Universal Approximation with Quadratic Deep Networks

Fenglei Fan, Student Member, IEEE, and Ge Wang, Fellow, IEEE

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 second-order 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. Regarding this, we ask three 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 a quadratic network can approximate much more efficiently than a conventional network? (3) Does a quadratic network give a new insight into universal approximation? Our main contributions are the three theorems shedding light upon these three questions and demonstrating the merits of a quadratic network in terms of expressive efficiency, unique capability, and compact architecture respectively.

Index Terms—Artificial Intelligence, machine learning, deep learning, artificial neural networks, second-order networks, approximation theory

I. INTRODUCTION

Over recent years, deep learning has become the mainstream approach for machine learning. Since AlexNet [1], increasingly more advanced neural networks [2-6] are being proposed, such as GoogleNet, ResNet, DenseNet, GAN and variants, to enable practical performance comparable to or beyond what the human delivers in computer vision [7], speech recognition [8], language processing [9] game playing [10], medical imaging [11-13], 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 generic 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 [14]. That is, the network can be extremely wide. With the emergence of deep neural networks, studies have been performed on theoretical benefits of these deep models over shallow ones [15-22]. One way [15, 19] 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 [16] 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 is reported in [22] that ResNet with one single neuron per layer is a universal approximator. Moreover, it is demonstrated in [15] 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.

In our previous studies [23-25], 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. For example, a single quadratic neuron can implement the famous XOR logic [23]. Furthermore, each quadratic neuron can be viewed as a generalized fuzzy logic gate, and a deep quadratic network is nothing but a deep fuzzy logic system [25]. Note that high-order neurons [26-27] were taken into account in the early stage of artificial intelligence, but they are not connected into deep networks and suffer from combinatorial explosion of parameters due to high order terms. In contrast, our quadratic neurons of a limited number of parameters (tripled that of a conventional neuron) performs the utility of high-order neurons in a deep network architecture. For this type of novel quadratic deep networks, we developed a general backpropagation algorithm [24] to make them trainable, paving the way for quadratic neurons to be applied for deep learning.

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. In this paper, we ask three 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? (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? If the answers to these questions are favorable, quadratic networks should be significantly more powerful in many machine learning tasks.

In this paper, we present three theorems addressing the above three 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 capability, and compact architecture respectively. We answer the first question with the first theorem, given the network with only one hidden layer, 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, any 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 in the same structure. Finally, with the third theorem we provide a new insight into the universal presentation from the perspective of the Algebraic Fundamental Theorem. Without introducing complex numbers, univariate polynomials of degree n can be uniquely factorized as multiplication of quadratic terms. Since a quadratic network can represent any univariate polynomial in this way, by the Weierstrass theorem and the Kolmogorov theorem that multivariate functions can be represented through summation of univariate functions, we can approximate any multivariate function with a well-structured quadratic neural network, justifying the universal approximation power of the quadratic network.

To further elaborate, several additional comments are in order. For the first theorem, it is commonly known that a conventional neural network with one hidden layer is a universal approximator. Hence, the best thing we can do to justify the quadratic network is to find a class of functions that can be more efficiently approximated by a quadratic network than with a conventional network. As related to the second question, the precious 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 [16]. Breaking this network width lower bound, our second theorem states that when a radial function is not constant in any direction, the network width 4 is sufficient for a quadratic network to approximate the function accurately. Our third theorem is most interesting that a general polynomial function can be exactly expressed by a quadratic network in a novel way of data-driven network-based algebraic factorization. Implied by the Weierstrass theorem, the quadratic network is a universal approximator. Different from the analyses on conventional neural networks with ReLU activation as universal approximators [14,16] that are of either infinitely wide or infinitely deep, for a given polynomial model the depth and width of our quadratic network are both finite for a perfect representation of a given order multivariate polynomial, which we call the size-bounded universal approximator. Notably, by factoring a generic polynomial globally a quadratic network can match the functional structure effectively and then be efficiently trained, avoiding brute-force piece-wise linear fitting into a target function.

There are prior papers related to but different from our contributions [41-45]. Motivated by a need for more powerful activation, Livni et al. [41] proposed to use the quadratic activation: $\sigma(z) = z^2$ in the neuron. Despite somewhat misleading in its name, networks with quadratic activation 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 [41], 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 [42], Du et al. showed that over-parametrization and weight decay are instrumental to the optimization aided by quadratic activation. [43] 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. [44] presents that $O(\log k)$ layers of binary units and $O(\log^2 k)$ ReLU units can approximate $f(x) = \sum_{i=0}^{n} a_i x^i$ with closeness of $\epsilon$. In contrast, our Theorem 3 is based on the Algebraic Fundamental Theorem to provide an exact representation of any finite-order polynomial. [45] is on factorization machine (FM) dedicated to combine high order features, clearly different from the polynomial factorization we propose to perform using a quadratic network.

