Spurious Local Minima Are Common for Deep Neural Networks With Piecewise Linear Activations

Bo Liu

Abstract—In this article, theoretically, it is shown that spurious local minima are common for deep fully connected networks and average-pooling convolutional neural networks (CNNs) with piecewise linear activations and datasets that cannot be fit by linear models. Motivating examples are given to explain why spurious local minima exist: each output neuron of deep fully connected networks and CNNs with piecewise linear activations produces a continuous piecewise linear (CPWL) function, and different pieces of the CPWL output can optimally fit disjoint groups of data samples when minimizing the empirical risk. Fitting data samples with different CPWL functions usually results in different levels of empirical risk, leading to the prevalence of spurious local minima. The results are proved in general settings with arbitrary continuous loss functions and general piecewise linear activations. The main proof technique is to represent a CPWL function as maximization over minimization of linear pieces. Deep networks with piecewise linear activations are then constructed to produce these linear pieces and implement the maximization over minimization operation.

Index Terms—Convolutional neural networks (CNNs), deep learning theory, deep neural networks, local minima, loss landscape.

I. INTRODUCTION

A LTHOUGH deep neural networks, such as deep fully connected feedforward networks and especially convolutional neural networks (CNNs) (e.g., [1], [2]), have been successfully applied to many disciplines, including computer vision (e.g., [1], [2], [3], [4], [5], [6]) and natural language processing, the theoretical understanding of deep neural networks is still limited. The long-term goal of deep learning theory [7] is to demystify the black box of deep neural networks, which will make sure that they can be applied reliably to those security-critical areas such as autonomous vehicles and medical diagnosis, and inspire the development of future deep learning algorithms.

In the field of deep learning theory, the loss landscape of deep neural networks [8], especially whether suboptimal (spurious) local minima are common, plays an important role and has a great impact on the training of deep neural networks. For example, gradient-based local search methods (such as gradient descent) can easily get stuck in spurious local minima. The nonlinearity in activation functions makes the loss landscape extremely nonconvex and complex. For one-hidden-layer networks with the ReLU activation, the fact that spurious local minima exist has been shown in [9], [10], [11], [12], and [13]. In these works, either specific data examples or networks were constructed or experiments were conducted to demonstrate the existence of spurious local minima. Recently, He et al. [14], Ding et al. [15], Goldblum et al. [16], and Liu et al. [17] exhibited existence of spurious local minima for deep fully connected ReLU networks. They constructed deep ReLU networks that locally emulate linear neural networks to fit all data samples and thus inherit the local minima of linear networks. All ReLU neurons are active at the constructed spurious local minima such that networks behave linearly. Therefore, the constructed spurious local minima are located in unimportant regions in parameter space and rarely encountered in practice when optimizing with appropriate initializations. Despite these works, the following fundamental theoretical questions remain unanswered or require a deeper understanding. Why do spurious local minima exist? Are they common and widespread in parameter space (not restricted to those specific locations in existing constructions)? The goal of this work is to understand why spurious local minima exist and construct spurious local minima that are common for deep fully connected networks and CNNs with piecewise linear activations.

In this article, starting with observations drawn from motivating examples, the reason why spurious local minima are common in deep fully connected networks and average-pooling CNNs with piecewise linear activations are explained. The key insight is that each output neuron of deep neural networks with piecewise linear activations produces a continuous piecewise linear (CPWL) output. When minimizing the empirical risk, each linear piece of the CPWL function can optimally fit a disjoint group of data samples, and different fitting patterns usually result in different levels of empirical risk, causing the prevalence of spurious local minima. Explicit constructions of deep fully connected networks and average-pooling CNNs are then presented to show the existence of spurious local minima in general settings with arbitrary continuous loss functions, general piecewise linear activations, and any datasets that cannot be fit by linear models. The main idea is to represent a locally optimal CPWL predictor as a maximization over minimization of its constituent linear pieces and construct deep networks with piecewise linear activations to produce these linear pieces and implement the maximization and minimization operations.
To the author’s best knowledge, this is the first theoretical work showing the prevalence of spurious local minima in deep fully connected networks and average-pooling CNNs with piecewise linear activations, which deepens our understanding of loss landscape of deep neural networks.

This article is organized as follows. Section II is related work. Section III describes the deep fully connected networks and convolutional neural networks (CNNs) and gives some mathematical preliminaries on spurious local minima. In Section IV, motivating examples are provided to illustrate the existence of spurious local minima and explain why they are common. Section V presents the main results and their proof sketches. In Section VI, the detailed proofs of theorems are given. Finally, conclusions and future directions are provided.

II. RELATED WORK

It has been shown that there are no spurious local minima for some networks and learning models, including deep linear networks [18], [19], [20], [21], [22], [23], [24], matrix completion and tensor decomposition (e.g., [25]), one-hidden-layer networks with quadratic activations [26], [27], deep linear residual networks [28], and deep quadratic networks [29].

For one-hidden-layer ReLU networks, by constructing specific examples of networks and data samples, the existence of spurious local minima has been demonstrated in [9], [10], [11], [12], [13], and [15]. Instead of specific constructions, some works discussed local minima for one-hidden-layer ReLU networks by analyzing the conditions for their existence. Soudry and Carmon [30] gave the conditions under which a differentiable local minimum has a zero loss. Laurent and von Brecht [31] showed that ReLU networks with hinge loss can only have nondifferentiable local minima and gave the conditions for their existence for linearly separable data. For overparameterized one-hidden-layer ReLU networks, Safran and Shamir [32] showed that there is a high probability of initializing in a low-loss basin. Soudry and Hoffer [33] exhibited that, given standard Gaussian data and Gaussian weight initialization, the volumes of differentiable regions containing spurious local minima are exponentially vanishing. He et al. [14] and Liu [34] proved that local minima are global in each differentiable region, and Liu [34] gave the necessary and sufficient conditions for the existence of spurious local minima in one-hidden-layer ReLU networks with mean square error (mse) loss. Sharifnassab et al. [13] constructed a complex data pattern to produce cup-shaped nondifferentiable local minima for one-hidden-layer ReLU networks.

For deep neural networks, He et al. [14], Ding et al. [15], Goldblum et al. [16], and Liu et al. [17] showed that spurious local minima exist for fully connected deep neural networks with arbitrary piecewise linear activations and a general class of differentiable loss functions. These works constructed examples of spurious local minima in which all ReLU neurons are active such that the deep neural networks are reduced to linear predictors, and therefore, these constructions are relatively trivial.

