connections are needed to integrate different blocks.

3.3. Theorem 3

Key idea for proving theorem 3: For universal approximation by neural networks, the current mainstream strategies are all based on piecewise approximation in terms of L_1, L_∞, or other distance measures. For the purpose of piecewise approximation, the functional space is divided into numerous hypercubes, which are intervals in the one-dimensional case, according to a specified accuracy to approximate a target function in every hypercube. With quadratic neurons and the trick that two ReLU neurons can organically execute a linear operation, we can instead use a global approximating method that is much more efficient. At the same time, the quadratic network structure is neither too wide nor too deep, which can be regarded as a size-bounded universal approximator, in contrast to what we have introduced before. What is more, aided by Algebraic Fundamental theorem, our novel proof reveals the uniqueness and facility of our proposed quadratic networks that cannot be elegantly made by networks with quadratic activation. More importantly, Theorem 3 gives a first-of-the-kind global universal approximator, facilitating feature representations in deep learning.

First, we show any univariate polynomial of degree N can be exactly expressed by a quadratic network with a complexity of $\mathcal{O}(log_2(N))$ in depth. Next we refer the result (Light & Cheney, 1985) regarding Hilbert's thirteen problem that multivariate functions can be represented with a group of separable functions, and then finalize the proof.

Lemma 5. *With ReLU as activation function, any univariate polynomial of degree N can be perfectly computed by a quadratic network with depth of $\mathcal{O}(\log_2(N))$ and width of no more than N.*

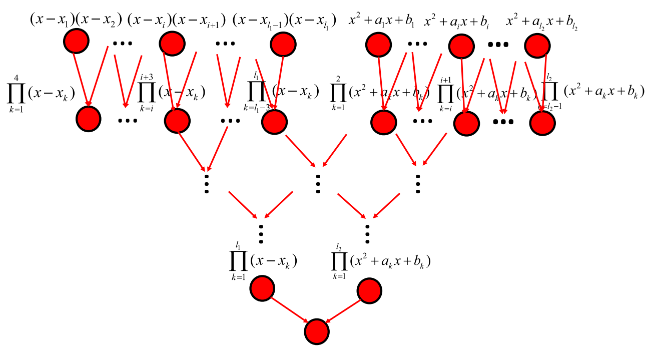

$(x-x_1)(x-x_2)$
$(x-x_i)(x-x_{i+1})$
$(x-x_{l_1-1})(x-x_{l_1})$
$x^2+a_1x+b_1$
$x^2+a_ix+b_i$
$x^2+a_{l_2}x+b_{l_2}$
$\prod_{k=1}^{4}(x-x_k)$
$\prod_{k=i}^{i+3}(x-x_k)$
$\prod_{k=l_1-3}^{l_1}(x-x_k)$
$\prod_{k=1}^{2}(x^2+a_kx+b_k)$
$\prod_{k=i}^{i+1}(x^2+a_kx+b_k)$
$\prod_{l_2-1}^{l_2}(x^2+a_kx+b_k)$
$\prod_{k=1}^{l_1}(x-x_k)$
$\prod_{k=1}^{l_2}(x^2+a_kx+b_k)$

Fig. 4. Quadratic network approximating a univariate polynomial according to the Algebraic Fundamental Theorem.

Proof. According to the Algebraic Fundamental Theorem (Remmert, 1991), a general univariate polynomial of degree N can be expressed as $P_N(x) = C\prod_i^{l_1}(x - x_i)\prod_j^{l_2}(x^2 + a_jx + b_j)$, where $l_1 + 2l_2 = N$. The ReLU activation has an important property: $f(x) = ReLU(f(x)) - ReLU(-f(x))$, which is often resorted in the literature, such as Ye, Han, and Cha (2018). The network we construct is shown in Fig. 4 (Each neuron actually represents two parallel neurons to implement a linear operation). Every two neurons are grouped in the first layer to compute $(x - x_i)$, $(x - x_i)(x - x_{i+1})$ or $x^2 + a_ix + b_i$, then the second layer merely uses half the number of neurons in the first layer to combine the outputs of the first layer. By repeating such a process, the quadratic network with the depth of $O(log_2(N))$ can exactly represent $P_N(x)$.