In the next section, we introduce some preliminaries. In the third section, we describe our three theorems, and include the corresponding lemmas and proofs. Numerical examples are also used for illustration. Finally, in the last section we discuss relevant issues and conclude the paper with some conjectures and future work.

**II. Preliminaries**

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

$$h(x) = \sum_{i=1}^{n} w_i x_i + b_r (\sum_{i=1}^{n} w_{1i} x_i + b_y) + \sum_{i=1}^{n} w_{2i} x_i^2 + c$$

$$= (w_r x^T + b_r)(w_y x^T + b_y) + w_b (x^2)^T + c,$$

(1)
where $x$ denotes the input vector, and the other variables are defined in [23]. Our definition of the quadratic function 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, [28] 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 function represented by a one-hidden-layer conventional network is as follows:

$$f_1(x) = \sum_{i=1}^{k} t_i \sigma_i \left( \sum_{i=1}^{n} w_{ix_i} + b_i^1 \right) = \sum_{i=1}^{k} t_i \sigma_i (w_i^1 x + b_i^1). \quad (2)$$

In contrast, the functions represented by a one-hidden-layer quadratic networks is:

$$f_2(x) = \sum_{l=1}^{k} t'_l \sigma_l [ (w'_l^1 x^T + b'_l^1) (w'_l^2 x^T + b'_l^2) + w'_l (x^2)^T + c_l^1 ]. \quad (3)$$

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

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

$$||f(x - y)|| \leq L||x - y||$$

**Radial Function:** A radial function only depends on the norm of its input vector, generically denoted as $f(||x||)$. The functions mentioned in Theorems 1 and 2 are all radial functions. By its nature, the quadratic neuron is well suited for modeling of a radial function. On the other hand, a general function can be regarded as the mixtures of radial functions, such as radial basis function networks can be used for universal approximation.

**Fourier Transform:** For a function $f : \mathbb{R}^d \to \mathbb{R}$, its Fourier transform is:

$$F(w) = \int_{\mathbb{R}^d} \exp[-2\pi i (w \cdot x)] f(x) \, dx$$

**Euclidean Unit-volume Ball:** In a $d$-dimensional space, let $R_d$ be the radius of a Euclidean ball $B_d$ such that $R_d B_d$ has the unit volume.

**The First Kind Bessel Function:** The first kind Bessel function $J_\nu$ of an order $\nu$ is denoted as:

$$J_\nu(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m! \Gamma(m + \nu + 1)} \left( \frac{x}{2} \right)^{2m + \nu}$$

**Lesbegue-integrable Function:** A non-negative measurable function is called Lesbegue-integrable if its Lesbegue integral is finite. For an arbitrary measurable function, it is integrable if its positive part and negative part are both Lesbegue integrable.

**Bernstein Polynomial:** $l_{n,m} = C_n^m x^m (1 - x)^{n-m}$, $0 \leq m \leq n$. The $n$-th Bernstein polynomial of $f(x)$ in $(0,1)$ is defined as:

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

The Bernstein Polynomials are used before to prove the Weierstrass theorem.

**III. Three Theorems**

First, we present three theorems, and then give their proofs.

**Theorem 1:** For an activation function $\sigma(\cdot)$ with $|\sigma(x)| \leq K(1 + |x|^k)$, $x \in \mathbb{R}$, and constants $K, \kappa > 0$, and for some universal constants $c > 0$, $C > 0$, $C' > 0$, $c_\sigma > 0$, there exist a probability measure $\mu$ and a radial function $g : R^d \to \mathbb{R}$, where $d > C$, that is bounded on $[-2,2]$ and supported on $||x|| \leq C\sqrt{d}$ satisfying:

1. $g$ can be approximated by a single-hidden-layer quadratic network with $C' c_\sigma d^{2.5}$ neurons, which is denoted as $f_2$.
2. For every function $f_1$ expressed by a single-hidden-layer conventional network with at most $c e^{cd}$ neurons, we have:

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

for some positive constant $\delta$.