Besides exploring the existence of spurious local minima in the empirical risk of neural networks, the loss landscape of neural networks has been studied from other perspectives. The nonexistence of spurious valley (bad basin) for ultrawide networks was explored in [15], [35], [36], [37], and [38]. Some works tried to reshape the loss landscape by modifying the network architecture or using regularization. Liang et al. [39], [40] and Kawaguchi and Kaebbling [41] showed that by adding a single-layer network or even a single special neuron in the shortcut connection, every local minimum becomes global. Ge et al. [42], Gao et al. [43], and Feizi et al. [44] designed new losses or special networks to make local minima global. Chen et al. [45] deformed loss surfaces to enhance the gradient descent optimizer’s ability to filter out sharp minima. Shamir [46] and Kawaguchi and Bengio [47] proved that losses of all local minima in ResNets are no worse than those of global minima in the corresponding shallow linear predictors. Milne [48] showed that the loss surface of a feedforward ReLU network regularized with weight decay is strongly convex in a special region. Choromanska et al. [49] used spin glass models in statistical physics to analyze the loss landscape, which greatly ignores the nonlinear nature of deep neural networks. Pennington and Worah [50] and Pennington and Bahri [51] used a random matrix theory to study the spectrum of Hessian matrix of loss. Mei et al. [52] and Zhou and Feng [53] studied the landscape of expected loss. Bianchini and Scarselli [54] showed the topological expressiveness of deep networks.

Some works studied loss landscape of deep neural networks empirically. Dauphin et al. [55] performed various experiments to show that training can get stuck in plateaus and advocated that saddle points are the main source of difficulty for optimizing deep networks. Freeman and Bruna [56] discovered that the loss landscape becomes bumpy for low empirical risks, indicating that spurious local minima are common in low-loss regions. Goodfellow et al. [57], Liao and Poggio [58], Li et al. [59], and Fort and Jastrzkebski [60] explored the visualization of loss landscape. By visualizing random slices of loss surface, Li et al. [59] observed that loss surface of deep feedforward networks transitions from nearly convex to chaotic when increasing depth. Mehmeh-Gopol et al. [61] explained this observation by a spectral analysis. Draxler et al. [62], Garipov et al. [63], and Anokhin and Yarotsky [64] revealed the so-called mode connectivity phenomenon in which different global minima are connected by low-loss curves for overparameterized networks.

Understanding the landscape of empirical loss focuses on the geometry side of neural network optimization. Another line of research studies optimization of neural networks by exploring the convergence of gradient descent. For example, Soltanolkotabi et al. [26], Tian [65], Zhong et al. [66], Li and Yuan [67], Zhou et al. [68], and Zhang et al. [69] studied the convergence of gradient descent for one-hidden-layer ReLU networks. Some recent works (e.g., [70], [71], [72], [73]) showed that gradient descent converges for deep networks if they are sufficiently wide, the step size is small and the weights are initialized in a small region. Instead of local analysis for overparameterized networks, the work in this article is more general by considering existence of spurious local minima in whole parameter space of a regular ReLU network. In a
related recent line of work, it has been shown that training of overparameterized deep neural networks can be characterized by the training dynamics of kernel regression with neural tangent kernel (NTK) [74].

Neural networks with piecewise linear activations output CPWL functions. The number of linear regions in such piecewise linear outputs was discussed in [75], [76], [77], and [78] for deep fully connected neural networks and in [79] for CNNs.

The works most related to this article include [11], [14], [15], [16], and [17]. The result of [11] was similar to that of [14] and only one-hidden-layer networks were considered in [11]. He et al. [14], Ding et al. [15], Goldblum et al. [16], and Liu et al. [17] dealt with deep networks with piecewise linear activation functions. The main differences between this article and these works include the following.

1) He et al. [14], Ding et al. [15], Goldblum et al. [16], and Liu et al. [17] used a single linear predictor to fit all data samples, resulting in a single spurious local minimum. In comparison, the work in this article uses a CPWL predictor to fit disjoint groups of samples. Thus, the predictors in this article are more flexible and can have many distinct fitting patterns, leading to prevalence of spurious local minima.

2) In [11], [14], [15], [16], and [17], all ReLU neurons in the constructed networks are active (thus, the networks are reduced to linear predictors), which is unusual for practical ReLU networks and the constructed spurious local minima are rarely encountered in practical optimization. In comparison, some ReLU neurons in the constructions of this article can be inactive and the resulted spurious local minima are more common.

3) Different techniques are used to prove why the constructed local minima are spurious. In [11], [14], and [17], it was shown that there exist perturbation directions in which the empirical risk decreases. In [16], whether the constructed local minima are spurious was merely addressed by providing numerical evidence, and Ding et al. [15] resorted to the assumption of realizability. In comparison, in this work, the local minima with lower empirical risks are constructed explicitly.

4) This work also deals with CNNs, which are not considered in [11], [14], [15], [16], and [17].

III. NOTATIONS AND PRELIMINARIES

A. Notations

\( M_{i,} \) and \( M(_, i) \) denote the \( i \)th row of matrix \( M \) and \( M(_, j) \) denote the \( j \)th column of \( M \). \( M_{i,j} \) and \( M_{ij} \) represent the \( (i, j) \) entry of \( M \). \( v_i \) denotes the \( i \)th component of vector \( v \). \( [N] \) is the abbreviation of \{1, 2, ..., N\}.

B. Deep Fully Connected Networks

Deep fully connected feedforward networks are described in this section. Consider a training set comprised of samples \( \{ (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) \} \), where \( x_i \in \mathbb{R}^d_i \) and \( y_i \in \mathbb{R}^d_y \) are, respectively, the input and output of the \( i \)th sample. Let \( L \) be the number of layers in the network. Denote the weight matrix and bias vector of the \( j \)th layer as \( W^j \) and \( b^j \), respectively, with \( W^j \in \mathbb{R}^{d_j \times d_{j-1}} \) and \( b^j \in \mathbb{R}^{d_j} \), where \( d_j \) is the number of neurons (or width) in the \( j \)th layer. The output of the \( i \)th data sample at the \( l \)th layer, denoted as \( y^{l,i} \), is computed by

\[
y^{l,i} = \sigma(W^l y^{l-1,i} + b^l), \quad (l = 1, 2, \ldots, L - 1).
\]