The following lemma shows that any univariate continuous function $f(x)$ on $[0, 1]$ can be approximated with a Bernstein polynomial up to any accuracy, which is the Bernstein version of the proof for the Weierstrass theorem.

Lemma 6. *Let $f(x)$ is a continuous function over* $[0, 1]$, *we have*

$$\lim_{n\to+\infty} B_n(x) = f(x)$$

Proof. It is well known; please refer to Jeffreys and Jeffreys (1988).

Corollary 2. *Any continuous univariate function supported on* $[0, 1]$ *can be approximated by a quadratic network with ReLUs up to any accuracy.*

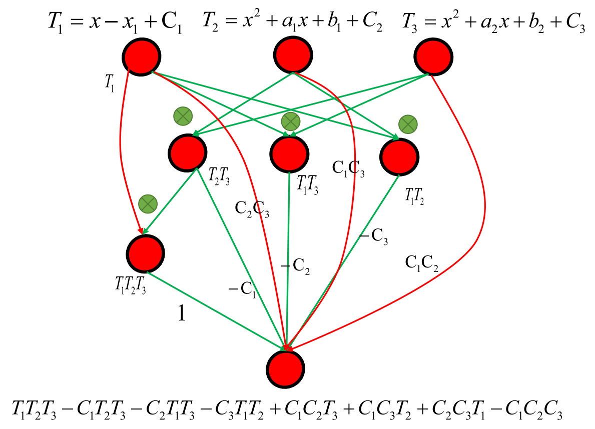
Lemma 7. *Every continuous n-variable function on* $[0, 1]^n$ *can be represented in the form:*

$$f(x_1, \dots, x_n) = \sum_{i=1}^{2n+1} g_i(\sum_{j=1}^{n} h_{ij}(x_j))$$

Proof. This is a classical theorem made by Kolmogorov and his student Arnold. Please refer to Light and Cheney (1985).

Proof of Theorem 3. Combining Lemmas 5–7, it can be shown that the ReLU-embedded quadratic network is a universal approximator, and importantly it realizes a universal approximation in a global manner.

Let us look at the following numerical example to illustrate our novel quadratic network based factorization method. First, 100 instances were sampled of a function, $g(x) = (x^2 + 1)(x - 1)(x^2 + 1.7x + 1.2)$ in $[-1,0]$. Instead of taking opposite weights



$T_1 = x - x_1 + C_1$
$T_2 = x^2 + a_1x + b_1 + C_2$
$T_3 = x^2 + a_2x + b_2 + C_3$
T_1
T_2T_3
T_1T_3
C_1C_3
T_1T_2
C_2C_3
$-C_3$
$-C_2$
$-C_1$
C_1C_2
$T_1T_2T_3$
1
$T_1T_2T_3 - C_1T_2T_3 - C_2T_1T_3 - C_3T_1T_2 + C_1C_2T_3 + C_1C_3T_2 + C_2C_3T_1 - C_1C_2C_3$

Fig. 5. Quadratic network with shortcuts to learn the soft factorization of an exemplary univariate polynomial.

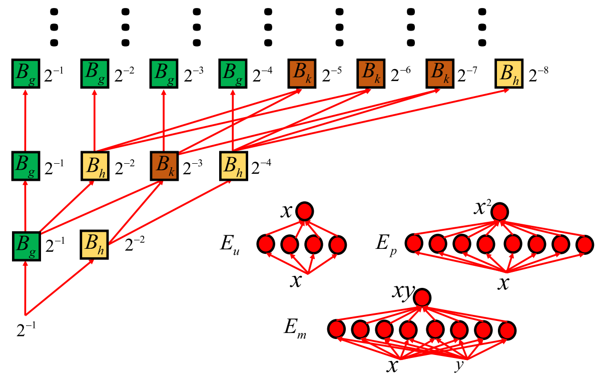


Fig. 6. Quantized quadratic ReLU networks with weight 2^{-1} and -2^{-1} to approximate any number with an error bound of 2^{-t}.