**Theorem 2:** For any Lesbegue-integrable radial function $f(||x||) : [-1,1]^n \to \mathbb{R}$, and any $\epsilon$, there exists a fully-connected ReLU quadratic network with no more than four neurons in each layer such that the corresponding function $F(x)$ expressed by this network satisfies:

$$\int_{||x|| \in [R_1,R_2]} |f(||x||) - F(x)| \, dx \leq \epsilon.$$

**Theorem 3:** For any multivariate polynomial $P(x_1,...,x_i,...,x_N) = \sum_{k=1}^{M} \prod_{j=1}^{N} x_j^{n_j(k)}$, which degrees of input components $x_1,...,x_i,...,x_N$ in the $k$-th term are $n_1(k), n_2(k),...,n_i(k),...,n_N(k)$ respectively. there is a quadratic network of width $\sum_{j=1}^{N} 2 \max_k \{n_j(k)\} + 2M$ and depth $\max_k \{n_j(k)\} + N$ that computes $P(x_1,...,x_i,...,x_N)$ exactly.

**A. Proof of Theorem 1**

**Key Idea for Proving Theorem 1:** The proof combines the observation from [15] and the utility of quadratic neurons to approximate a radial function. For convenience and consistency, we use some definitions and notations in [15]. The form of functions represented by a single-hidden-layer conventional network is $f(x) = \sum_{i=1}^{k} t_i \sigma_i (w_i x + b_i^1)$. It is observed that the distribution of the Fourier transform of $f(x)$ is supported on a finite collection of lines. The support covered by the finite lines are sparse in the Fourier space, especially for a high dimensionality 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 found in [15]:

\[ \hat{g}(x) = \sum_{i=1}^{N} \varepsilon_i g_i(\|x\|), \]

where \( \varepsilon_i \in \{-1,1\} \), \( N \) is a polynomial function of \( d \), \( g_i(\|x\|) = 1(\|x\| \in \Delta_i) \) are radial indicator functions over disconnected intervals. Although the constructed \( \hat{g} \) is hard to approximate by a conventional network, it is easy to approximate by a quadratic network, because Eq. (1) contains square terms that can be rewritten as \( h(x) = \beta \alpha (\|x\|^2 + \gamma_0) \) to compute the norm and then the 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 \( \hat{g}(x) \) is discontinuous, and cannot be perfectly expressed by a neural network with continuous activation functions. Here we use a probability measure \( \mu \). With \( \mu \), \( \hat{g} \) can be approximated by \( f(x) \) represented by a network in the sense of \( \hat{E}_{x \sim \mu}(f(x) - \hat{g}(x))^2 \).

Proposition 1 in [15] has demonstrated that \( \hat{g} \) cannot be well approximated by a single-hidden-layer conventional network with a polynomial number of neurons. We put his proposition here as Lemma 1 for readability and coherence of our paper.

**Lemma 1:** For a fixed dimension \( d \), suppose that \( d > C \), \( \alpha > C \) and \( N \geq C \alpha^{1.5}d^2 \), and \( k \) an integer satisfying \( k \leq c_{\delta}d^2 \) with universal constants \( c, C > 0 \), there exists a function \( \hat{g}(x) = \sum_{i=1}^{N} \varepsilon_i g_i(x) \), where \( \varepsilon_i \in \{-1,1\} \), \( i = 1, 2, ..., N \) such that for any function of the form: \( f_1(x) = \sum_{l=1}^{k} t^l \sigma_l(w^lx + b^l) \) with \( \sigma_l(x) \leq K(1 + |x|^\kappa) \) for some \( K, \kappa \), we have

\[ ||f_1 - \hat{g}||_{L_2(\mu)} \geq \delta/\alpha, \]

where \( \delta \) is a universal constant.

To illustrate \( \hat{g} \) is approximable with a quadratic network, we know from Lemma 12 of [15] that a continuous Lipschitz function \( g \) can approximate \( \hat{g} \), 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 a proper activation function \( \sigma \), there is a constant \( c_\sigma \geq 1 \) (depending on \( \sigma \) and other parameters) such that for any \( L \)-Lipschitz function \( f: \mathbb{R} \rightarrow \mathbb{R} \), which is constant outside a bounded interval \( [r, R] \), and any \( \delta \), there exist scalars \( \{\alpha_i, \beta_i, \gamma_i\} \), \( i = 1, \ldots, w \), \( w \leq c_{\sigma}(\frac{R-r)\delta}{\delta} \) with which we have

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

satisfies

\[ \sup_{x \in \mathbb{R}} |f(\|x\|) - h(x)| < \delta. \]

Proof: Without loss of generality, we assume that the nonlinear activation function is ReLU: \( \sigma(z) = \max\{0, z\} \). However, our proof is also applicable to other nonlinear activation functions.