We denote that \( y^{0,i} := x_i \). \( \sigma \) is the activation function, and \( \sigma(x) = \max(0, x) \) for ReLU networks. The output vector of the network is

\[
y^L = W^L y^{L-1} + b^L.
\]

To represent the activation pattern of ReLU neurons, let us introduce a diagonal matrix \( I^{l,i} \in \mathbb{R}^{d_l \times d_l} \) \((1 \leq l \leq L - 1)\) for each sample and each ReLU layer, whose diagonal entries are defined as \( I_{k,k}^{l,i} = 1 \) if \( W^l(k, \cdot) y^{l-1,i} + b^l_k > 0 \) and \( I_{k,k}^{l,i} = 0 \) otherwise. Consequently, there is \( y^{l,i} = I^{l,i} (W^l y^{l-1,i} + b^l) \) \((1 \leq l \leq L - 1)\) for deep ReLU networks. Network output can then be written as

\[
y^L = W^L I^{L-1,i} (W^{L-1} \cdots (W^1 x_i + b^1) + \cdots + b^{L-1}) + b^L.
\]

The empirical risk for network training is

\[
R(W^1, b^1, \ldots, W^L, b^L) = \frac{1}{N} \sum_{i=1}^{N} l(y^i, y_i)
\]

where \( l \) is the loss function, such as the widely used MSE loss and cross-entropy loss.

To be succinct, the biases will be absorbed in weights (correspondingly, the output vector of each layer will be augmented with a scalar 1). The output of the \( l \)th layer is then written as \( y^{l,i} = \sigma(W^l y^{l-1,i}) \), \( 1 \leq l \leq L - 1 \), and the empirical risk is abbreviated as \( R(W) := R(W^1, \ldots, W^L) \).

C. Convolutional Neural Networks

A CNN usually includes some convolutional layers, pooling (max pooling or average pooling) layers, and fully connected layers. Fully connected layers are defined in the same way as in Section III-B.

Convolutional layers are designed to take advantage of the translational invariance inherent in such data as images. Suppose that the \( l \)th layer is a convolutional layer and the output of each neuron in layer \( l \) is obtained by first taking the inner product between a filter of layer \( l \) and a patch of neighboring neurons at layer \( (l - 1) \), adding the bias and then applying the activation function. Let \( P_l \) and \( s_l \) be, respectively, the number of patches and the size of each patch at layer \( l \). For input \( x \), denote \( \{ y^{l,i}_1, \ldots, y^{l,i}_{P_l} \} \in \mathbb{R}^{s_l \times d_l} \) as the set of patches at layer \( l \). Denote by \( T_l \) the number of convolutional filters and by \( W^l = \{ w^l_1, \ldots, w^l_{T_l} \}^T \in \mathbb{R}^{T_l \times s_l \times d_l} \) the weight matrix of layer \( l \). Each row of \( W^l \) corresponds to one filter. The number of neurons at layer \( l \) is \( n_l = T_l s_l d_{l-1} \). Given the \( p \)th \((p \in \{1, \ldots, T_l \})\) patch \( y^{l-1, p}_{s_l-1} \) and the \( r \)th \((r \in \{1, \ldots, T_l \})\) filter \( w^l_r \), the output of corresponding neuron in layer \( l \) is given as follows:

\[
y^{l,i}_j = \sigma ( (w^l_r y^{l-1, p}_{s_l-1}) + b^l_j )
\]
where $h = (t - 1)p_{t-1} + p$, \( \langle \cdot \rangle \) denotes the inner product. By absorbing the biases into weights and using the notation * to denote convolution, the output of the $l$th layer is abbreviated as $y^l = \sigma(W^l * y^{l-1})$.

A layer $l$ is called a max-pooling layer if the output of its $p$th neuron $y^l_p$ is computed by

$$y^l_p = \max(y^l_{p-1}^{i_1}, \ldots, y^l_{p-1}^{i_{n_{p-1}}}), \quad p \in [P_{l-1}]$$

where $(y^l_{p-1}^{i})$ is the $i$th element in the $p$th patch of layer $(l - 1)$. Similarly, if the average of every patch is computed

$$y^l_p = \text{mean}(y^l_{p-1}^{i_1}, \ldots, y^l_{p-1}^{i_{n_{p-1}}}), \quad p \in [P_{l-1}]$$

then layer $l$ is called an average-pooling layer.

Given the CNN output $\hat{y}_l$ for each sample $x_i$ ($i \in [N]$), the empirical risk for network training is still given by (3).

**D. Definitions: Local Minima and Spurious Local Minima**

$\hat{W} = (\hat{W}^1, \ldots, \hat{W}^L)$ is called a global minimum of loss $R(W)$ if and only if $\forall W, R(\hat{W}) \leq R(W)$. $\hat{W}$ is called a critical (or stationary) point of loss $R(W)$ if and only if $(\partial R/\partial W)(\hat{W}) = 0$. $W^*$ is defined as a local minimum of loss $R(W)$ if and only if there exists an open set $D$, which contains $W^*$ such that $\forall W \in D, R(W^*) \leq R(W)$, and $W^*$ is called a strict local minimum if for all $W$ in $D$ other than $W^*$, there is $R(W^*) < R(W)$. A spurious local minimum is a local minimum $W^*$ satisfying $R(\hat{W}) < R(W^*)$. The local maximum can be defined in a similar way. A saddle point is defined as a critical point, which is neither a local minimum nor a local maximum.

One main goal of this article is to construct spurious local minima that are frequently encountered for the empirical losses of deep fully connected networks and CNNs given in (3).

**IV. MOTIVATING EXAMPLES**

In this section, intuitive examples will be presented to illustrate why spurious local minima are common in deep ReLU networks, and subsequently, the main ideas of this article will be drawn. The key insight is that for neural networks with piecewise linear activations, each output neuron produces a CPWL function of input that is optimized to fit data samples when minimizing the empirical risk. The ReLU activation function is piecewise linear with two pieces. When taking CPWL functions as input, the outputs of both affine transformations and ReLU activations are still CPWL. As a result, any neuron in the hidden and output layers of ReLU networks outputs a CPWL function.

Consider the regression problems using ReLU networks. In Fig. 1, there are $N = 40$ samples lying on a curve whose $x$ coordinates are evenly spaced in interval $[-1, 1]$ and the target is generated by $y_i = x_i^2$ ($i \in [N]$). The empirical risk is

$$R = \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i)$$

where $f(x_i)$ denotes the output of the ReLU network that takes sample $x_i$ as input. The loss function $l$ is assumed to be the mse loss.