and biases to create linearity as used in proof for clarity, here we have incorporated shortcuts to make our factorization method trainable in terms of adaptive offsets. Using the ReLU activation function, we trained a four-layer network 4-3-1-1 with shortcuts to factorize this polynomial, as shown in Fig. 5. The parameters in connections marked by green symbols are fixed to perform multiplication. The shortcut connections and vanilla connection are denoted by green and red lines respectively. The neurons in the first layer will learn the shifted factors T_1, T_2, T_3, with unknown constant offsets C_1, C_2, C_3. The whole network will be combined in pairs to form T_1T_2, T_2T_3, T_1T_3 in the next layer. Then, the neurons in the third layer will multiply T_1 and T_2T_3 to obtain $T_1T_2T_3$. Finally, the neuron in the output layer is a linear one that will be trained to undo the effect of the constant offsets aided by the shortcuts. We trained the network to learn the factorization by initializing the parameters multiple times, with the number of iterations 600 and the learning rate 2.0e−3. The final average error is less than 0.0051. In this way, the function was learned to be $g'(x) = (1.0117592x^2 + 0.02033176x + 1.0040002)(0.00819827x^2 + 0.9977611x - 1.0023365)(1.0113174x^2 + 1.7042247x + 1.1885077)$, which agrees well with the target function $g(x) = (x^2 + 2)(x - 1)(x^2 + 1.7x + 1.2)$.

Mathematically, we can handle the general factorization representation problem as follows. Let us denote $A_i = m_i, A_{ij} = m_im_j, A_{ijk} = m_im_jm_k, \ldots, A_{123\ldots n} = \prod_i^n m_i$, which are generic factors for $i = 1, 2, \ldots, n$, the question becomes if $\prod_i^n(m_i + C_i) - \prod_i^n m_i$ can always be represented as the linear combination of $A_0, A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(n-1)}$ and a constant bias? If the answer is positive, then our above-illustrated factorization method can be extended for factorization of any univariate polynomial. Here we offer a proof by mathematical induction.

For $n = 1$, we have $m_1 + C_1 - m_1 = C_1$. Assume that we can represent $\prod_i^p(m_i + C_i) - \prod_i^p m_i$ with a linear combination of $A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(p-1)}$ and a constant, denoted as $f(A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(p-1)})$. Then, $\prod_i^{p+1}(m_i + C_i) - \prod_i^{p+1} m_i = (m_{p+1} + C_{p+1})(\prod_i^p(m_i + C_i) - \prod_i^p m_i) + C_{p+1}\prod_i^p m_p = (m_{p+1} + C_{p+1})f(A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(p-1)}) + C_{p+1}\prod_i^p m_p$. Clearly, the above terms $m_{p+1}f(A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(p-1)})$, $C_{p+1}\prod_i^p m_p$ are both linear terms of $A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots p}$ and $C_{p+1}f(A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(p-1)})$ is a linear representation of $A_i, A_{ij}, A_{ijk}, \ldots, A_{123\ldots(p-1)}$ plus a constant bias as well. Combining these together, we immediately have a desirable linear representation of $\prod_i^{p+1}(m_i + C_i) - \prod_i^{p+1} m_i$ that concludes our proof.

3.4. Theorem 4

Key idea for proving theorem 4: It is notable that a quadratic neuron internally realizes addition and multiplication operations in a unique way, which enables quantized quadratic neurons as building blocks for construction of an arbitrary function in a top-down way. That is the key mechanism by which the saving of $\mathcal{O}\left(\lambda\left(log^{\frac{1}{\lambda-1}+1}(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$, $\lambda \geq 2$ is achieved in terms of the number of weights. Fig. 6 clearly demonstrates our heuristics.

Lemma 8. *A connection with a weight $w \in [-1, 1]$ can be approximated by a ReLU network with $\lambda \geq 2$ distinct weights, satisfying that (i) the ReLU network is equivalent to weight w' with error no more than 2^{-t}, specifically, $|w - w'| < 2^{-t}$; (ii) the depth is $\mathcal{O}[log(t)]$; (iii) the width is $\mathcal{O}(t)$; and (iv) the number of weights is at most $\mathcal{O}(t)$.*

