Fuzzy Logic Interpretation of Artificial Neural Networks

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

***Abstract* — Over past several years, deep learning has achieved huge successes in various applications. However, such a data-driven approach is often criticized for lack of interpretability. Recently, we proposed artificial quadratic neural networks consisting of second-order neurons in potentially many layers. In each quadratic neuron, a quadratic function is used in the place of the inner product in a traditional neuron, and then undergoes a nonlinear activation. With a single second-order neuron, any fuzzy logic operation, such as XOR, can be implemented. In this sense, any deep network constructed with quadratic neurons can be interpreted as a deep fuzzy logic system. Since traditional neural networks and second-order counterparts can represent each other and fuzzy logic operations are naturally implemented in second-order neural networks, it is plausible to explain how a deep neural network works with a quadratic network as the system model. In this paper, we generalize and categorize fuzzy logic operations implementable with individual quadratic neurons, and then perform statistical/information theoretic analyses of exemplary quadratic neural networks.**

***Index Terms* — Machine learning, artificial neural network (ANN), 2nd order neuron, quadratic network, fuzzy logic.**

I. Introduction

In the field of machine learning, artificial neural networks (ANNs), especially deep neural networks (CNNs), have recently achieved impressive successes in important applications including classification, unsupervised learning, prediction, image processing and analysis, as well as tomographic reconstruction [1-3]. However, a neural network is often considered as a black box without interpretability. Questions are often asked include what a neural network does in generating its output, and more importantly how a network can be designed and optimized. The deep learning researchers definitely want to have the model effective, efficient, explainable, and modifiable under guidelines. Lacking explainability/interpretability has become a primary obstacle to the wide-spread adaptation and further development of deep learning techniques [4]. Since understanding the mechanism of deep neural networks is challenging, over recent years extensive efforts have been made along this direction. Heuristically, existing methods can be categorized into three classes [5]. Based on the categorization of [5], we add one more class: advanced mathematic/physics methods, which are gaining traction recently. Specifically, we would group the existing methods as follows.

Hidden Neuron Analysis Methods: The hidden neuron analysis methods decipher the neural networks by interpreting the features that are learnt by constituting neurons individually and collectively [6-11]. For example, Yosinski *et al.* [8] visualized the activation patterns of neurons when Convnet processes an image or video. To understand the image representation by hidden neurons, Mahendran *et al.* [6] used the encoding of an image to reconstruct the image itself. Zhou *et al.* [10] proposed to label the hidden units by associating individual hidden units with visual semantic concepts. However, the hidden neuron analysis methods only offer qualitative insights, falling short of the optimization of the network structure or performance.

Model Mimicking Methods: Model mimicking methods [12-16] find the well-performed models that are more interpretable than existing models. For example, Wu *et al.* [12] utilized tree regularization to mimic a deep model and deliver a directly interpretable decision tree. Fan [13] proposed a generalized hamming network based on the fact that neurons calculate generalized hamming distance when bias is appropriately configured. Albeit seemingly reasonable, the model mimicking methods do not reveal the real mechanism based on which the existing models are successful. In addition, the mimicking models are relatively simple, it is hard to guarantee that these simplified models can imitate the deep neural networks satisfactorily.

Local Interpretation Methods: Local interpretation methods are dedicated to understanding the neural networks by perturbing the input and observing the output change of the network [17-21]. With robust analysis, training points can be identified that are most responsible for a prediction. Zhou *et al.* [21] found that global average pooling forms the attention of convolutional networks so as to provide a localized deep representation. P. W. Koh *et al.* [20] explained the prediction of a neural network by identifying most responsible points aided by the influence function in classical robustness statistics. However, the local interpretation methods do not offer direct insight into the inner working of a network and may be interfered by biases of the inputs.

Mathematic/Physics Methods: Mathematic/physics methods

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

Fenglei Fan and Ge Wang are with Department of Biomedical Engineering, Rensselaer Polytechnic Institute, Troy 12180, United States (fanf2@rpi.edu, wangg6@rpi.edu).

build a top-level connection between deep networks and advanced mathematical or physical theories in order to unlock the mystery of deep learning. Gu *et al.* 2017 [22] offered an elegant explanation of the adversarial mechanism of GAN from the viewpoint of optimal mass transportation. Dong *et al.* 2017 [23] established a correspondence between a deep neural network and ordinary differential equations to guide the design of a network with skip connections. In a perspective on deep imaging [24], it was suggested that "with deep neural networks, depth and width can be combined to efficiently represent functions with a high precision, and perform powerful multi-scale analysis, quite like wavelet analysis but in a nonlinear manner." It was recently shown [25] that convolutional autoencoders are nothing but a framelet representation scheme. In another study, the success of deep learning is attributed to fundamental physical and statistical principles [26]. In spite of being inspiring and in-depth, high-dimensionality and high nonlinearity are aggressively ignored in these explanations, which are actually the key ingredients of deep neural networks.

The current artificial neurons were inspired by the functionality of biological neurons. In biology, multiple types of neurons interact for various neural activities. In our earlier studies [27-28], we proposed to construct a neural network with quadratic neurons (in which data are integrated with a quadratic function, instead of an inner product). Pilot results demonstrated that quadratic networks enjoy the merits of a stronger representation ability and a more compact structure than the popular linear neurons. Since the inner product is replaced by the quadratic operation in a quadratic neuron, an individual quadratic neuron can implement not only the XOR logic but also a generalized fuzzy operation. Livni et al. [29] proposed to utilize a quadratic activation $\sigma(z) = z^2$ to replace the existing type of activation functions. At an increased computational cost, neurons with quadratic activation are still characterized by linear boundaries, restricting the expressive ability of neurons.

As engineers, we tend to interpret neural networks from the perspective of engineering. A modern computer operates on binary logic, and metaphorically called an electronic brain. As the counterpart of this metaphor, we consider the deep neural networks as an integrated system of fuzzy logic gates. Every fuzzy gate has a specific function. When each neuron is viewed as a specific fuzzy logic module, the overall neural network can be disentangled into these basic unites, thereby facilitating us to infer how features are extracted and processed by each layer, and transported between layers. Thus, quadratic fuzzy modules allow a top-level explanation according to matrix spectrum theory, by which each quadratic function module can be topologically delineated and identified by its eigen spectrum. Therefore, we claim that quadratic neural systems enjoy clear interpretability from the fuzzy logic perspective than the traditional counterparts.

As the first step toward our understanding of deep quadratic networks, in the next section we apply matrix spectrum analysis to characterize and categorize the fuzzy operations performed by second-order neurons. In the third section, we describe minima of a well-trained second-order network, investigate the loss change when the network goes wider, and offer insight into network complexity. Finally, we discuss relevant issues and conclude the paper.