Let us partition the samples into several disjoint groups. By doing so, we have

$$R(f) = \frac{1}{N} \sum_{j=1}^{P} \sum_{i=1}^{N_j} l(f(x_i), y_i)$$

where $P$ is the number of groups and $N_j$ is the number of samples in the $j$th group. Fig. 1(a) shows an example of partitioning into two groups. Then, optimal line segments $f_j^*$ ($j \in [P]$) are used to fit different groups of samples by minimizing the mse loss

$$f_j^* = \arg\min_{f_j} \sum_{i=1}^{N_j} l(f_j(x_i), y_i), \quad j \in [P].$$

Such optimal line segments can be obtained by the least-squares method.

Excluding the trivial dataset in which all data samples are identical, there is an intermediate region on the $x$-axis between any two neighboring disjoint groups of samples in which there are no samples. If neighboring optimal line segments intersect at points lying within such intermediate regions, then as shown in Fig. 1(a), a CPWL function $f$ is formed in which all samples of the $j$th ($j \in [P]$) group are predicted by the $j$th optimal line segment and not affected by others. If intersection points of neighboring optimal line segments are not located in desired intermediate regions, one can use auxiliary line segments within the intermediate regions to connect neighboring optimal line segments, as shown in Fig. 1(b). Therefore, CPWL functions can always be constructed to fit data samples. Such CPWL functions can be generated by ReLU networks with appropriate architectures and parameters.

Given a CPWL predictor, the output for each sample will be uniquely determined by a single line segment. No training samples will be predicted by the introduced auxiliary line segments, and thus, they have no effect on the empirical risk. If $f^*$ is the CPWL function composed of linear pieces $f_1^*, \ldots, f_p^*$, then the corresponding empirical risk is

$$R(f^*) = \frac{1}{N} \sum_{j=1}^{P} \sum_{i=1}^{N_j} l(f_j^*(x_i), y_i).$$

Since each optimal line segment $f_j^*$ minimizes the mse loss for a group of samples, see (9), the total loss $R(f)$ is also minimized by $f^*$. When perturbing the line segments around the optimal ones, the CPWL predictor $f$ changes correspondingly. However, if the perturbation is small enough, each sample is still predicted by the same linear piece (with perturbation) as before. Use $f'_j$ to denote the $j$th line segment after perturbation and $f'$ to denote the resulted CPWL predictor, after small perturbation the empirical risk becomes

$$R(f') = \frac{1}{N} \sum_{j=1}^{P} \sum_{i=1}^{N_j} l(f'_j(x_i), y_i).$$

By the fact that $R(f)$ is locally minimized by $f^*$, we have

$$R(f') \geq R(f^*).$$

In other words, the CPWL predictor $f^*$, obtained by fitting line segments to different disjoint groups of samples, is a
local minimum of empirical risk. CPWL predictor \( f^* \) can be produced by ReLU networks with appropriate architectures and parameters (described in detail in Section VI), which means that the parameters producing \( f^* \) are local minima for such networks.

It is easy to see that these local minima are usually spurious. One can partition the samples into more disjoint groups and fit each group linearly by minimizing the mse error as before. When the dataset cannot be fit perfectly by a linear predictor, fitting with more line segments can often reduce the fitting error. Consequently, \( \{f^*_j \mid j \in [P']\} \) with \( P' > P \) (not counting the auxiliary segments) may be a local minimum whose empirical risk is lower than that of \( \{f^*_j \mid j \in [P]\} \), and hence, the parameters of ReLU networks producing \( \{f^*_j \mid j \in [P]\} \) may be spurious local minima. Fig. 1(b) and (c) shows, respectively, the cases of partitioning samples into three and four disjoint groups. There are many possible partition and fitting patterns, and therefore, spurious local minima are common as long as finer-grained fittings produce lower empirical risks.

Fig. 2 shows another example, fitting \( N = 20 \) data points whose \( x \) coordinates are evenly spaced in interval \([0, 1]\) with \( y = x^4 - 1.5x^2 + 1 \). Fig. 2(a)–(c) shows, respectively, the cases of partitioning data samples into two, three, and four disjoint groups. In each case, there is an auxiliary line segment shown in blue to form a CPWL function.

From these simple examples, several key observations can be made.

1) The reason why spurious local minima exist in ReLU networks is given as follows: ReLU networks output CPWL functions to fit discrete samples and different fitting patterns usually yield different fitting errors. These facts are also true for networks with any CPWL activations (such as LeakyReLU [80], [81]), which still generate piecewise linear predictors.

2) The idea of regarding network training as fitting samples with CPWL predictors can be applied to networks and datasets with arbitrary sizes. On one hand, the empirical risk can be zero for overparameterized networks where a sufficient number of linear pieces are produced and each sample can be fit by a single linear piece with zero loss. The remaining linear pieces will not correspond to data samples and act like auxiliary pieces. However, spurious local minima can still exist for overparameterized networks as long as it is possible to fit a group of samples using a single linear piece with nonzero loss. On the other hand, if the number of samples is small enough such that a single linear piece can fit them perfectly, the empirical risk is zero, and thus, no spurious local minima exist. For example, a dataset of two samples in any input dimension will not produce spurious local minima.

Although, in the above examples, regression problems are analyzed with mse loss and 1-D input and output, the principle behind is general and applicable to networks with arbitrary continuous loss functions and piecewise linear activations, and higher dimensional input and output. The general results and their proofs will be given in Sections V and VI.

V. MAIN RESULTS

In this section, the assumptions, main theorems, and the proof sketch will be given.

A. Assumptions

The main theorems of this article are based on Assumption 1.

**Assumption 1 (Data):** The training data cannot be fit by a linear model. i.e., for \( \hat{y}_i = \mathbf{W}x_i + \mathbf{b} \), there is \( \forall (W, b), \sum_{i=1}^{N} l(\hat{y}_i, y_i) \neq 0 \).
For datasets that can be fit by linear models, \( R(f^*) \) will be zero, and thus, no spurious local minima exist. Practical data, such as image set, usually cannot be fit by linear models, and therefore, this assumption is reasonable.

**B. Theorems**

The main results of this article are summarized in Theorems 1 and 2. Theorem 1 is for deep fully connected networks and Theorem 2 is for CNNs.