Proof. We first consider $\lambda = 2$ (we select two different weights as $2^{-1}, -2^{-1}$), $w \geq 0$ and $x \geq 0$, and will relax the constraints later. Setting $w = 0$ directly leads to an empty operation. There are three other necessary operations perfectly represented by a ReLU quadratic network: E_u (unity operation) E_p (power operation) and E_m (multiplication), which take 4, 8, 8 neurons respectively, as shown in Fig. 6. Our goal is to approximate w in accuracy of 2^{-t}. Let us define

$$K_c \equiv \{2^{-1}, 2^{-2}, \ldots, 2^{-(t-1)}, 2^{-t}\}.$$

With a homogeneous weight 2^{-1}, all the members in K_c can be represented by the composition of E_u, E_p and E_m operations. Because a binary expansion of W_c can denote any integral multiple of 2^{-t} like a numeral system with the radix of 2^{-t}, which is equivalent to represent any number with an error bound of 2^{-t}. To implement a binary expansion, one more layer with unity and empty operations is needed. Therefore, the overall quantized quadratic network has $log(t) + 1$ layers, with no more than $24t$ weights.

Regarding the situation of $w < 0$, we are able to construct -1 with -2^{-1} as the final layer. To relax the assumption $x \geq 0$, we can use two parallel sub-networks that take x and $-x$ respectively. Due to the gate property of ReLU, the output sign of a sub-network can be flipped in the last layer. In the case of $\lambda > 2$, because our construction is top-down, it is interesting that $\frac{1}{2}$ and $-\frac{1}{2}$ are enough for the construction. We can assign two weights as $\frac{1}{2}$ and $-\frac{1}{2}$ respectively among λ distinct weights. Therefore λ does not affect the final complexity.

Proof of Theorem 4: The proof consists of the following two steps. First, we utilize f' to approximate $f \in \mathcal{F}_d^n$, and then we construct f' precisely with our quantized ReLU quadratic networks.

For $\mathbf{m} = (m_1, \ldots, m_d) \in \{0, 1, \ldots, N\}^d$, $\psi_{\mathbf{m}}(\mathbf{x})$ is defined as:

$\psi_{\mathbf{m}}(\mathbf{x}) = \Pi_{k=1}^{d} h(3Nx_k - 3m_k)$,

where N is a constant and

$$h(x) = \begin{cases} 1 & |x| \le 1 \\ 2 - |x| & 1 \le |x| \le 2 \\ 0 & |x| \ge 2 \end{cases}$$

$\psi_{\mathbf{m}}(\mathbf{x})$ forms a partition of the unity on $[0, 1]^d$: $\sum_{\mathbf{m}} \psi_{\mathbf{m}}(\mathbf{x}) \equiv 1$, where $\mathbf{x}$ is supported on $[0, 1]^d$. Note that the support of $\psi_{\mathbf{m}}(\mathbf{x})$ is $\{\mathbf{x} : |x_k - \frac{m_k}{N}| < \frac{1}{N}, \forall k\}$. For any $\mathbf{m}$, there is a corresponding order $n-1$ Taylor expansion of the function f at $x = \frac{\mathbf{m}}{N}$:

$$Q_{\mathbf{m}}(\mathbf{x}) = \sum_{\mathbf{n}:|\mathbf{n}|<n} \frac{D^n f}{\mathbf{n}!}\Big|_{\mathbf{x}=\frac{\mathbf{m}}{N}} (x - \frac{\mathbf{m}}{N})^{\mathbf{n}},$$

where $\mathbf{n}! = \Pi_{k=1}^{d} n_k!$, and $(x - \frac{\mathbf{m}}{N})^{\mathbf{n}} = \Pi_{k=0}^{d}(x_k - \frac{m_k}{N})^{n_k}$. Furthermore, we define $Q'_{\mathbf{m}}(\mathbf{x}) = \sum_{\mathbf{n}:|\mathbf{n}|<n} \gamma_{\mathbf{m},\mathbf{n}}(x - \frac{\mathbf{m}}{N})^{\mathbf{n}}$, where $\gamma_{\mathbf{m},\mathbf{n}}$ denotes some integral multiple of $\frac{1}{n}(\frac{d}{N})^{n-|\mathbf{n}|}$ that is the closest to $\frac{D^n f}{\mathbf{n}!}|_{\mathbf{x}=\frac{\mathbf{m}}{N}}$. Thus, the approximation of f is constructed as:

$$f'(\mathbf{x}) = \sum_{\mathbf{m}} \psi_{\mathbf{m}} Q'_{\mathbf{m}}(\mathbf{x})$$