II. Generalized Fuzzy Logic Operations

A. Generalized Fuzzy Logic

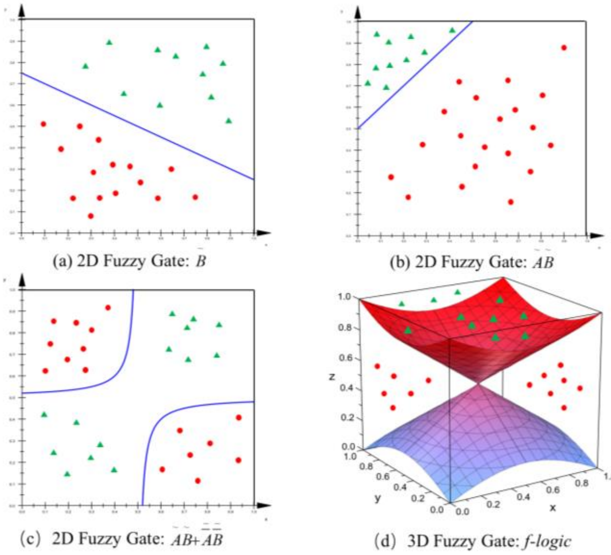
Aggregation refers to combine multiple values into a single value. Up to now, fuzzy aggregation [30] are extremely rich families of operations. Despite the flexibility in its mathematical expression, fuzzy aggregation is restricted by the range of logic values bounded in [0 1], which seems unnecessary because data should have different dynamic ranges as appropriate, which can be always normalized if needed. Here we view such an extension as "generalized fuzzy logic". Then, quadratic neurons in artificial neural networks exactly carry on generalized fuzzy logic aggregation operations, each of which is nothing but a fuzzy logic gate.

In our quadratic neurons [27], a quadratic function summates the multiplication of two inner products and one weighted norm term before feeding into a nonlinear activation function. For compact notation, we define $w_{0r} \coloneqq \mathrm{b}_1, w_{0g} \coloneqq b_2$ and $x_0 = 1$. Then, the output function is expressed as:

$$f(\vec{x}) = \left(\sum_{i=0}^{n} w_{ir}x_i\right)\left(\sum_{i=0}^{n} w_{ig}x_i\right) + \sum_{i=0}^{n} w_{ig}x_i^2$$
$$= \left(\sum_{i=1}^{n} w_{ir}x_i + b_1\right)\left(\sum_{i=1}^{n} w_{ig}x_i + b_2\right) + \sum_{i=1}^{n} w_{ig}x_i^2 + c \tag{1}$$

The condition $f(\vec{x}) = 0$ forms a decision boundary to separate the input vector space into two regions: the inputs from one region make the function negative, and those from the other region make it positive. The effect of the nonlinear excitation is to produce a fuzzy logic output in a proper range.

Now, let us briefly illustrate the generalized fuzzy logic operations of the conventional and quadratic neurons respectively with the 2D and 3D examples. In Fig. 1, the data cloud with two classes ("red circle" and "green triangle") are split by the boundary shaped by the first-order neurons in Fig. 1(a)-(b) and the second-order neurons in Fig. 1(c)-(d). We denote two variables in the input vector as A (horizontal axis) and B (vertical axis) respectively. If we assign the logic value "1" to "green triangle" points and "0" to "red circle" points, the neurons in Fig. 1(a)-(c) approximately perform logic operations $\tilde{B}$, $\tilde{A}\tilde{B}$ and $\widetilde{XOR}(\mathrm{A},\mathrm{B})$, respectively. In Fig. 1(d), the aggregation operation of three variables is implemented by a quadratic function *f*, which cannot be approximately described as a conventional logic gate. Conveniently, a generalized logic gate is denoted as a "*f*-logic gate".



(a) 2D Fuzzy Gate: $\bar{B}$

(b) 2D Fuzzy Gate: $\bar{A}\bar{B}$

(c) 2D Fuzzy Gate: $\bar{A}\bar{B}+\bar{\bar{A}}\bar{\bar{B}}$

(d) 3D Fuzzy Gate: *f-logic*

Fig. 1. Generalized fuzzy logic operations implemented using the first-order and second-order neurons respectively.

B. Categorization of Fuzzy Logic Gates by Quadratic Neurons

For simplicity, the generalized fuzzy operations performed by a quadratic neuron are written as a quadratic logic operation. Accordingly, a quadratic neuron is equivalently called quadratic fuzzy logic gate. To realize a target quadratic function, the quadratic neuron only uses approximate $3n$ parameters for n inputs. In the light of generalized fuzzy logic, every quadratic function implements a unique logic operation. Yet we can still find some high-level topological similarities that are shared by quadratic fuzzy logic gates. For this purpose, we comprehensively characterize quadratic fuzzy logic operations in light of matrix spectrum theory and then categorize them.

Given the input vector $\vec{x}$ with a dimensionality n, its function $f(\vec{x})$ is calculated according to Eq. (1). The parameters b_1, b_2 and c control offsets instead of the topological characteristics of the function in a high-dimensional space. Therefore, the decision boundary shape of the function defined in Eq. (1) is essentially the same as that of the following equation:

$$f'(\vec{x}) = \left(\sum_{i=1}^{n} w_{ir}x_i\right)\left(\sum_{i=1}^{n} w_{ig}x_i\right) + \sum_{i=1}^{n} w_{ig}x_i^2 \qquad (2)$$

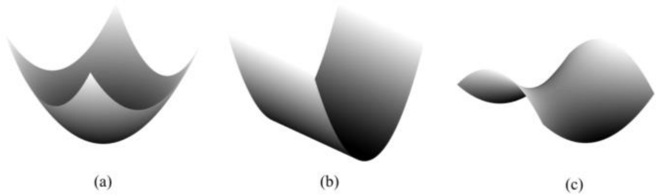


Fig. 2. Topological properties of the quadratic functions in two variables. (a) A parabolic surface, (b) an elongated valley, and (c) a saddle configuration, as characterized by the eigen values and vectors of the corresponding *J* matrices respectively.

In linear algebra, the quadratic function can be simplified into a quadratic form. That is, we can write the right-hand side of Eq. (2) as xJx^T, where J is an n by n real symmetric matrix with $J_{ii} = w_{ir}w_{ig}$ and $J_{ij} = J_{ji} = (w_{ir}w_{jg} + w_{jr}w_{ig})/2$. The topology of the quadratic function is dominated by the spectrum of J, especially by the sign of eigenvalues. Fig. 2 gives us examples to show how the signs of eigenvalues affect the landscape of a quadratic function of two variables. The signs of two eigenvalues are (+, +), (+, 0), (+, −) in Fig. 2(a)-(c) respectively. A "+" indicates that the functional value along the corresponding eigen axis will grow up to positive infinity. A "0" (or very small) eigenvalue suggests a valley along the corresponding axis. The interesting saddle surface is generated in the neighboring axes with opposite signs of eigenvalues.

Since the congruent matrix transformation will not change the index of inertia of J, we can categorize the quadratic fuzzy operations according to the positive, negative, and zero indices of inertia. Considering a quadratic function of n input variables, how many categories should we put the quadratic function into? This is a combinatorial problem that is equivalent to that "for $a + b + c = n$, how many nonnegative, unique integer solutions for (a, b, c) ." The answer is $(n + 2)(n + 1)/2$, which means that individual fuzzy logic operations can be grouped into $(n + 2)(n + 1)/2$ types. Take a four-inputs quadratic function as an example, there are theoretically 15 semi-topographic types of fuzzy logic operations. While in reality, eigenvalues seldom show up as pure zero but a positive or negative number with a relatively small norm compared with other eigenvalues. Hence, we split significant eigenvalues according to their signs, and consider the eigenvalues either positive or negative. Then the combinatorial problem is simplified as "for $a + b = n$, how many nonnegative, unique integer solutions for (a, b)." This time, the answer becomes $n + 1$ if we ignore any difference in permutation.