**Theorem 1:** For datasets satisfying Assumption 1 and deep fully connected ReLU networks, the empirical risk has spurious local minima for networks with as few as two hidden layers and 2 \( d_y \) neurons in each hidden layer. For datasets satisfying Assumption 1 and deep fully connected networks with any other CPWL activations, the empirical risk has spurious local minima for networks with as few as two hidden layers and 3 \( d_y \) neurons in each hidden layer. The number of nonhomogeneous spurious local minima grows with the size of deep fully connected networks.

**Theorem 2:** For datasets satisfying Assumption 1 and average-pooling CNNs with ReLU activations, the empirical risk has spurious local minima for networks with as few as two fully connected hidden layers, 2 \( d_y \) neurons in each fully connected hidden layer and 2 \( d_y \) filters in each convolutional layer. For datasets satisfying Assumption 1 and average-pooling CNNs with any other CPWL activations, the empirical risk has spurious local minima for networks with as few as two fully connected hidden layers, 3 \( d_y \) neurons in each fully connected hidden layer and 2 \( d_y \) filters in each convolutional layer. The number of nonhomogeneous spurious local minima grows with the size of average-pooling CNNs.

In these theorems, the meaning of homogeneousness is given as follows. Since multiplying a constant \( a \neq 0 \) with the weights of one layer and multiplying \((1/a)\) with the weights of an adjacent layer will not change the empirical risk, one can scale the weights of a spurious local minimum in this way to get another spurious local minimum. Such homogeneousness is excluded when counting the number of spurious local minima in Theorems 1 and 2.

The detailed constructions of network parameters to produce spurious local minima will be given in the following proof sketch and the complete proofs will appear in Section VI. It should be pointed out that although the explicit constructions in this article require a width of 2 \( d_y \) by producing the CPWL predictor of each component independently, spurious local minima exist for much smaller deep networks when \( d_y \gg 1 \) (the explanation will be given in Remark 1).

**C. Proof Sketch**

In principle, networks with any CPWL activations will output CPWL predictors, and when using such CPWL predictors to fit data samples, the existence of different fitting patterns leads to spurious local minima. Generally speaking, spurious local minima can be shown by any networks that can produce locally optimal CPWL predictors. In this article, following the idea given in Section IV, Theorems 1 and 2 will be proved by constructing deep networks that produce spurious local minima in an intuitive way.

The key proof steps and their rough ideas are given as follows.

1) **Partition the Domain Into Disjoint Convex Polytopes:** Partition the domain of interest that contains all samples \( \{x_i \in \mathbb{R}^d, i \in [N]\} \) into some disjoint convex polytopes. Data samples are then divided into disjoint groups by these polytopes. Such partition is always possible, and since there are gaps between neighboring samples, one can always construct partitions where there exist intermediate empty regions between adjacent groups of samples.

2) **Fit Each Group of Samples With a Linear Predictor:** If the intersections (which are convex) of adjacent linear predictors do not lie within the intermediate regions between adjacent groups of samples, the intermediate regions are then further partitioned into finer polytopes and an auxiliary linear piece is introduced for each finer polytope, such as the auxiliary line segment in Fig. 1(b). The purpose of introducing such auxiliary linear pieces is to form a CPWL predictor. All samples in the same group will be predicted by a single linear piece.

3) **Formulate the CPWL Predictor as a Maximum Over Minimum of Affine Functions:** It has been shown [82] that any CPWL function can be formulated as \( \max_i (\min_j K_j f_j(x)) \), where \( f_j \) is the \( j \)th linear piece of the CPWL predictor, and the meaning of \( K_j \) will be made clear in Section VI.

4) **Construct a Deep Fully Connected Network or a CNN With Arbitrary CPWL Activation to Produce the CPWL Predictor:** By designing appropriate network architectures and parameters, the desired CPWL predictors can be generated in the form of \( \max_i (\min_j K_j f_j(x)) \).

5) **Show That Network Parameters Producing Desired CPWL Predictors Are Local Minima:** This can be seen by showing that small perturbation of network parameters does not change the fitting pattern, i.e., each sample is still predicted by the original linear piece (with perturbation) responsible for it and by the fact that empirical risk of each group of samples has been minimized by corresponding linear piece in step 2.

6) **Show That Local Minima Producing CPWL Predictors Are Spurious:** Take a group of samples with nonzero empirical risk and select a sample in it with nonzero loss, and further divide the region of this group into some finer convex polytopes such that only the single selected sample is located in a specific polytope. By fitting this sample with a single linear piece and thereby reducing its loss to zero and optimally fitting other groups of samples in remaining finer polytopes, the total empirical risk will be decreased. In other words, there exist finer partitions of samples with lower empirical risks, and thus, the local minima resulted from coarser partitions are spurious.

**VI. PROOFS**

In this section, following the proof sketch in Section V-C, the detailed proofs will be presented. The proofs are general and can be applied to any continuous loss function, any dataset.
A. Deep Fully Connected Networks With Arbitrary CPWL Activations

1) Partition Data Samples Into Disjoint Groups: Let us partition the domain of interest $D$ into some disjoint convex polytopes $\{R_i, i \in [P]\}$ such that $D = \bigcup_{i=1}^{P} R_i$ and $R_i \cap R_j = \emptyset$ (i ≠ j), where $P$ is the number of partitions. The samples are correspondingly divided into groups and samples in each group are located in the same polytope. It is required that there exist intermediate empty (no samples) regions between adjacent groups of samples. Such partition always exists since there are gaps between data points. It is possible that some polytopes contain no samples.

2) Fit Each Group of Samples With a Linear Predictor: The samples in the same polytope are then fit using a linear predictor. Formally, the optimal linear predictor $f_i^*$ (which is a vector if $d_x > 1$) for the $i$th group of samples is given by

$$f_i^* = \arg \min_{f_i} \sum_{x_k \in R_i} l(f_i(x_k), y_k), \quad i \in [P] \quad (13)$$

where $f_i(x) = Ax + b \in \mathbb{R}^{d_y}$ is a linear function of input $x$ with $A \in \mathbb{R}^{d_x \times d_y}$ and $b \in \mathbb{R}^{d_y}$, region $R_i$ is the polytope corresponding to the $i$th group of samples, and $f_i^*$ always exists for arbitrary loss function $l$.

If the convex intersections of adjacent optimal linear predictors are located outside the corresponding intermediate regions, in order to form a proper CPWL predictor, the involved adjacent polytopes are shrunk slightly such that the grouping of samples does not change. Then, the empty intermediate regions between shrunk polytopes are partitioned into some auxiliary polytopes (such as partitioned into stripes of simplexes). In each auxiliary polytope, a linear piece is constructed that connects the linear pieces in adjacent polytopes such that a CPWL function is formed. There are no samples in auxiliary polytopes, the total empirical risk is thus not affected by them.