Hence, we have

$$\begin{aligned}
&\sup_{\mathbf{x}\in[0,1]^d} |f(\mathbf{x}) - f'(\mathbf{x})| \\
&= \sup_{\mathbf{x}\in[0,1]^d} |\sum_{\mathbf{m}} \psi_{\mathbf{m}}(f(\mathbf{x}) - Q'_{\mathbf{m}}(\mathbf{x}))| \\
&\le \sum_{\mathbf{m}} \sup_{\mathbf{x}\in\{|x_k - \frac{m_k}{N}|<\frac{1}{N},\forall k\}} |f(\mathbf{x}) - Q'_{\mathbf{m}}(\mathbf{x})| \\
&\le 2^d \max_{\mathbf{x}\in\{|x_k - \frac{m_k}{N}|<\frac{1}{N},\forall k\}} |f(\mathbf{x}) - Q_{\mathbf{m}}(\mathbf{x})| \\
&+ 2^d \max_{\mathbf{x}\in\{|x_k - \frac{m_k}{N}|<\frac{1}{N},\forall k\}} |Q'_{\mathbf{m}}(\mathbf{x}) - Q_{\mathbf{m}}(\mathbf{x})|
\end{aligned}$$

$$\begin{aligned}
&\le \frac{2^d d^n}{n!}(\frac{1}{N})^n \max_{\mathbf{n}:|\mathbf{n}|=n} ess\sup_{\mathbf{x}\in[0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})| \\
&+ 2^d \sum_{\mathbf{n}:|\mathbf{n}|<n} \max\left(|\frac{D^n f}{\mathbf{n}!} - \gamma_{\mathbf{m},\mathbf{n}}|\right)(x - \frac{\mathbf{m}}{N})^{\mathbf{n}} \\
&\le \frac{2^d d^n}{n!}(\frac{1}{N})^n (\frac{1}{N})^n + \frac{2^d}{n}\left((\frac{d}{N})^n + \cdots + (\frac{d}{N})^1 (\frac{d}{N})^{n-1}\right) \\
&\le 2^d (\frac{d}{N})^n (1 + \frac{1}{n!}) \le 2^{d+1}(\frac{d}{N})^n
\end{aligned}$$

Therefore, if we choose $N \ge (\frac{2^{d+1}d^n}{\epsilon})^{\frac{1}{n}}$, for $\epsilon \in (0, 1)$, any $f \in \mathcal{F}_{d,n}$ can be approximated by f' with an error bound ϵ. We rewrite

$$\begin{aligned}
f'(\mathbf{x}) &= \sum_{\mathbf{m}\in\{0,1,\ldots,N\}^d} \sum_{\mathbf{n}:|\mathbf{n}|<n} \gamma_{\mathbf{m},\mathbf{n}} \psi_{\mathbf{m}}(x - \frac{\mathbf{m}}{N})^{\mathbf{n}} \\
&= \sum_{\mathbf{m}\in\{0,1,\ldots,N\}^d} \sum_{\mathbf{n}:|\mathbf{n}|<n} \gamma_{\mathbf{m},\mathbf{n}} f'_{\mathbf{m},\mathbf{n}}(\mathbf{x}),
\end{aligned}$$

where $f'_{\mathbf{m},\mathbf{n}}(\mathbf{x}) = \psi_{\mathbf{m}}(x - \frac{\mathbf{m}}{N})^{\mathbf{n}}$. $f'(\mathbf{x})$ is the linear combination of no more than $d^n(N+1)^d$ terms of $f'_{\mathbf{m},\mathbf{n}}(\mathbf{x})$. By Lemma 8, with $t = log(\frac{nN^n}{d^n})$ we can use a sub-network of $\mathcal{O}[log(t)]$ to represent the constant weights $\gamma_{\mathbf{m},\mathbf{n}}$. Then, the remaining work is to construct $f'_{\mathbf{m},\mathbf{n}}(\mathbf{x})$. N is chosen as 2^l, where l is some large integer, such that all $\frac{m_i}{N}$ can be obtained precisely in the way of Lemma 8.