If \( (R-r)\delta < \delta \), then \( h(x) \) can be trivially constructed by setting it to be a 0 function. Otherwise, we have \( R-r \geq \delta/L \). We assume that there is an integer \( m \) satisfying \( m\delta/L < R-r \leq (m+1)\delta/L \), dividing \( [r, R] \) into \( (m+1) \) intervals \( [r, r + \delta/L], [\delta/L, 2\delta/L], \ldots, [i\delta/L, (i+1)\delta/L], ..., [r+m\delta/L, R] \). We set \( \beta_i = 1, \gamma_i = (r+i\delta/L)^2, i = 0, \ldots, m, \gamma_{m+1} = R^2, a = h(r), \) and

\[ \alpha_i = \left( f(\sqrt{\gamma_i+1}) - f(\sqrt{\gamma_0}) \right)/(\gamma_i+1 - \gamma_0) \]

Then, by such construction, the lemma holds. Here, the number of the used neurons is \( w \), which is no more than \( \left[ \frac{(R-r)\delta}{\delta} \right] + 1 = c_{\sigma}(\frac{R-r)\delta}{\delta} \), where \( \lfloor \cdot \rfloor \) is the floor function.

As we know, Eq.(1) represents a quadratic network with a single hidden layer. Lemma 2 confirms that a Lipschitz radial function \( f \) can be well expressed by such a single-hidden layer network.

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

\[ ||f_2 - \hat{g}||_{L_2(\mu)} \leq \frac{\sqrt{3}}{Cd^2} + \delta. \]

Proof: In Lemma 2, we make the following substitutions: \( R = 2\alpha \sqrt{\delta}, r = \alpha \sqrt{\delta}, L = N, h = g \). Thus, \( g(\|x\|) \) is expressible by the a single-hidden-layer quadratic network with at most \( c_{\sigma}N\alpha \sqrt{d} \) neurons. Coupled with Lemma 12 of [15], 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 \( \alpha = C, N = C\alpha^{1.5}d^2 \), we have

\[ ||f_1 - \hat{g}||_{L_2(\mu)} \geq \delta_1 \]

Let \( \delta \leq \frac{\delta_1}{2} - \frac{\sqrt{3}}{Cd^2} \), to approximate \( \hat{g} \) we need the number of quadratic neurons being at most

\[ c_{\sigma}Na\sqrt{d}/\delta = c_{\sigma}C^{3.5}d^{2.5}/\delta \leq C'c_{\sigma}d^{2.5} \]

such that

\[ ||f_2 - \hat{g}||_{L_2(\mu)} \leq \frac{\sqrt{3}}{Cd^2} + \delta \leq \frac{\delta_1}{2}. \]

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

**Classification Example:** To demonstrate the exponential difference between the conventonal and quadratic networks claimed by Theorem 1, 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
A neuron that computes the square of norm, then we have
the output of the function and approximate any univariate function accurately.

Also, two neurons are needed to store the truncated-parabola function. In every interval, a truncated-parabola function can approximate a piecewise constant function in the $L_1$ sense. Also, two neurons are needed to store the truncated-parabola function by the pair of neurons. We encapsulate these four neurons in total three layer as a module. By connecting these modules properly, we can express a piecewise trapezoid-like function and approximate any univariate function accurately.

Assume that the input variable is $x$, an interval is $||x|| \in [a_i, a_{i+1}]$, the network utilizes the ReLU function as the activation function, and $t = ||x||$, we use $R_{i,j}$ to represent the output of the $j$-th neuron in the $i$-th layer (we ignore the neuron that computes the square of norm), then we have

$$R_{i+2,1} = \left\{ \begin{array} {l}
\frac{b_i}{C_i} (t^2 - a_i^2) (a_{i+1}^2 - a_i^2) \\
\quad t \in [a_i, a_i + \delta (a_{i+1} - a_i)] \\
\frac{b_i}{C_i} (t^2 - a_i^2) (a_{i+1}^2 - t^2) \\
\quad t \in [a_i + \delta (a_{i+1} - a_i), \sqrt{a_{i+1}^2 + a_i^2} - (a_i + \delta (a_{i+1} - a_i))^2, a_{i+1}] \\
\end{array} \right.$$  
(4)

where we have:

$$a_{i+1} - \sqrt{a_{i+1}^2 + a_i^2} - (a_i + \delta (a_{i+1} - a_i))^2 < \delta (a_{i+1} - a_i)$$

B. Theorem 2

Key Idea for Proving Theorem 2: It was proved in [16] 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, when such a function is radially defined, it becomes feasible to approximate the function by a quadratic network with width=4, which breaks the lower width bound claimed in [16].

To compute a radial function, we need to find the norm and then evaluate the function at the norm. With a quadratic neuron, the norm is naturally found. With respect to the norm, the radial function is intrinsically univariate. Therefore, 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.