Define $f^*$ as the CPWL function composed of linear pieces $f^*_i$ (i ∈ [P]) and auxiliary pieces, i.e., $f^* = f^*_i$ in the $i$th region, and then, the total empirical risk is given by

$$R(f^*) = \frac{1}{N} \sum_{i=1}^{P} \sum_{k=1}^{N_i} l(f^*_i(x_k), y_k). \quad (14)$$

3) Formulate the CPWL Predictor as a Maximum Over Minimum of Linear Pieces: Given a scalar CPWL function $f$ and a set of linear functions $\{f_i, i \in [P]\}$, suppose that there are $P$ regions $\{R_i\}$ in the domain, and in each region, there is $f = f_i$ (i ∈ [P]), i.e., each region corresponds to a single linear piece. By [82], any CPWL function $f$ can be formulated as $f = \max_i (\min_{j=1}^{K_i} f_j)$. More specifically,

$$f = \max_{i=1}^{P} \Psi_i \quad (15)$$

with

$$\Psi_i = \min_{j=1}^{K_i} f_j \quad \forall f_j \mid f_j \geq f_i \text{ in every point of region } R_i. \quad (16)$$

$K_i$ is the number of affine functions $\{f_j\}$ that satisfy $f_j \geq f_i$ in every point of $R_i$.

The process of computing $f$ can be illustrated by the example shown in Fig. 3. There are three affine functions in Fig. 3: $f_1$, $f_2$, and $f_3$, and the blue thick line segments represent the CPWL function $f$. The real domain is divided into three disjoint regions: $R_1$, $R_2$, and $R_3$, and in each region, the CPWL function equals a specific affine function. $\Psi_1$ corresponding to region $R_1$ is $\Psi_1 = \min(f_1, f_2)$; similarly, $\Psi_2 = \min(f_1, f_2)$ and $\Psi_3 = f_3$. Consequently, $\Psi_1 = f_1$ in region $R_1$, $\Psi_1 = f_2$ in region $R_2$, and $\Psi_1 = f_3$ in region $R_3$. Note that $\Psi_1$ may vary from region to region. Similarly, $\Psi_2 = f_1$ in region $R_1$, $\Psi_2 = f_2$ in region $R_2$, and $\Psi_2 = f_3$ in region $R_3$, $\Psi_3 = f_3$ in all regions. Then, by (15), $f = \max_{i=1}^{P} \Psi_i$, so $f = \max(f_1, f_3) = f_1$ in region $R_1$, $f = \max(f_2, f_3) = f_2$ in region $R_2$, and $f = \max(f_2, f_3) = f_3$ in region $R_3$. This is exactly the desired CPWL function shown in blue thick line segments. The interested readers are referred to [82] for detailed proof.

4) Construct Deep Networks to Produce the CPWL Predictor: Given any dataset satisfying Assumption 1, deep networks with appropriate architectures and parameters will be designed in this step to produce the CPWL function formulated in (15) and (16). Since, in step 3, the method of representing a CPWL predictor as a maximum over minimum of linear pieces is designed for scalar functions, so each component of the optimal CPWL predictor $f^*$ will be produced independently and then concatenated. In the following, deep networks producing scalar CPWL predictors will be mainly described. Deep ReLU networks will be presented at first, followed by deep networks that use arbitrary CPWL activations.

a) Deep ReLU networks: Given an optimal scalar CPWL function $f^*$ and its constituent linear pieces $\{f^*_i, i \in [P]\}$, each linear piece $f^*_i$ can be produced by a single ReLU neuron that outputs $\sigma(w_i^T x + b_i)$, where weight vector $w_i$ and bias $b_i$ are parameters of $f^*_i$. These ReLU neurons producing linear pieces $\{f^*_i, i \in [P]\}$ are put at the first hidden layer that directly connects to input $x$. Since the ReLU activation truncates negative input, in order to preserve the shape of linear pieces, a large positive constant bias $c$ is added to each ReLU neuron in the first hidden layer such that its output is positive in the domain of interest. By doing so, each ReLU neuron in the first hidden layer outputs a linear piece in the domain of interest $D$

$$f^*_i = \sigma(w_i^T x + b_i + c) = w_i^T x + b_i + c > 0 \quad \forall x \in D, \ i \in [P]. \quad (17)$$
The effect of this large positive constant bias will be canceled at the output layer by subtracting \( c \) from each output component.

The next question is how to implement the maximization and minimization operations using ReLU networks. Minimization can be written as

\[
\min(a, b) = b - \max(0, b - a) = \sigma(\sigma(b) - \sigma(b - a))
\]

and thus, minimization can be represented by a two-layer ReLU network with two hidden neurons and the specific weights shown in Fig. 4(a). The minimization over \( K_i > 2 \) affine functions can be implemented hierarchically, where a group of minimizations over two affine functions are obtained at each level and the intermediate results are fed into the next level, until finally \( \min_{K_i=1} f_1 \) is obtained. Fig. 4(c) shows the hierarchical architecture used to compute the minimization over four affine functions.

The maximum over two variables can be written as

\[
\max(a, b) = a + \max(0, b - a) = \sigma(\sigma(a) + \sigma(b - a))
\]

Similarly, \( \max^p_i \Psi_i \) can be implemented hierarchically.

By using identity weight matrices in higher layers, the results of maximization over minimization are then propagated to the output layer without altering them until finally each neuron in the output layer produces the desired CPWL predictor. The neurons not involved in above computations are set with zero weight vectors and biases.

b) Deep networks that use arbitrary CPWL activations: If an arbitrary CPWL activation function \( \sigma_1 \) (such as LeakyReLU [80], [81] and those in [83], [84], [85], [86], and [87]) is used, we can also construct a deep network to produce the CPWL predictor. The idea is to construct a locally ReLU-shaped piecewise linear activation function using \( \sigma_1 \) and its shifted version, and then scale and shift the network parameters such that the outputs of relevant neurons are localized in this ReLU-shaped region. Without loss of generality, suppose that there are at least two linear pieces in \( \sigma_1 \), and two neighboring pieces in it with finite slopes will be utilized. Let the slopes and intercepts of the two neighboring pieces be \( k_1 \) and \( k_2 \) and \( b_1 \) and \( b_2 \), respectively, and the intersect point of these two pieces be \((u', v')\). Therefore,

\[
v' = k_1 \cdot u' + b_1 = k_2 \cdot u' + b_2.
\]