The strategy of using a quadratic network to construct f' is illustrated in Fig. 7. Table 1 presents the complexities of individual blocks of interest, which helps us to analyze the complexity of the whole network. Although the whole complexity can be expressed in an explicit way, here we use the $\mathcal{O}$ notation for clarity and simplicity. Let N_d and N_w be the network depth and the number of weights. According to Table 1, we have $N_d = \mathcal{O}(log(t) + log(N))$ and $N_w = \mathcal{O}\left(N^d log(t) + N^d log(N)\right)$. Because $t = log(\frac{nN^n}{d^n})$ and $N = (\frac{2^{d+1}d^n}{\epsilon})^{\frac{1}{n}} = \mathcal{O}\left((\frac{1}{\epsilon})^{\frac{1}{n}}\right)$, substituting them into the estimates results in

$$N_d = \mathcal{O}\left(log(log(\frac{1}{\epsilon})) + log(\frac{1}{\epsilon})\right)$$

$$N_w = \mathcal{O}\left(log(log(\frac{1}{\epsilon}))(\frac{1}{\epsilon})^{\frac{d}{n}} + log(\frac{1}{\epsilon})(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$$

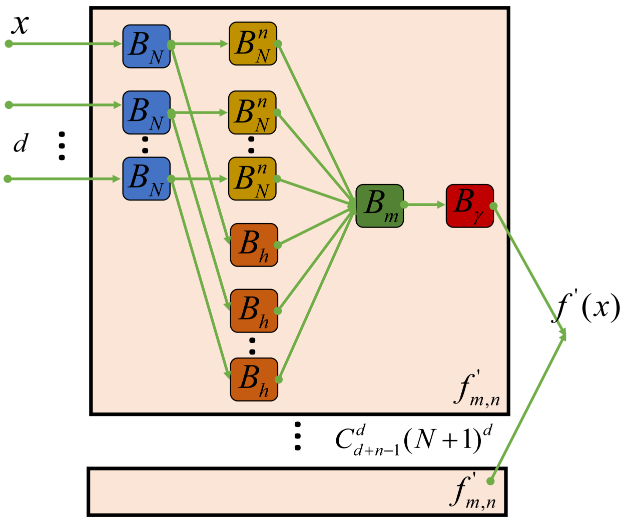


Fig. 7. Strategy of using a quantized quadratic network to implement $f'(\mathbf{x})$.

Table 1
Descriptions of different building blocks.

Block	Operation	Width	Depth
B_N	Construct $\frac{1}{N}, \ldots, \frac{N-1}{N}$	N	$log(N)$
B_N^n	Construct $(x_k - m_k)^{n_k}$	n	$log(n)$
B_h	Implement $h(x)$	12	1
B_m	Multiply d terms	$2d$	$log(2d)$
B_γ	Construct γ	t	$log(t)$

As we know, the quantization of the network (Han, 2016) is an effective and robust way for model compression, even a binarized network performs well in some meaningful tasks. However, the reason why quantization is robust and useful was rarely studied. Ding, Liu, Xiong, and Shi (2019) reported in their Theorem 1 that to obtain an approximation error ϵ, the upper bound for a quantized ReLU network is delineated by

$$\mathcal{O}\left(\lambda\left(log^{\frac{1}{\lambda-1}+1}(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}} + log(\frac{1}{\epsilon})(\frac{1}{\epsilon})^{\frac{d}{n}}\right), \quad (4)$$

where λ is the number of distinct weights. Because it is convenient for quadratic neurons to implement multiplication, our quantized ReLU quadratic network considerably reduces the upper bound to $\mathcal{O}\left(log\left(log(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}} + log(\frac{1}{\epsilon})(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$. It is observed that $\mathcal{O}\left(log\left(log(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$ is significantly lower than $\mathcal{O}\left(\lambda\left(log^{\frac{1}{\lambda-1}+1}(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$, therefore the saving made by quantized quadratic network is almost in an order of $\mathcal{O}\left(\lambda\left(log^{\frac{1}{\lambda-1}+1}(\frac{1}{\epsilon})\right)(\frac{1}{\epsilon})^{\frac{d}{n}}\right)$. More interesting, the complexity bound achieved by quantized quadratic networks is independent of the value of λ, as explained in our proof.

4. Discussions and conclusion

The rational for Theorem 1 is easily understandable. Since it is commonly known that a conventional network with one hidden layer is a universal approximator, the best thing we can do to justify the quadratic network is to find a class of functions that can be more efficiently approximated with a quadratic network than with a conventional network. By the first theorem, the quadratic network is indeed more efficient than the conventional counterpart.