The trick of approximation by a deep conventional network is adopted here to study a deep quadratic network, in the same spirit to approach a function via composition layer by layer. Specifically, we use one quadratic neuron for the squared norm of an input vector, and a neuron to form a truncated-parabola function as a building block of the radial function. In every interval, a truncated-parabola function can approximate a piecewise constant function in the $L_1$ sense. Also, two neurons are required to store the truncated-parabola function by the pair of neurons. We encapsulate these four neurons in total three layer as a module. By connecting these modules properly, we can express a piecewise trapezoid-like function and approximate any univariate function accurately.
In this way, each module will correspond to a truncated-parabola function over a unique interval. By decreasing $\delta$, our truncated-parabola function would approximate the target piecewise constant function with increasing accuracy. Since piecewise constant functions can approximate any Lipschitz-integrable function in the $L_1$ sense, a deep quadratic network with four neurons in each layer can express any Lipschitz-integrable radial function. Our final network structure is shown in Fig. 3. A blue rectangle is a neuron, and a red rectangle forms a layer. Depending on the interval width, our network can be very deep.

In the following, we will prove Theorem 2 formally. In the proof, the closeness between two functions is measured in the $L_1$ sense, and for convenience we partially adopted the notations used in [16].

**Notations:** For a Lesbegue-integrable radial function $f(||x||)$, suppose that $||x||$ is supported on $[R_1, R_2]$, we define:

$$f^+(t) = \begin{cases} \max\{f(t), 0\} & t \in [R_1, R_2] \\ 0 & t \notin [R_1, R_2] \end{cases}$$

$$f^-(t) = \begin{cases} \max\{-f(t), 0\} & t \in [R_1, R_2] \\ 0 & t \notin [R_1, R_2] \end{cases}$$

$S^+ = \{(x, y), ||x|| \in E, 0 < y < f^+(||x||)\}$

$S^- = \{(x, y), ||x|| \in E, 0 < y < f^-(||x||)\}$

Because $f(t)$ is Lesbegue integrable, $S^+$ and $S^-$ are measurable. Then, there exits a series of cylindrical tubes $J_{+,i}, J_{-,i}$ having the property:

$$m(S^\pm \triangle \bigcup_i J_{\pm,i}) < \frac{\epsilon}{4} \quad (5)$$

where $m(\cdot)$ is the Lesbegue measure. For cylindrical tubes $J_{\pm,i}$, we assume $J_{\pm,i} = X_{\pm,i} \times [0, b^\pm_i]$, where $X_{\pm,i} = \{x_{a_i} < ||x|| < a_{i+1}\}$, and $b^\pm_i \geq 0$ is the corresponding height of a cylindrical tube. Then, we define the corresponding indicator function as $\phi_{\pm,i} = 1\{x \in X_{\pm,i}\}$, and we have the following lemma.

**Lemma 4:** For any $f(||x||)$ and $\epsilon$, the weighted sum of the indicator functions, which represent the cylindrical tubes, satisfies:

$$\int_{||x|| \in [R_1, R_2]} |f(||x||) - (\sum_i b^+_{i,\phi_{+,i}} - \sum_i b^-_{i,\phi_{-,i}})|| dx \leq \frac{\epsilon}{2}.$$  

Proof: Lemma 4 is evident by Eq. (5).

By stacking the modules gradually, we reconstruct the target function over more and more intervals until the approximation is complete. As we discussed, the function generated by our composite three layer modules is positive. If the corresponding cylindrical tube is positive, then it can be used to approximate $b^+_{i,\phi_{+,i}}$ directly. If the corresponding cylindrical tube is negative, we need to subtract the output of the module when its output is transmitted into the next layer. The network will eventually produce a function with many trapezoid-like pieces. We denote such a function is $F(x)$, $F^+||x||$ and $F^-||x||$ are the positive and negative parts of $F(x)$, which are actually $K^+$ and $K^-$. Then, we have

$$\int_{||x|| \in [R_1, R_2]} |F^+(||x||) - \sum_i b^+_{i,\phi_{+,i}}| dx$$

$$+ \int_{||x|| \in [R_1, R_2]} |F^-(||x||) - \sum_i b^-_{i,\phi_{-,i}}| dx$$

$$\leq (2\delta) \int_{||x|| \in [R_1, R_2]} \left(\sum_i b^+_{i,\phi_{+,i}} + \sum_i b^-_{i,\phi_{-,i}}\right) dx.$$  

Because $f(x)$ is Lesbegue integrable, its cover $\sum_i b^+_{i,\phi_{+,i}}$ is Lesbegue-integrable as well. Therefore, $\sum_i b^+_{i,\phi_{+,i}} < \infty$. Let us denote $\sum_i b^+_{i,\phi_{+,i}} + \sum_i b^-_{i,\phi_{-,i}} = C$ and $\delta = \frac{\epsilon}{C}$, we can approximate $\sum_i b^+_{i,\phi_{+,i}}$ by our quadratic network up to a given accuracy $\epsilon$.