C. Characterization of Minima

The above-described categorization of quadratic fuzzy logic gates captures the intrinsic shape of the manifold defined by a neuron with an invariability under orthonormal transforms. However, this semi-topological picture is insufficient to describe the operation of the quadratic neuron comprehensively. In other words, although orthonormal transforms have no impact on the index of inertia, they determine the location and orientation of a decision boundary, generally changing the specific function of the neuron. *Nevertheless, topologically, neurons in the same class conducts more or less similar fuzzy aggregations*, and neurons in different classes behave much more differently. We underline that our eigen-characterization of the quadratic fuzzy gates is semi-quantitative in order to offer a top-level insight. Based on this motivation, we can develop a spectral analysis method to describe the properties of a neural network.

For a given well-trained quadratic network with L layer at some minimum, suppose that the number of input various at the l^{th} layer is $n(l)$, the number of possible types of spectra the neurons in the l^{th} layer may have are $(n(l) + 2)(n(l) + 1)/2$, orderly denoted as $S_i^l, i = 1,2, \dots (n(l) + 2)(n(l) + 1)/2$. We represent the total number of S_i^l in the l^{th} layer as $N(S_i^l)$. By summarizing $N(S_i^l)$, we obtain the spectral characterization of the whole network $S = \{N(S_i^l) | l = 1,2 \ldots, L,\ i = 1,2, \ldots (n(l)+2)(n(l)+1)/2\}$. In the perspective of fuzzy logic, the spectrum S for a given network minimum represents a particular combination of quadratic fuzzy gates implemented by all involved neurons. *In a good sense, S is an "ID" that helps us to distinguish the different minima,* facilitating solutions of important problems with neural networks. Please note that spectral analysis is made possible by the introduction of quadratic neural networks.

D. Robustness of Spectral Characterization

The training process for a network usually will not stop at the exact minimum point. Rather, the optimization trajectory will be probably terminated in a neighboring region of some minimum even after a large number of iterations. A potential concern is whether the spectrum S' of neighboring points will be different from the spectrum S of a given minimum or not. More mathematically, the concern is if the eigenvalues of a symmetric matrix are stable with respect to a small perturbation. Luckily, we have the following Hoffman and Wielandt Theorem [31] to guarantee the robustness of our spectral characterization method.

Let A, B are $n\ by\ n$ symmetric matrices, λ_i^A is the eigenvalue of A and λ_i^B is the eigenvalue of B, then there exists a permutation σ satisfying: $\sum_i^n |\lambda_i^A - \lambda_{\sigma(i)}^B| \le ||A-B||$, where $||\cdot||$ denotes the Frobenius Norm. This theorem shows that the eigen spectrum of a symmetric matrix is rather robust: as the optimization trajectory comes close to a minimum, the eigenvalues of each neuron will approach the corresponding eigenvalues associated with the minimum. Accordingly, the type of spectra of each neuron will also align with that of the minimum, which makes the spectrum S' of neighboring points consistent with the spectrum S.

With the above robustness property, in the next section we will demonstrate how our spectral analysis and categorization will help us meaningfully understand the characteristics of neural networks.

III. SPECTRAL ANALYSIS AND CATEGORIZATION

In the preceding section we have described how to characterize the quadratic fuzzy operations, which suggests a spectral analysis method to decode the neural network for interpretability. In this section, several applications of spectral analysis are presented: we first scrutinize the spectra of three different minima in a well-trained second-order network for recognition of Arabic digits from the benchmark dataset MNIST. Then, we study the loss change in terms of good minima as the network goes wider. Furthermore, we apply spectral analysis to test if it is correct that flat minima generalize well, and propose a new measure to evaluate the network structure. Finally, we discuss network training in terms of spectral entropy.

A. Spectra of Minima in Second-order Networks for MNIST

MNIST is one of the most popular datasets in the machine learning community. The network framework "LeNet-5" [32] achieved an excellent performance on MNIST. To show the interpretability of neural networks, we built a second-order network based on LeNet to analyze MNIST.

Specifically, we adopted two "convolutional" layers, two pooling layers, one fully-connected layer, and one softmax layer. Here, we borrowed the concept of convolution from traditional convolutional neural networks for convenience, although typically kernels in second-order networks do not represent linear convolutional operations. Actually, these kernels are shifted over an image/pixels as the convolution operations do. We set 32 3X3 "convolutional" kernels with stride 1X1 in the first "convolutional" layer, and 20 4X4 kernels for every channel in the second "convolutional" layer. Both the pooling layers perform 2X2 down-sampling with a stride 2X2. In our second-order network, all the neurons are second-order neurons with the activation function ReLU, except for the softmax layer which is not necessary to use second-order neurons for classification. All the weights and biases were initialized with the Tensorflow internal function "*tf.truncated_normal*" with the standard deviation=0.1. The loss function was defined as cross-entropy. We set the 55,000 iterations for training, where 20,000 iterations were done with a learning rate 10^{-4} and then the rate was lowered to $0.4 * 10^{-4}$ for the remaining part of the training process.

Previously, the LeNet-5 network utilized the technique of distortion, which is to create new instances for training by distorting existing images. LeNet-5 had error rates 0.95, 0.85, and 0.8 respectively in the case of no distortion, distortion, and huge distortion respectively. Our second-order network achieved a lower error rate 0.76 without data argumentation, which is the state of the art performance on MNIST. Then, we investigated the types of quadratic operations in our quadratic network. As shown in Fig. 3, we colored and inflated the neurons in the "convolutional" layers to present our interpretation of this well-trained second-order network. Due to the dominating role of convolution in the convolutional neural network, we only processed the neurons in the convolutional layers. In the conv1 and conv2 layers, the neurons took 9 and 16 inputs respectively. As we discussed before, the total number of types of quadratic operations are $n+1$ for n-input second-order neurons. Then, we saw 10 types of quadratic operations for the conv1 layer and 17 types of quadratic operations for the conv2 layer, as shown in Fig. 3. The colors of neurons indicate different types of quadratic operations performed by the neurons. The area of the neuron is proportional to the count of corresponding operations, as labeled inside the neuron. The subsequent squares denote the feature maps that are extracted by the corresponding quadratic operations. We blackened the feature maps from the conv2 layer because the feature maps were exclusively from the solitary neuron.