The previous studies demonstrate that any function of $n-$ variables that is not constant along any direction cannot be well represented by a fully-connected ReLU network with no more than $n-1$ neurons in each layer (Lu et al., 2017). However, breaking this network width bound, our Theorem 2 states that when a radial function is not constant along any direction, the network width 4 is sufficient for a ReLU quadratic network to approximate the function accurately. Since it is more convenient to represent a nonlinear function with more nonlinear neurons, the weakness of conventional networks can be addressed by quadratic networks. Theorem 2 implies a slimmer and/or sparser width-bounded universal approximator.

Our Theorem 3 is inspiring that a general polynomial function can be exactly expressed by a quadratic network via ReLU activation in through a data-driven network-based algebraic factorization. Although the third theorem assumes the ReLU activation function, considering ReLU is arguably the most important activation function, it makes a great sense in illustrating the capacity of quadratic networks. Most importantly, Theorem 4 shows that quadratic networks are advantageous in terms of expressibility and efficiency. By calculating a quadratic function of input variables, quadratic neurons create second-order terms. The second-order terms serve as nonlinear building blocks that are different from non-linearity generated only through nonlinear activation, and clearly preferred from the perspective of the Algebraic Fundamental Theorem. Accurately forming a polynomial allows global computation and high efficiency as compared to the popular piece-wise approximation with conventional neurons in a deep network. In our quadratic network construction, both the depth and width are limited, while all previous universal approximators based on ReLU are either too wide or too deep to achieve a rather high approximation accuracy. This fact suggests that quadratic networks could potentially minimize the number of neurons and free parameters for superior machine learning performance. It is surprising that although every quadratic neuron possesses a number of parameters three times as many as that of the corresponding conventional neuron, quadratic networks can use less parameters than the conventional counterparts in important cases, such as the presentation of radial functions and the approximation to multivariate polynomials.

Interestingly, our results (Corollary 1 and Theorem 3) imply that quadratic networks are highly adaptive. For example, while a quadratic network can do piece-wise approximation by degenerating into conventional networks, a generic quadratic network is capable of approximating any function in a global way by implementing basic algebraic factors. Also, quadratic networks can mimic RBF networks via "wavelet" analysis, as indicated by Corollary 1.

As we know, deep learning generalizes well in many applications. Some researchers questioned that quadratic networks may tend to over-fitting because the added neural complexity may not be necessary. With our above proved theorems, we refute that since quadratic networks *de facto* are more compact and more effective than the conventional counterpart in major circumstances, quadratic networks could improve the generalization ability, according to Occam's razor principle. Since general features are either quadratic or can be expressed as combinations or compositions of quadratic features, quadratic deep networks will be universal, instead of being restricted to the circumstances where only quadratic features are relevant.

In conclusion, we have analyzed the approximation ability of the quadratic networks and proved four theorems, suggesting a great potential of quadratic networks in terms of expressive efficiency, unique representation, compact structure and computational capacity. An important direction for our future research will be focused on evaluating the performance of deep quadratic networks in typical applications.

Acknowledgement

We thank the valuable comments from anonymous reviewers. This work was partially support by NIH/NCI under award numbers R01CA233888 and R01CA237267, and NIH/NIBIB under award number R01EB026646.