$$\int_{||x|| \in [R_1, R_2]} |F^+(||x||) - \sum_i b^+_{i,\phi_{+,i}}| dx$$

$$+ \int_{||x|| \in [R_1, R_2]} |F^-(||x||) - \sum_i b^-_{i,\phi_{-,i}}| dx \leq \frac{\epsilon}{2} \quad (6)$$

**Lemma 5:**

$$\int_{||x|| \in [R_1, R_2]} |F(||x||) - (\sum_i b^+_{i,\phi_{+,i}} - \sum_i b^-_{i,\phi_{-,i}})| dx \leq \frac{\epsilon}{2}.$$  

Proof: Because of Eq. (6), we apply the triangle inequality:

$$\int_{||x|| \in [R_1, R_2]} |F(||x||) - (\sum_i b^+_{i,\phi_{+,i}} - \sum_i b^-_{i,\phi_{-,i}})| dx$$

$$\leq \int_{||x|| \in [R_1, R_2]} |F^+(||x||) - \sum_i b^+_{i,\phi_{+,i}}| dx$$

$$+ \int_{||x|| \in [R_1, R_2]} |F^-(||x||) - \sum_i b^-_{i,\phi_{-,i}}| dx$$

$$\leq \frac{\epsilon}{2}.$$  

**Proof of Theorem 2:** For any radial function $f(||x||)$ whose support is bounded and any given closeness $\epsilon$, there is a network with no more than four neurons in each layer of the quadratic network as shown in Fig. 3. Applying the
can be regarded as a size-bounded universal approximator, network structure is neither too wide nor too deep, which be much more efficient. At the same time, the quadratic can instead use a global approximating method, which can according to a specified accuracy to approximate a target hypercubes, which are intervals in the one-dimensional case, approximation, the functional space is divided into numerous strategies are all based on piecewise approximation in approximation by neural networks, the current mainstream C. Theorem 3

\[
\begin{align*}
 f(\|x\|) &= \cos(\frac{3\pi}{200}\|x\|^2 + \frac{3}{2}) , \text{ where } x \text{ is supported on } ||x|| \in [0, 10\sqrt{2}], \text{ as shown in Fig. 4(left). We approximate } f(||x||) \text{ by stacking three modules with 9 layers in total, as shown in Fig. 4(right). The resultant function } F(||x||) \text{ is:} \\
 F(||x||) &= -(1 - (1 - \frac{1}{1022.53} \|x\|^2 - \frac{200}{3} - \|x\|^2))^+ + (1 - (1 - \frac{1}{1042.53} \|x\|^2 - \frac{400}{3} - \|x\|^2))^+ + (1 - (1 - \frac{1}{1092.53} \|x\|^2 - \frac{400}{3} - \|x\|^2))^+ + (1 - (1 - \frac{1}{1092.53} \|x\|^2 - \frac{400}{3} - \|x\|^2))^+ \\
 \text{which divides } [0, 10\sqrt{2}] \text{ into three pieces } [0, \sqrt{\frac{200}{3}}), [\sqrt{\frac{200}{3}}, \sqrt{\frac{400}{3}}], [\sqrt{\frac{400}{3}}, 10\sqrt{2}], \text{ and } b_1 = b_2 = b_3 = 1. \text{ If the support of } f(||x||) \text{ is divided into more intervals, the closeness of } f(||x||) \text{ and } F(||x||) \text{ will be further improved.}
\end{align*}
\]

C. 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 \(L_1\), \(L_\infty\), or other distances. For such 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, we can instead use a global approximating method, which can be 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 [14][16]. In [14], a single-layer network may have an infinite width. On the other hand, in [16] the network width is restricted to be no more that \(d + 4\) but network depth goes infinity. What’s 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.

First, we show any univariate polynomial of degree \(N\) can be exactly expressed by a quadratic network with a complexity of \(O(\log_2(N))\) in depth. Next we refer the result [37] regarding Hilbert’s thirteen problem that multivariate functions can be represented with a group of separable functions, and then finalize the proof.

\textbf{Lemma 6:} Any univariate polynomial of degree \(N\) can be perfectly computed by a quadratic network with depth of \(O(\log_2(N))\) and width of no more than \(N\).