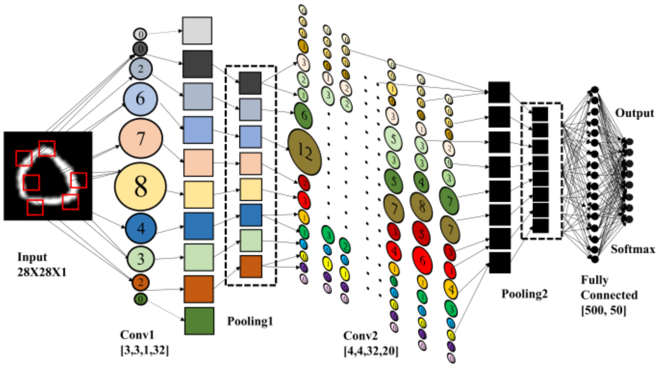


Fig. 3. Second-order network for recognition of Arabic digits from the MNIST dataset. The neurons in the “convolutional” layers are colored and inflated to demonstrate the types and frequencies of quadratic operations.

To evaluate the distribution of various quadratic neurons associated with a well-behaved minimum, we repeated the training process to have two other minima that achieved the state-of-the-art error rates 0.77 and 0.81 respectively. We counted the number of quadratic operations per each fuzzy gate type for Conv1 and Conv2 layers respectively, and obtained the graphs in Fig. 4. “Type-0” on the horizontal axis indicates the number of positive indices of inertia equal to 0, “Type-1” denotes the number of positive indices of inertia equal to 1, and so on.

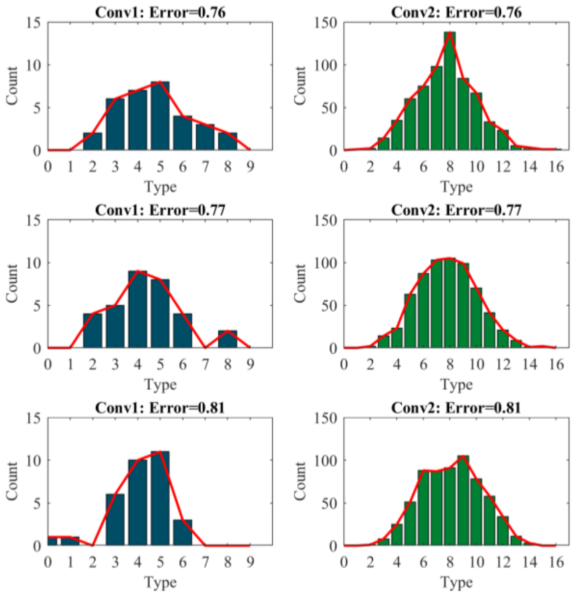


Fig. 4. Spectra of the number of positive indices for three minima respectively.

As shown in Fig. 4, the spectra of fuzzy logic systems associated with these three minima differ significantly in the first and second convolutional layers respectively. In the Conv1 layer of the three networks, the spectra are irregularly distributed, despite neurons of types 3-5 are dominant in all the cases. For example, the counts of “Types 3-5” in the network with error rate=0.81 took up 84.38% of all the neurons. In contrast, the spectrum for Conv2 is very close to a normal distribution. Interestingly, the spectra of Conv1 became more and more even and smooth when the error was decreased, which indicates that the involvement of diverse operations contributed to the performance improvement [34]. The popular view is that the functions of deeper convolutional layers are to extract more abstract features. According to our data, the discrepancy in the spectra may suggest the redundancy and functionality of the network as well.

B. *Network Complexity and Loss Function*

As we know, for a given training dataset, the landscape of the loss function is controlled by the adopted network structure. Usually the non-convexity of the loss function determines the trainability, in other words, the easiness in finding a good minimum with a low generalization error [38]. *It is of significance to investigate how the network structure affects the landscape of the loss function*. However, due to the dimensionality curse, it is prohibitive to study the loss function directly by looping through all variable values. Hence, current studies were either theoretical [33-37] or visual [38-41]. For instance, by aggressively assuming neurons were linear, Hardt & Ma [41] showed that local minimum was the global minimum by the principle of identity parameterization. Visualization methods such as 1D linear interpolation [39-40] and 2D contour plot [41] were widely used to study the “sharpness” of minima and the trajectories of optimization methods.

For the second-order network amendable for fuzzy logic interpretation, we can now characterize a minimum by its spectrum S, such as we did in Fig. 4, where we plotted the spectra of a convolutional layer for three minima respectively. We are especially interested in the spectra of “good” minima, meaning that minima can achieve decent performance on the training dataset. The inferior minima are not desirable. As we know, the important questions for the landscape of the loss function are how many good minima in total and how sharp each of these minima is. As we argued, the spectrum S of the whole network works somewhat like the “ID” for the minimum of the network, thereby allowing us to differentiate different minima on a high level. Since the spectrum contains critical information on the minimum, how many good minima S are there of the loss function is directly related to how many types of spectra are favorable. Furthermore, with random initialization, acquiring a sharp minimum requires to specify the initial value with high precision. In contrast, a flat minimum merely requires a rough initial value. Thus, the sharpness of a minimum is closely correlated with the probability of getting this minimum. Said differently, the sharper the minimum is, the harder to find this minimum. Consequently, the sharpness of a minimum determines the frequency of getting the spectrum of that minimum. Aside from discussing the minima of the loss function, we can also evaluate the complexity of the loss function in terms of good minima through spectral analysis. In the following, we report numerical experiments to quantitatively discuss how the network structure will change the loss function in terms of reaching good minima, and justify the feasibility of our spectral analysis method in network performance evaluation. Finally, we discuss the complexity of the loss function.

- *Experiments*

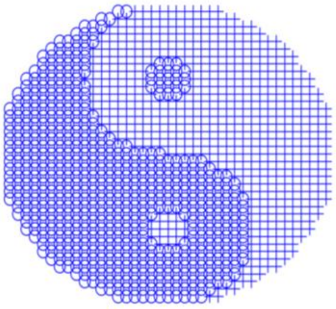


Fig. 5. Tai Ji benchmark used in this study, which is not used in the machine learning community before.

The experiment was designed as follows. Three different network configurations 2-6-6-1, 2-8-6-1 and 2-10-6-1 were selected to represent gradually widened network. The benchmark dataset was generated in reference to the most popular pattern Tai Ji, which represents, to many people, the essence of Chinese culture. The training dataset was made by gridding the two vectors -1:0.05:1 and -1:0.05:1 and then labeling the points with "+" and "o" according to the Tai Ji pattern. Finally, the 1,245 instances were synthesized as the training dataset, as shown in Fig. 5. To our best knowledge, the Tai Ji benchmark was not used before. For a specific network, we conducted training using the standard backpropagation algorithm after randomly initializing weights and biases with the MATLAB "*rands*" function. The learning rate was set to 0.004. The number of iterations was set to 1,000.