We have \( k_1 \neq k_2 \) since they are slopes of neighboring pieces. \( \sigma_1 \) can be locally expressed as

\[
\sigma_1(u) = \begin{cases} 
  k_2 \cdot u + b_1, & \text{if } u'' \leq u < u' \\
  k_2 \cdot u + b_2, & \text{if } u' \leq u \leq u''
\end{cases}
\]

where \((u'', u')\) and \((u', u'')\) are the endpoints of the two linear pieces. We then have

\[
u = \frac{1}{k_2} \sigma_1(u) - \frac{b_2}{k_2}, \quad u' \leq u \leq u''.
\]

This means that given an input \( u \), after passing through the neuron and getting \( \sigma_1(u) \), we can adjust the weight and bias of each successor by multiplying \((1/k_2)\) with the weight and adding \(- (b_2/k_2)\) to the bias, and then, the value of \( u \) will be maintained.

Given \( \sigma_1(u) \), a new function is constructed using it and its shifted version as follows:

\[
\eta(u) = \sigma_1(u) - \sigma_1(u - c') - k_1 \cdot c' 
\]

where \( 0 < c' < \min(u'' - u', u' - u'') \) is the amount of shift. By simple calculation, we have

\[
\eta(u) = \begin{cases} 
  0, & \text{if } u'' + c' \leq u < u' \\
  (k_2 - k_1) \cdot u + (b_2 - b_1), & \text{if } u' \leq u < u' + c' \\
  (k_2 - k_1) \cdot c', & \text{if } u' + c' \leq u \leq u''.
\end{cases}
\]

For example, in the interval \( u' \leq u < u' + c' \), we have

\[
\eta(u) = k_2 \cdot u + b_2 - (k_1 \cdot (u - c') + b_1) - k_1 \cdot c' = (k_2 - k_1) \cdot u + (b_2 - b_1).
\]

A function \( \eta_1(u) \) is then defined as follows:

\[
\eta_1(u) = \frac{1}{k_2 - k_1} \eta(u + u')
\]

and we can obtain

\[
\eta_1(u) = \begin{cases} 
  0, & \text{if } u'' + c' - u' \leq u < 0 \\
  u, & \text{if } 0 \leq u < c' \\
  c', & \text{if } c' \leq u \leq u'' - u'
\end{cases}
\]

where (20) is utilized for the case \( 0 \leq u < c' \). As desired, the first two segments of \( \eta_1(u) \) perform exactly the ReLU operation.

Now, we can express the minimization and maximization operations using the CPWL activation function \( \sigma_1 \). Take \( \min(a, b) \) as an example. If \( u' \leq a, b \leq u', u'' + c' - u' \leq b - a < c' \), we have

\[
\min(a, b) = b - \max(0, b - a) = b - \eta_1(b - a) = b - \frac{1}{k_2 - k_1} (\sigma_1(b - a + u') - \sigma_1(b - a + u' - c') - k_1 \cdot c') = \frac{1}{k_2} \sigma_1(b) - \frac{1}{k_2 - k_1} \sigma_1(b - a + u') + \frac{1}{k_2 - k_1} \sigma_1(b - a + u' - c') + \left( \frac{k_1 \cdot c' - b_2}{k_2 - k_1} \right).
\]

Therefore, a minimization operation can be implemented by a small network with activation function \( \sigma_1 \), two input neurons \( (a, b) \), three hidden neurons, and one output neuron. Compared with the expression in (18) for ReLU networks, only one more hidden neuron is needed. The expression for maximization operation can be derived similarly.

Finally, for the inputs \( a \) and \( b \) in minimization operation \( \min(a, b) \), we need to make sure that \( u' \leq a, b \leq u' \), and \( u'' + c' - u' \leq b - a < c' \) are satisfied so that (28) holds. Given an optimal scalar CPWL function \( f^* \) and its linear pieces \( \{ f^*_i, i \in [P] \} \), a positive constant bias \( c_1 = u' + (u'' - u')/2 \), which is the middle of the second
piece, is added to each neuron in the first hidden layer. \((w_i, b_i)\) are scaled using a sufficiently small positive constant \(\lambda\). If \(\lambda \ll (\min(|u'' + c' - u'|, c')) / (\max(||w_i||_2, |b_i|))\), then for any sample \(x\) in the domain of interest \(D\), \(\lambda w_i^2 x + \lambda b_i + c_i\) will be located in a small interval around \(c_i\).

By \(\lambda f^*_j(x) + c_i = \lambda w_i^2 x + \lambda b_i + c_i\), we then have

\[
\begin{align*}
    u' &\leq \lambda f^*_j(x) + c_i \leq u'', \quad i \in [P] \\
    u'' + c' - u' &\leq (\lambda f^*_j(x) + c_i) - (\lambda f^*_j(x) + c_i) < c', \quad i \neq j.
\end{align*}
\]

(29)

\(\{\lambda f^*_j(x) + c_i, \ i \in [P]\}\) are taken as the inputs to the maximization over four affine functions. The first hidden layer neurons are used to compute \(\{\lambda f^*_j(x) + c_i, \ i \in [P]\}\).

Because

\[
\sigma_1(\lambda f^*_j(x) + c_i) = k_2(\lambda f^*_j(x) + c_i) + b_2, \quad x \in D, \ i \in [P]
\]

(30)

as in (22), we can adjust the weights and biases of successors to get \(\{\lambda f^*_j(x) + c_i, \ i \in [P]\}\). All neurons in higher layers can be treated in the same way as in (22) if we want to make their outputs equal to their inputs. The effects of \(c_1\) and \(\lambda\) are canceled at the output layer by adding \(-c_1\) to each output component and then multiplying with \(1/\lambda\).

To produce CPWL predictors, we can see that the basic modules are similar in both deep ReLU networks and deep networks that use arbitrary CPWL activations. Therefore, the following will be focused on deep ReLU networks, and deep networks with arbitrary CPWL activations will be mentioned only if necessary.

5) Show That Parameters Producing CPWL Predictors Are Local Minima: \(\{f^*_j, \ i \in [P]\}\) minimizes the empirical risk of each region of samples. In the following, it will be shown that there exists a small neighborhood around the optimal parameters in which the fitting pattern does not change, i.e., each data sample is still fit by the linear piece originally responsible for it.