\textbf{Proof:} According to Algebraic Fundamental Theorem [38], a general univariate polynomial of degree \(N\) can be expressed as
\[
\begin{align*}
P_N(x) &= \prod_{i=1}^{N}(x-x_i) \prod_{j=1}^{2}(x^2 + a_j x + b_j),
\end{align*}
\]
where \(l_1 + 2l_2 = N\). Similarly, we set \(O_{ij}\) as the output of the \(j\)-th unit in the \(i\)-th layer. The network construct is shown in Fig. 7. Every neuron in the first layer computes \((x-x_i)(x-x_{i+1})\) or \(x^2 + a_j x + b_i\), then the second layer merely use half the number of neurons as that of the first layer to combine the outputs of the first layer. By repeating such a process, with the depth of \(O(\log_2(N))\) the quadratic network can exactly represent \(P_N(x)\).

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

\textbf{Lemma 7:} Let \(f(x)\) is a continuous function over \([0, 1]\), we have
\[
\lim_{n \to +\infty} B_n(x) = f(x)
\]
\textbf{Proof:} It is well known; please refer to [30].

\textbf{Corollary 1:} Any continuous univariate function supported on \([0, 1]\) can be approximated by a quadratic network up to any accuracy.

\textbf{Lemma 8:} Every continuous \(n\)-variable function on \([0, 1]^n\) can be represented in the form:
\[
f(x_1, ..., x_n) = \sum_{i=1}^{2^{n+1}} g_i(\sum_{j=1}^{n} h_{ij}(x_j))
\]
\textbf{Proof:} This is a classical theorem made by Kolmogorov and his student Arnold. Please refer to [39].

\textbf{Corollary 2:} The quadratic network is a universal approximator.

\textbf{Proof:} Referring to Lemmas 6, 7 and 8, it can be shown that the quadratic network is a universal approximator. Specifically, assuming that \(h_{i,j}\) is a polynomial of degree
$N_{ij}$ and $g_i$ is a polynomial of degree $N_i^g$, the size of a quadratic network can be estimated to approximate $f(x_1, ..., x_n)$ well. The representation of $\sum_{i=1}^n h_{ij}$ requires a network with a width of $\sum_{i=1}^n N_{ij}$ and a depth of $\max_j(\log_2 N_{ij}^h)$. Then, the representation of $g_i$ demands an additional configuration with a width of $N_i^g$ and a depth of $\log_2 N_i^g$. Therefore, the structure used for $g_i(\sum_{j=1}^n h_{ij}(x_j))$ is of width $\max\{\sum_{j=1}^N N_i^h_j, N_i^g\}$ and depth $\max_j(\log_2 N_{ij}^h) + \log_2 N_i^g$. Integrating all $g_i(\sum_{j=1}^N h_{ij}(x_j))$ terms in parallel, the final quadratic network architecture will be of at most width $\sum_{i=1}^{2n+1} \max\{\sum_{j=1}^n N_i^h_j, N_i^g\}$ and depth $\max_j\{\log_2 N_{ij}^h + \log_2 N_i^g\}$.

Also, aided by the concept of partially separable functions, the complexity of the quadratic network can be further reduced, such as in the case of computing an $L^th$ separable function. By the $L^th$ separable function, we mean that $f(x_1, ..., x_n)$ is $L^th$ separable defined as follows:

$$f(x_1, ..., x_n) = \sum_{l=1}^L \prod_i \phi_{i,l}(x_i).$$

In practice, almost all continuous functions can be represented as $L^th$ separable functions, which are of low ranks at the same time.

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 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. 6. 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 $g(x) = (1.0117592x^2 + 0.02033176x + 1.0040002)(0.00819827x^2 + 0.9977611x - 1.0023365)(1.0113174x^2 + 1.70424247x + 1.1885077)$, 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 $A_1 = m_i, A_{ij} = m_im_j, A_{ijk} = m_im_jm_k, ..., A_{123...n} = \prod_i m_i$, which are generic factors for $i = 1, 2, ..., n$, the question becomes if $\prod_i (m_i + C_i) - \prod_i m_i$ can always be represented as the linear combination of $A_0, A_1, A_2, A_3, ..., A_{123...n-1}$ 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 $A_1, A_{ij}, A_{ijk}, ..., A_{123...n-1}$ and a constant, denoted as $f(A_1, A_{ij}, A_{ijk}, ..., A_{123...n-1})$. Then,

$$\prod_i^{p+1} (m_i + C_i) - \prod_i^{p+1} m_i = (m_{p+1} + C_{p+1})(\prod_i^p (m_i + C_i) - \prod_i^p m_i) + C_{p+1}\prod_i^p m_i = (m_{p+1} + C_{p+1})f(A_1, A_{ij}, A_{ijk}, ..., A_{123...n-1}) + C_{p+1}\prod_i^p m_i.$$