When the optimization process was trapped at minima, we calculated the performance over the dataset. We recorded the spectral types for the first hidden layer for good minima (>96.39% accuracy) and dropped bad points. In the fuzzy operation view, the neurons with two input variables in the first layer must belong to one of three quadratic operation types, denoted by $(+, +)$, $(+, -)$, $(-, -)$. With respect to the first hidden layer neurons in the 2-6-6-1 network configuration, if one minimum has 2 neurons perform the $(+, +)$ operations, 3 neurons conduct the $(+, -)$ operation, and 1 neuron does the $(-, -)$ operation , then the spectral type of this minimum is compactly denoted as (2,3,1). The spectral type in the second hidden layer is not recorded because herein the neurons in the three structures have different numbers of inputs, which cannot be compared directly in this feasibility study (certainly, they can be compared after certain normalized in a follow-up study). We repeated this process until enough good minimum points were recorded. It can be proven that given the number of neurons n there are potentially C_{n+2}^{2} types of spectra. Therefore, for the configurations 2-6-6-1, 2-8-6-1 and 2-10-6-1 with 6, 8, 10 neurons respectively in the first hidden layer, there are maximumly 28, 45, 66 types of spectra that can potentially show up for good minima. Hence, we respectively recorded 56, 90, 132 good minima for structures 2-6-6-1, 2-8-6-1 and 2-10-6-1 to keep the ratio of good minima identical for the configurations. The results of the spectra and corresponding frequencies are shown in Fig. 6.

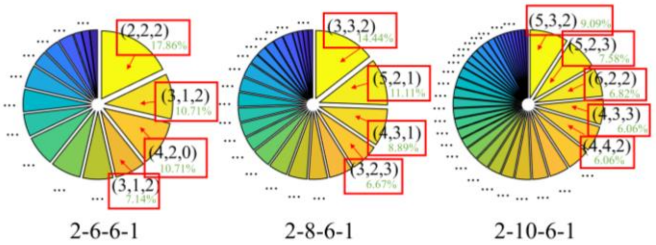


Fig. 6. Spectral analysis on good minima. The number of spectral types is correlated to the number of good minima. The frequency of a spectrum suggests the flatness of a good minimum.

Our experiment retrieved 16, 24, 37 types of "good" spectra for the 2-6-6-1, 2-8-6-1 and 2-10-6-1 configurations respectively. The increased types of good spectra from 2-6-6-1 to 2-10-6-1 means the increased numbers of good minima of the corresponding loss functions. This result is not surprising because the newly added neurons, coupled with old neurons, in the first layer should create new minima. These new minima are from effective combinations of fuzzy logic modules.

Furthermore, we can visualize the frequency of spectral components in green fonts in Fig. 6. For clarity, we only marked some main spectral components and their corresponding frequencies. The frequency of a spectral component does matter because it reflects the sharpness of a good minima. In the loss function of the network 2-6-6-1, the minima with spectrum type (2,2,2) are dominating with a frequency 17.86%, while the dominant minima in the loss functions of the networks 2-8-6-1 and 2-10-6-1 account for 14.44% and 9.09% respectively, which means the optimizer are easier to converge to the dominant minima in 2-6-6-1 compared to that in 2-8-6-1 and 2-10-6-1 configurations. Additionally, we calculated the variance of frequency of spectral components for the three configurations, which are 0.0017, 0.0011, 0.0004 for 2-6-6-1, 2-8-6-1 and 2-10-6-1 respectively. These data indicate that the sharpness measures of different minima of the loss function of 2-10-6-1 are closer with each other than that of 2-8-6-1 and 2-10-6-1.

To be more comprehensive, we define the entropy of good minima (EGM) as $-\sum_i p_i \lg_2(p_i)$, which describes the complexity of the loss function specific to a network, where p_i is the probability of the i^{th} good minimum. Generally speaking, the more minima the loss function contains, the less organized the minima, and the higher the complexity of the loss function. Hence, this entropy metric agrees with the common understanding of the complexity of the loss function. In reality, we do not have a method to characterize good minima individually, but we can approximately calculate EGM by replacing spectral components with minima and then probabilities of minima with frequencies of spectral components. It turns out that the entropy of the loss functions of 2-6-6-1, 2-8-6-1 and 2-10-6-1 are 3.7269, 4.2366, 4.8233 respectively. As expected, the added neurons indeed increased the complexity of the loss function. Generally, structural complexity means functional complexity, being consistent to the principle in biology that structure determines functionality.

- *Theoretical Analysis on EGM*

Indeed, EGM will ascend as the network goes wider. Without loss of generality, it is assumed that the networks are multilayer perceptrons (MLPs), and our conclusion also holds for convolutional models after some adaptions. By wider networks we mean that the number of neurons in every hidden layer is at least the same as that of the reference network; for example, the structure 2-5-3-2 is not wider than structure 2-3-4-2.

Proposition: Entropy of Good Minima (EGM) of the MLP will not decrease if this MLP is augmented with new neurons.

Proof: By definition, the loss function of the parent network has at least one good minimum; otherwise this network is not well structured to compute EGM. Without loss of generality, it is further assumed that the loss function of a network ϕ has k good minima. If the chance of finding the i^{th} minima is p_i^{ϕ}, the corresponding EGM is $\sum_i^k p_i^{\phi}\,\mathrm{lg}_2(p_i^{\phi})$. Next, we add new neurons into ϕ, thereby building a wider network Φ.

Then, we target good minima of the network Φ formed by setting parameters inside the newly added neurons to zeros. At the same time, considering that the added neurons and old neurons are of the same structure, we can exchange new neurons with the same number of old neurons in the same layer, and set their parameters to zero. Clearly, the total number of good minima will be enlarged to Ck, where C denotes a certain combinatorial constant. Due to the equivalence, the probabilities of finding these Ck minima end up being divided by C, as $\frac{p_1^{\phi}}{\mathrm{C}},\ldots,\frac{p_1^{\phi}}{\mathrm{C}},\ldots,\frac{p_k^{\phi}}{\mathrm{C}},\ldots,\frac{p_k^{\phi}}{\mathrm{C}}$ respectively. Then, the EGM of Φ is computed as follows:

$$-\sum_i^k C\left(\frac{p_i^{\phi}}{c}\right)\log\left(\frac{p_i^{\phi}}{c}\right)=-\sum_i^k p_i^{\phi}\log\left(\frac{p_i^{\phi}}{c}\right)$$
$$>-\sum_i^k p_i^{\phi}\log\left(p_i^{\phi}\right) \qquad (3)$$

However, the added neurons can work nontrivially, which means that the added neurons together with old neurons can create new good minima in addition to the just-mentioned Ck good minima. Let us say that there are l new good minima so created and the corresponding probabilities of finding them are $p_1^{\phi},\ldots,p_l^{\phi}$ respectively. Then, the new minima will take away the likelihood of the previous minima by a factor $\alpha, \alpha = 1-\sum_i^l p_i^{\phi}$, and the EGM of the new network is

$$-\sum_i^l p_i^{\phi}\,\mathrm{lg}_2(p_i^{\phi})-\sum_j^k C(\alpha p_j^{\phi}/C)\,\mathrm{lg}_2(\alpha p_j^{\phi}/C) \qquad (4)$$

Due to the monotonically decreasing property of $-xlg_2x$ over (0,1), the right term of Eq. (4) satisfies: $-\sum_j^k\left(\frac{\alpha p_j^{\phi}}{c}\right)\mathrm{lg}_2\left(\frac{\alpha p_j^{\phi}}{c}\right)>-\sum_j^k\left(\frac{p_j^{\phi}}{c}\right)\mathrm{lg}_2\left(\frac{p_j^{\phi}}{c}\right)$. Since $-\sum_i^l p_i^{\phi}\,\mathrm{lg}_2\left(p_i^{\phi}\right)>0$, we have

$$-\sum_i^l p_i^{\phi}\,\mathrm{lg}_2(p_i^{\phi})-\sum_j^k C(\frac{\alpha p_j^{\phi}}{C})\,\mathrm{lg}_2(\frac{\alpha p_j^{\phi}}{C})$$
$$>-\sum_j^k p_j^{\phi}\,\mathrm{lg}_2\left(\frac{p_j^{\phi}}{c}\right)>-\sum_j^k p_j^{\phi}\,\mathrm{lg}_2\left(p_j^{\phi}\right) \qquad (5)$$

By Eqs. (3) and (5), we finish our proof.