References

Andoni, A., Panigrahy, R., Valiant, G., & Zhang, L. (2014). Learning polynomials with neural networks. In *ICML*.
Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, *39*(3), 930–945.
Bianchini, M., & Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, *25*, 1553–1565.
Blondel, M., et al. (2016). Higher-order factorization machines. In *NIPS*.
Cohen, N., Sharir, O., & Shashua, A. (2016). On the expressive power of deep learning: A tensor analysis. In *COLT*.
Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 30–42.
Ding, Y., Liu, J., Xiong, J., & Shi, Y. (2019). On the universal approximability and complexity bounds of quantized ReLU neural networks. In *ICLR*.
Du, S. S., & Lee, J. D. (2018). On the power of over-parametrization in neural networks with quadratic activation. arXiv preprint arXiv:1803.01206.
Eldan, R., & Shamir, O. (2016). The power of depth for feedforward neural networks. In *COLT*.
Fan, F., Cong, W., & Wang, G. (2017). Generalized back propagation algorithm for training second-order neural networks. *International Journal for Numerical Methods in Biomedical Engineering*, http://dx.doi.org/10.1002/cnm.2956.
Fan, F., Cong, W., & Wang, G. (2018). A new type of neurons for machinelearning. *International Journal for Numerical Methods in Biomedical Engineering*, *34*(2).
Fan, F., Shan, H., & Wang, G. (2019). Quadratic autoencoder for low-dose CT denoising. arXiv preprint arXiv:1901.05593.
Fan, F., & Wang, G. (2018). Fuzzy logic interpretation of artificial neural networks. arXiv preprint arXiv:1807.03215.
Giles, C. L., & Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, *26*, 4972–4978.
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge: MIT press.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.
Han, S. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.
Hong, Y., Kim, J., Chen, G., Lin, W., Yap, P. T., & Shen, D. (2019). Longitudinal prediction of infant diffusion MRI data via graph convolutional adversarial networks. *IEEE Transactions on Medical Imaging*, https://ieeexplore.ieee.org/document/8691605.
Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.
Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*.
Jeffreys, H., & Jeffreys, B. S. (1988). Weierstrass's theorem on approximation by polynomials and extension of Weierstrass's approximation theory. In *Methods of mathematical physics* (3rd ed.). Cambridge, England, US: Cambridge University Press.

Kainen, P. C., Kurkova, V. V., & Sanguineti, M. (2012). Dependence of computational models on input dimension: Tractability of approximation and optimization tasks. *IEEE Transactions on Information Theory*, *58*(2), 1203–1214.

Kon, M. A., & Plaskota, L. (2000). Information complexity of neural networks. *Neural Networks*, *13*(3), 365–375.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Krotov, D., & Hopfield, J. (2018). Dense associative memory is robust to adversarial inputs. *Neural Computation*, *30*(12), 3151–3167.

Kumar, A., et al. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

Kurkova, V., & Sanguineti, M. (2017). Probabilistic lower bounds for approximation by shallow perceptron networks. *Neural Networks*, *91*(2017), 34–41.

Liang, S., & Srikant, R. (2017). Why deep neural networks for function approximation? In *ICLR*.

Light, W. A., & Cheney, E. W. (1985). *Approximation theory in tensor product spaces*. Springer-Verlag.

Lin, H., & Jegelka, S. (2018). ResNet with one-neuron hidden layers is a universal approximator. arXiv preprint arXiv:1806.10909.

Lin, H. W., Tegmark, M., & Rolnick, D. (2017). Why does deep and cheap learning work so well? *Journal of Statistical Physics*, *168*(6), 1223–1247.

Livni, R., et al. (2014). On the computational efficiency of training neural networks. In *NIPS*.

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The expressive power of neural networks: a view from the width. In *NIPS*.

Mhaskar, H. N., & Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, *14*(6), 829–848.

Minsky, M. L., & Papert, S. (1969). *Perceptrons*. Cambridge: MIT Press.

Park, J., & Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computation*, *3*(2), 246–257.

Remmert, R. (1991). The fundamental theorem of algebra. In *Numbers* (pp. 97–122). New York, NY: Springer.

Schmitt, M. (2000). Lower bounds on the complexity of approximating continuous functions by sigmoidal neural networks. In *NIPS*.

Silver, D., et al. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *CVPR*.

Szegedy, C., et al. (2015). Going deeper with convolutions. In *CVPR*.

Szymanski, L., & McCane, B. (2014). Deep networks are effective encoders of periodicity. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(10), 1816–1827.

Telgarsky, M. (2016). Benefits of depth in neural networks. In *COLT*.

Tsapanos, N., Tefas, A., Nikolaidis, N., & Pitas, I. Neurons with paraboloid decision boundaries for improved neural network classification performance. *IEEE Transactions on Neural Networks and Learning Systems*, *99*, 1–11.

Wang, G. (2016). A perspective on deep imaging. *IEEE Access*, *4*, 8914–8924.

Ye, J. C., Han, Y., & Cha, E. (2018). Deep convolutional framelets: A general deep learning framework for inverse problems. *SIAM Journal on Imaging Sciences*, *11*(2), 991–1048.

Zhang, Y., & Yu, H. (2018). Convolutional neural network based metal artifact reduction in X-ray computed tomography. *IEEE Transactions on Medical Imaging*, *37*(6), 1370–1381.