Clearly, the above terms $m_{p+1}f(A_1, A_{ij}, A_{ijk}, ..., A_{123...n-1})$ and $C_{p+1}\prod_i^p m_i$ are both linear terms of $A_1, A_{ij}, A_{ijk}, ..., A_{123...p}$ and $C_{p+1}f(A_1, A_{ij}, A_{ijk}, ..., A_{123...n-1})$ is a linear representation of $A_1, A_{ij}, A_{ijk}, ..., A_{123...n-1}$ 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.
IV. DISCUSSIONS AND CONCLUSION

Recently, major efforts have been made to develop and understand deep neural networks for not only high accuracy but also high robustness. To achieve this goal, many techniques [31-36] have been developed based on structure diversity, distilled knowledge, randomization, pruning/dropout, low-rank approximation, and so on. Some of these methods are dedicated to compress an established network model without compromising its performance significantly. Encouraged by our Theorems 1, which shows an exponential advantage of the quadratic network over the conventional counterpart in terms of the network complexity, we believe that there are important tasks particularly suitable for use of quadratic networks, leading to a significantly more compact architecture for comparable or better accuracy and robustness than the conventional network. Therefore, quadratic networks could be used in the setting with memory restriction, for example, in mobile applications or other embedded systems.

Representing multivariable functions in desirable ways is pivotal in applications of deep learning, including unsupervised, semi-supervised, and supervised learning [39]. From this point of view, we need to consider expressibility and complexity when evaluating the quality of a neural network [40]. Expressibility means how rich a family of functions is that can be expressed by the neural network. Complexity means the number of consumed neurons, which should be, in a good sense, as small as possible. Theorem 1 has shown that the complexity of the quadratic network is intrinsically lower than that of the conventional network in representing radial functions. Regarding "Expressibility", quadratic networks actually provide a new paradigm, as suggested by Theorem 2. Given a $n$-input function that is not constant along any direction, the conventional network with width less than $n-1$ fails to do good approximation in terms of the $L_1$ distance. We do not consider the failure surprising since such a $n$-input function is by definition on a $n$-dimensional manifold, and compressing it with less than $n$ neurons is subject to the dimensionality curse (in the opposite sense of the conventional meaning). In contrast, a quadratic network can only use four neurons per layer to approximate any multivariate radial function, which breaks the limit associated with the conventional network.

Most importantly, our Theorem 3 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 naturally. The second-order terms serve as nonlinear building blocks that are different from nonlinearity generated with nonlinear activation, and clearly preferred from the perspective of the Algebraic Fundamental Theorem. Accurately forming any polynomial allows global computation and high efficiency relative to the popular piece-wise approximation with conventional linear 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 can be either too wide or too deep to achieve a 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 the number parameters three times as many as that of a 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.

As we know, deep learning generalizes well in many applications. Some researchers questioned that quadratic networks may lead to overfitting 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 three theorems, suggesting a great potential of quadratic networks in terms of expressibility and efficiency. Future research will be focused on evaluating the performance of deep quadratic networks in typical applications.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks," In NIPS, 2012.
[2] C. Szegedy, et al., "Going deeper with convolutions," In CVPR, 2015.
[3] K. Simonyan, A. Zisserman. "Very deep convolutional networks for large-scale image recognition," arXiv preprint [arXiv:1409.1556], 2014.
[4] K. He, et al., "Deep residual learning for image recognition," In CVPR, 2016.
[5] G. Huang, et al., "Densely Connected Convolutional Networks," In CVPR, 2017.
[6] J. Goodfellow, et al., "Generative adversarial nets," In NIPS, 2014.
[7] C. Szegedy, et al., "Rethinking the inception architecture for computer vision," In CVPR, 2016.
[8] G. E. Dahl, D. Yu, L. Deng and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on audio, speech, and language processing," vol. 20, no. 1, pp. 30-42, 2012.
[9] A. Kumar, et al., "Ask me anything: Dynamic memory networks for natural language processing. In ICML, 2016.
[10] D. Silver, et al., "Mastering the game of Go without human knowledge," Nature, vol. 550, no. 7676, pp. 354, 2017.
[11] G. Wang, "A Perspective on Deep Imaging," IEEE Access, vol. 4, pp. 8914-8924, 2016.
[12] Y. Zhang and H. Yu, "Convolutional Neural Network based Metal Artifact Reduction in X-ray Computed Tomography," IEEE Trans. Med. Imaging, vol. 37, no. 6, pp.1370-1381, 2018.
[13] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, S. Mougiakakou, "Lung pattern classification for interstitial lung diseases using a deep convolutional neural network," IEEE Trans. Med. Imaging, vol.35, no.5, pp. 12071216, 2016.
[14] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators," Neural networks, vol. 2, no. 5, pp. 359-366, 1989.