EGM describes the complexity of the loss function of a network. As the above proof shows, if the network goes wider, the complexity of the loss function will increase. It seems that structural enhancement will result in the functional enhancement. Can this trend hold when network goes deeper? The answer is yes, and a very similar proof can be given, which is not included here for conciseness.

In reality, how to find every individual good minimum is still far from being clear, a practical training strategy is needed. For second-order networks, we can use spectral analysis to assess good minima, thereby labeling/ranking the good minima into several classes. Hence, spectral signature seems a helpful hint on good or bad minima. In our aforementioned experiment, the entropy values for the 2-6-6-1, 2-8-6-1 and 2-10-6-1 configurations are 3.7269, 4.2366, and 4.8233 respectively. This result agrees with the non-decreasing trend of EGM, predicted by our theoretical proof.

C. Flat Minima and Network Evaluation

Now, we extend precious results to the generalization ability of good minima and further discuss whether flat minima would generalize better and how to design the network structure.

- *Generalization Ability of Flat Minima*

As we mentioned before, we need to initialize network parameters with high precision in order to sharp minima instead of being trapped elsewhere. In contrast, flat minima require loose specification of parameters. In the terminology of information theory, fewer information is needed or we have better chance to pick up flat minima [42]. In general, it is claimed that less description length means low network complexity and high generalization performance [42], as indicated by Occam's razor principle. Hence, most people believe that flat minima can generalize better than sharp minima. Hinton and Van Camp [43] justified the generalization ability of flat minima by the Bayes argument via KL-divergence. However, there are papers arguing that sharp minima could generalize better for deep models [44]. Here, with spectral analysis, we can compare the generalization performance of flat and sharp minima in the case of second-order networks.

In the forerunner experiments, we recorded 56, 90, 132 good minima and corresponding spectra respectively for the three configurations. Here we first constructed 7,825 instances as a test dataset by gridding two finer vector -1:0.02:1, then we evaluated the generalization performance of these good minima. Taking the 2-6-6-1 network as an example, green line segments connecting green triangles in Fig. 7 show the mean generalization performance of every spectrum of the 2-6-6-1 structure, while bar shows the frequency number of obtaining the corresponding spectrum.

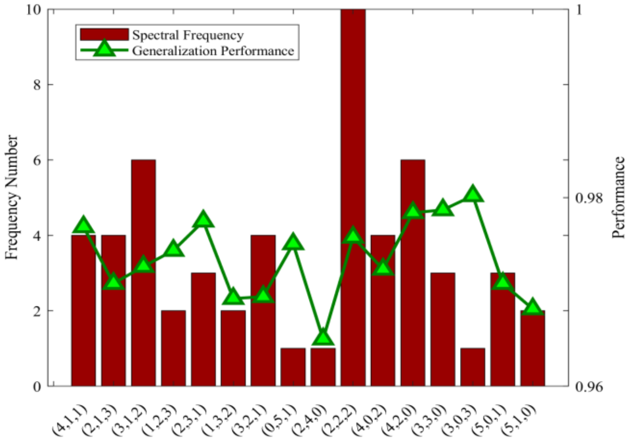


Fig. 7. In the 2-6-6-1 network, the sharp minima generalize better over the Tai Ji dataset.

As we stated, the sharpness of a minimum is indirectly reflected by the frequency of getting this minimum. Therefore, minima of spectrum (3,0,3) is one of the sharpest minima. However, it has the best mean generalization ability admitting the performance of 98.02%. This is a counterexample demonstrating that sharp minimum can generalize well. On the other hand, aside from showing a solitary counterexample, we also need to conduct statistical comparison, since the argument that flat minima generalize better is the general opinion. Hence, we compared the mean generalization performance of spectra with top 7 highest frequencies to that of remained spectra. Because top 7 spectra accounts for 67.86% of minima, average flatness of these 7 spectra should be larger than that of the remained spectra. In this way, a statistical sense is made. It turns out that the generalization performance of top 7 spectra is 97.17%, while the generalization performance of the remained spectra is 97.48% for the 2-6-6-1 network. Similar calculations for the spectra of the 2-8-6-1 and 2-10-6-1 networks were done as well. In the 2-8-6-1 configuration, 24 different types of spectra showed up, the mean generalization performance of top 5 spectra (accounting for 41% frequency in total) is 97.49%, which is slightly lower than 97.50% of the remained spectra. Consistently, in the 2-10-6-1 configuration, 37 spectra were identified, and the performance of top 5 spectra (accounting for 32.58% frequency in total) is 97.58%, lower than 97.70% of remained 32 spectra.

Our data demonstrate that sharp minima may generalize better than flat minima on average. Since there are other experiments showing that flat minima generalize better, we think that there may not be necessarily a simple relationship between the sharpness of minima and the generalized ability.

- *Network Evaluation*

Recently, more and more researchers are exploring guidelines for design of network architectures [45-53]. In [47-48], the target network was evaluated according to their performance on a given cross-validation dataset using a reinforcement learning technique. In [45-46], sparsity regularization was employed to generate a compact network that admits performance better than that of a large network. In [54], an attempt was made to widen the network instead of making it deeper. How to assess the structure of a given network and optimize its structure and/or parametrization is still an open question, mainly due to the black-box nature of neural networks. Here, we propose a novel measure to evaluate the network based on spectral analysis.

Given a training dataset, evaluating the network structure is equivalent to evaluating its loss function. For deep learning engineers, they not only like network candidates that have minima with a strong generalization ability but also prefer those candidates from which they can obtain desirable minima more easily in the training process. Said differently, engineers hope that the minimum will be obtained easily and yet possess good generalization ability so that they can finish tuning parameters with least time to meet the performance requirement. Based on this consideration, we define a new measure as: $M = \sum_i p_i K_i$, where p_i is the probability of the i^{th} good minimum and K_i is corresponding generalization performance. The network with high M value will enjoy a good balance between trainability and performance. To calculate M, we can again replace probability with frequency of spectral components as an approximation.

Next, we use M to evaluate the three network structures over Tai Ji data in terms of spectral frequency and generalization performance. The score of the three structures 2-6-6-1, 2-8-6-1 and 2-10-6-1 are 0.9741, 0.9741 and 0.9762 respectively. Since 2-10-6-1 achieved the highest score, we believe that the network structure 2-10-6-1 is the best among these structures. As for 2-6-6-1 and 2-8-6-1, their scores are identical, we believe the difference between them is slim in trainability.

D. *Entropy View of Training*

The network training usually starts with randomly initialized parameters. After hundreds and thousands of iterations, the gradient search algorithm will often converge to a certain point. With spectral analysis, we have a top-level understanding of the training process. Suppose that we randomly generate lots of initial points and search for a minimum from them in parallel and at the same pace. Then, we apply our spectral analysis to classify these points into multiple classes in every step. Resembling to what we did before, we can calculate the entropy $-\sum_i p_i \lg_2 p_i$ in every step, according to the frequencies in these classes respectively. The entropy measures how organized these points are distributed, as commonly done in statistical physics.

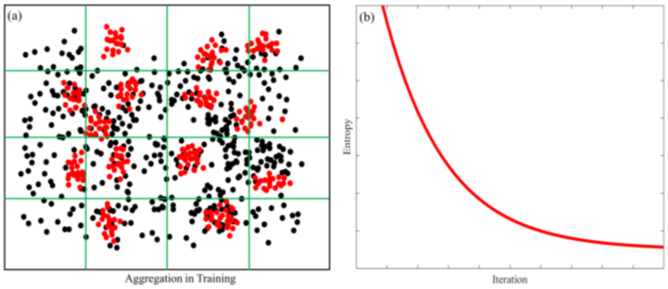


Fig. 8. Training process making the randomly generated points organized, with the entropy constantly reduced over iteration.

As shown in Fig. 8(a), each black spot represents a group of parameters generated in random initialization. The grid formed by green lines denotes the different types found in our spectral analysis. The size of each class is identical implying that the randomly initialized points will fall into any of these classes with the same probability. By training, these initial points tend to move towards to the nearest minima driven by gradient searching. After a sufficient number of iterations, these points will aggregate into a number of clusters, shown as the red points. In the viewpoint of entropy, random points are initially quite disordered and equally distributed in each class leading to the maximum entropy. The training process is to drive these disordered points to different minima such that more instances in the dataset are classified correctly. Once the points are clustered to reach the equilibrium, the training process converges. Correspondingly, the entropy is continuously descending until its plateau, as shown in Fig. 8(b). It is safe to say that the training process empowers a neural network with structures, regularities, and information. If the loss function is convex, all the points will aggregate in a single cluster for the global minimum, then the corresponding entropy is zero. In contrast, if the loss function is highly non-convex, the final entropy could be still very large. Hence, the entropy loss in the training process is related to the convexity of the loss function of the network.

IV. Discussions and Conclusion

With the concept of "generalized fuzzy operations", the quadratic neurons are categorized into various eigen types, which can be used to implement various fuzzy operation gates. Categorization is the first step to figure out the role of second-order neurons in a neural network, thereby allowing a fuzzy logic understanding of the functionalities of neurons, layers, and networks. As a positive outcome, our spectral analysis method has given valuable hints on the functionalities of the convolutional layers, minima of the loss function as the network goes wider, network design, and the training process. More efforts are needed to unveil the deep mechanisms that govern the neural network of second-order neurons, such as the effect when a network goes deep, wide, both ways, or connected with in graphs and with memories.

Eigenvalue characterization is a feasible path to describe the functions of neurons and networks, which is well fitted to second-order networks. In our fuzzy interpretation, each spectral type represents a generic family of functional modules embedded in second-order neurons. As we asserted before, our eigenvalue characterization is based on the intrinsic semi-topological feature of second-order neurons. However, the decision boundary formed by a neuron must be addressed when we move our spectral analysis from top to bottom levels [55]. This is the reason why our spectral analysis is only a global approximation to the true picture. For example, with the use of one second-order neuron to classify an XOR-pattern in Fig. 9, all three different decision boundary curves implementable with a second-order neuron, denoted in three colors respectively, are qualified to finish this task.

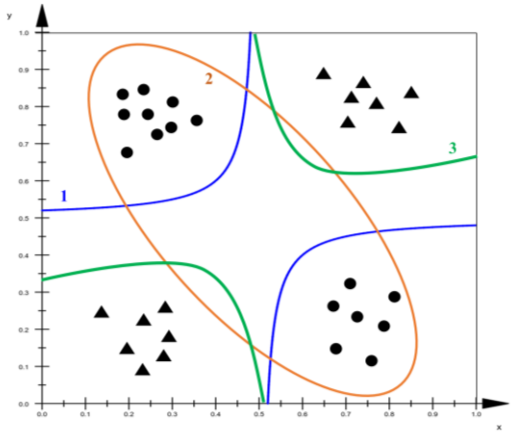


Fig. 9. Eigenvalue characterization fail to differentiate decision boundary curves 1-3.

Our eigenvalue characterization method can tell curve 2 from curves 1 and 3 but fail to differentiate curves 1 and 3, although curves 1 and 3 are two different solutions for XOR-like task. Despite this limitation of spectral analysis, we can still get reliable results in a statistical sense. In contrast, the networks with conventional neurons do not offer such a unique yet convenient angle to characterize the network.

Through interconnected fuzzy gates, complicated system tasks are accomplished by composite functions whose building blocks are fuzzy operations or second-order neurons. Interpreting the inner working of a neural network in terms of basic fuzzy logic operations is a typical case of reductionism. It is feasible to understand how a complex system works by figuring out the functions of components of that system and their interconnections. In the history of engineering, modular methods are often effective, such as for software development [56] and integrated circuits design [57]. By constructing neural networks in this modular way, there are opportunities to optimize the network architecture design and improve the end-to-end performance.

Recently, Ali Rahimi heuristically compared the aberrations correction in optical system design with the optimization of neural networks in his Blog [58]. In the optical design, engineers have a strong sense what optical elements do, which guide them to tune the optical network efficiently. However, for deep learning engineers, the neural network is a black-box that is difficult to decipher. A possible solution is to identify the function module correctly in successful networks. In light of these networks, valuable modules can be identified, and made with second-order neurons as fuzzy logic modules for advanced networks and new applications.

In Boolean algebra, logic expressions can be possibly simplified based on the truth table. The simplified expression is logically equivalent to the expression before simplification. Our hypothesis is that the generalized fuzzy logic expressions are reducible as well. Then, a complicated neural network consisting of neurons representing generalized fuzzy operations can be made equivalent to more compact neural networks. As the next step, we plan to find the equivalent networks of a well-trained network so that our fuzzy logic approach offers not only interpretability but also guidelines for deep learning practice.

In conclusion, with specific examples we have demonstrated that a fuzzy logic interpretation can be given to any neural network. Aided by spectral analysis and entropy measure, unique insights can be obtained in the context of machine learning with neural networks. Our approach is complementary to other efforts offering interpretability of neural networks, and whose potential is yet to be further explored.

REFERENCES

[1] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain Tumor Segmentation using Convolutional Neural Networks in MRI Images," IEEE Trans Med Imaging, Mar 04, 2016.

[2] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, "Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network," IEEE Trans Med Imaging, vol. 35, no. 5, pp. 1207-1216, May, 2016.

[3] C. You, et al., "Structure-sensitive Multi-scale Deep Neural Network for Low-Dose CT Denoising", arXiv preprint arXiv:1805.00587, 2018.

[4] G. Wang, M. Kalra, & C. G. Orton, "Machine learning will transform radiology significantly within the next 5 years," Medical physics, vol. 44, no. 6, pp. 2041-2044, 2017.

[5] L. Chu, X. Hu, J. Hu, L. Wang, & J. Pei, "Exact and Consistent Interpretation for Piecewise Linear Neural Networks: A Closed Form Solution, " in KDD, 2018.

[6] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," In CVPR, 2015.

[7] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and Andrew Y Ng, "Multimodal deep learning," In ICML, 2011.

[8] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," arXiv:1506.06579, 2015.

[9] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," arXiv:1506.06579, 2015.

[10] B. Zhou, D. Bau, A. Oliva, and A. Torralba, "Interpreting Deep Visual Representations via Network Dissection," arXiv:1711.05611, 2017

[11] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," In CVPR, 2016.

[12] M. Wu, M. C. Hughes, S. Parbhoo, et al. "Beyond Sparsity: Tree Regularization of Deep Models for Interpretability," arXiv preprint arXiv:1711.06178, 2017.

[13] L. Fan, "Revisit Fuzzy Neural Network: Demystifying Batch Normalization and ReLU with Generalized Hamming Network," in NIPS, 2017.

[14] J. Ba and R. Caruana, "Do deep nets really need to be deep?" In NIPS, 2014.

[15] O. Bastani, C. Kim, and H. Bastani, "Interpreting Blackbox Models via Model Extraction," arXiv:1705.08504, 2017.

[16] Z. Che, S. Purushotham, R. Khemani, and Y. Liu, "Distilling knowledge from deep networks with applications to healthcare domain," arXiv:1512.03542, 2015.

[17] P. Adler, et al., "Auditing black-box models for indirect influence," Knowledge and Information Systems, vol. 54, no. 1, pp. 95-122, 2018

[18] R. S. Ramprasaath, et al., "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," in CVPR, 2016.

[19] S. Lundberg, & S. I. Lee, "An unexpected unity among methods for interpreting model predictions," in NIPS, 2017.

[20] P. W. Koh, P. Liang, "Understanding black-box predictions via influence functions," in ICML, 2017.

[21] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," In CVPR, 2016.

[22] N. Lei, K. Su, L. Cui, S. T. Yau, & D. X. Gu, "A Geometric View of Optimal Transportation and Generative Model," arXiv preprint arXiv:1710.05488, 2017.

[23] Y. Lu, A. Zhong, Q. Li & B. Dong, "Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations," arXiv preprint arXiv:1710.10121, 2017.

[24] G. Wang, "A perspective on deep imaging," IEEE Access, vol. 4, pp. 8914-8924, 2016.

[25] J. C. Ye, Y. Han, & E. Cha, "Deep convolutional framelets: A general deep learning framework for inverse problems," SIAM Journal on Imaging Sciences, vol. 11, no. 2, pp. 991-1048, 2018.

[26] P. Mehta, D. J. Schwab, "An exact mapping between the variational renormalization group and deep learning," arXiv preprint arXiv:1410.3831, 2014.

[27] F. Fan, W. Cong, G. Wang, "A new type of neurons for machine learning," International Journal for Numerical Methods in Biomedical Engineering, vol. 34, no. 2, Feb. 2018.

[28] F. Fan, W. Cong & G. Wang, "Generalized Backpropagation Algorithm for Training Second-order Neural Networks," International Journal for Numerical Methods in Biomedical Engineering, doi.org/10.1002/cnm.2956, 2017.

[29] R. Livni, et el., "On the computational efficiency of training neural networks." In NIPS, 2014.

[30] T. J. Ross, Fuzzy Logic with Engineering Applications. New York: McGraw-Hill, 1995.

[31] A. J. Hoffman & H. W. Wielandt, "The variation of the spectrum of a normal matrix," In Selected Papers Of Alan J Hoffman: With Commentary, pp. 118-120.

[32] Y. LeCun, et al. "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[33] Y. N. Dauphin, et al., "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in NIPS, 2014.

[34] M. Hardt and T. Ma, "Identity matters in deep learning," in ICLR, 2017.

[35] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," in ICML, 2017.

[36] C. Yun, S. Sra, and A. Jadbabaie, "Global optimality conditions for deep neural networks," arXiv preprint arXiv:1707.02444, 2017.

[37] B. Xie, Y. Liang, and L. Song, "Diverse neural network learns true target functions," in AISTATS, 2017

[38] H. Li, et al., "Visualizing the Loss Landscape of Neural Nets," arXiv preprint arXiv:1712.09913, 2017.

[39] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in ICLR, 2017.

[40] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, "Sharp minima can generalize for deep nets," in ICLR, 2017.

[41] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, "Qualitatively characterizing neural network optimization problems," in ICLR, 2015.

[42] S. Hochreiter, & J. Schmidhuber, "Simplifying neural nets by discovering flat minima. in NIPS, 1995.

[43] G. E. Hinton, and D. V. Camp, "Keeping the neural networks simple by minimizing the description length of the weights," In Proceedings of the sixth annual conference on Computational learning theory, pp. 5–13. ACM, 1993.

[44] L. Dinh, R. Pascanu, S. Bengio, & Y. Bengio, "Sharp minima can generalize for deep nets," arXiv preprint arXiv:1703.04933, 2017.

[45] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in ECCV, 2016.

[46] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in NIPS, 2016.

[47] B. Zoph, & Q. V. Le, "Neural architecture search with reinforcement learning," in ICLR, 2017.

[48] B. Baker, et al., "Designing neural network architectures using reinforcement learning," in ICLR, 2017

[49] K. Li and J. Malik, "Learning to optimize," arXiv preprint arXiv:1606.01885, 2016.

[50] D. Wierstra, F. J Gomez, and J. Schmidhuber, "Modeling systems with internal state using evolino," in GECCO, 2005.

[51] M. Andrychowicz, et al, "Learning to learn by gradient descent by gradient descent," in NIPS, 2016.

[52] T. Y. Kwok, & D. Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," IEEE transactions on neural networks, vol. 8, no. 3, pp. 630-645, 1997.

[53] M. G. Bello, "Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks," IEEE Transactions on Neural networks, vol. 3, no. 6, pp. 864–875, 1992.

[54] G. Pandey & A. Dukkipati, "To go deep or wide in learning?" arXiv preprint arXiv:1402.5634, 2014.

[55] N. Lei, Z. Luo, S. T. Yau & D. X. Gu, "Geometric Understanding of Deep Learning," arXiv preprint arXiv:1805.10451, 2018

[56] B. C. Pierce, Types and Programming Languages. Cambridge: MIT press, 2002.

[57] J. M. Rabaey, A. P. Chandrakasan, B. Nikolic, Digital Integrated Circuits. Englewood Cliffs: Prentice Hall, 2002.

[58] A. Rahimi, "Lessons from Optics, The Other Deep Learning", Retrieved from http://www.argmin.net/2018/01/25/optics/