ReLU network architectures have been presented in step 4 to produce each component of \(f^*\) independently. By arranging the components of \(f^*\) sequentially, the output of network that produces \(f^*\) can be expressed as

\[
\hat{y}^i = W^L I^{L-1,i} W^{L-1} I^{L-2,i} W^{L-2}, \ldots, W^1 x_i.
\]

(31)

Here, the biases have been absorbed in weights. The weight matrix \(W^1\) in the first layer specifies the linear pieces \(f^L_j, \ i \in [P], j \in [d_j]\) for different regions and components, where \(f^L_j\) is the \(j\)th component of \(f^L\). The layers from 2 to \(L\) with associated weight matrix \(W^l (l = 2, 3, \ldots, L)\) implement the minimization and maximization operations of each component used to produce the CPWL predictor. \(\{I^{l,j}, l = 2, 3, \ldots, L - 1\}\) are determined by the results of these minimization and maximization operations, and some ReLU neurons in layers 2 to \(L - 1\) may be inactive.

Given fixed parameters, output \(\hat{y}\) will be a linear function of input \(x\) if \(\{I^{l,j}, l \in [L - 1]\}\) remain constant. The samples in each group are predicted by the same linear predictor and hence they share the same activation pattern, i.e., \(\{I^{l,j}, l \in [L - 1]\}\) are identical for all samples \(x_i\) in the same group and so their outputs \(\hat{y}^i\) have the same linear expressions. Given the optimal linear predictors \(\{f^*_j, i \in [P]\}\) and the fact that \(\forall x_i \in R_j\), \(f^*_j(x_i) = f^*_i(x_i) = W^L I^{L-1,i} W^{L-1} I^{L-2,i} W^{L-2}, \ldots, W^1 x_i\), the empirical risk can be written as

\[
R(W^1, \ldots, W^L) = \frac{1}{N} \sum_{j=1}^{p} \sum_{i=1}^{N_j} l(f^*_j(x_i), y_i).
\]

(32)

When perturbing the parameters that produce \(f^*\) as \(W^l = W^l + \delta W^l (l \in [L]),\) the linear predictors \(f^*_j\) and activation patterns \(I^{l,j}\) will be perturbed correspondingly. However, if the perturbation is small enough, \(\{I^{l,j}, i \in [N], l \in [L - 1]\}\) will keep constant. Let \(\epsilon = \min_{(l,m,n)} |\delta W^l(m,n)|\) such that \(\{I^{l,j}, i \in [N], l \in [L - 1]\}\) keep constant after perturbation, and then, the forms of linear predictors \(f^*_j, j \in [P]\) do not change after perturbing with \(\max_{(l,m,n)} |\delta W^l(m,n)| \leq \epsilon\).

\[
f^*_j(x_i) = (W^L + \delta W^L) I^{L-1,i} (W^{L-1} + \delta W^{L-1}) \times \cdots (W^1 + \delta W^1) x_i
\]

(33)

After rearrangement

\[
\begin{align*}
    f^*_j(x_i) &= W^L I^{L-1,i} W^{L-1} I^{L-2,i} W^{L-2}, \ldots, W^1 x_i \\
    &+ \delta W^L I^{L-1,i} W^{L-1} I^{L-2,i} W^{L-2}, \ldots, W^1 x_i \\
    &+ W^L I^{L-1,i} \delta W^{L-1} I^{L-2,i} W^{L-2}, \ldots, W^1 x_i \\
    &+ \cdots + \delta W^L I^{L-1,i} \delta W^{L-1} I^{L-2,i} \delta W^{L-2}, \ldots, \delta W^1 x_i
\end{align*}
\]

(34)

where \(\delta f^*_j(x_i) = [\delta W^L I^{L-1,i} W^{L-1} I^{L-2,i} W^{L-2}, \ldots, W^1 + W^L I^{L-1,i} \delta W^{L-1} I^{L-2,i} W^{L-2}, \ldots, \delta W^{L-2}, \ldots, \delta W^1] x_i\).
\[ \delta W^L | I^{L-1} \delta W^{L-1} I^{L-2} \delta W^{L-2} \ldots, \delta W^1 | x \], the form of \( \delta f_j \) is the same for all samples in a group since they share the same activation pattern \( \{ I^l, x_i \in R_j, l \in [L-1] \} \), and consequently, these samples share the same linear predictor after perturbation. Note that \( f_j^*(x) + \delta f_j(x) \) is still a linear function of \( x \in R_j \).

The empirical risk after perturbation is given by

\[ R(W^1, \ldots, W^L) = \frac{1}{N} \sum_{j=1}^{P} \sum_{i=1}^{N_j} l(f_j^*(x_i), y_i). \]  

(35)

However, since \( f_j^* \) minimizes \( \sum_{i=1}^{N_j} l(f_j(x_i), y_i) \), there exist a small neighborhood around \( (W^1, \ldots, W^L) \) such that, for all points \( (W^1, \ldots, W^L) \) satisfying \( \max_{l,m,n} |W^l(m,n) - W^l(m,n)| < \epsilon \), there is

\[ \sum_{i=1}^{N_j} l(f_j^*(x_i), y_i) \leq \sum_{i=1}^{N_j} l(f_j(x_i) + \delta f_j(x_i), y_i), \quad j \in [P]. \]  

(36)

Consequently, we have

\[ R(W^1, \ldots, W^L) \leq R(W^1, \ldots, W^L) \]  

in a small neighborhood around \( (W^1, \ldots, W^L) \) with \( \max_{l,m,n} |W^l(m,n) - W^l(m,n)| < \min(\epsilon, \epsilon') \). Therefore, \( W = (W^1, \ldots, W^L) \) is a local minimum for the network constructed in step 4.

6) Show That Local Minima Producing CPWL Predictors Are Often Spurious: In this step, it will be shown that the local minima constructed in step 4 are usually spurious. Take a group of samples with a nonzero empirical risk (without loss of generality, suppose that it is the \( k \)th group), i.e., \( \sum_{i=1}^{N_j} l(f(x_i), y_i) \neq 0 \). By Assumption 1, such group always exists, for example, by fitting all samples with a single linear predictor. Due to the nonnegative nature of loss function, there must be some samples in the \( k \)th group with nonzero losses. Pick any sample \( x_0 \) with a nonzero loss in the \( k \)th group, the empirical risk can then be expanded as

\[ R = \frac{1}{N} \sum_{j=1}^{P} \sum_{i=1}^{N_j} l(f_j(x_i), y_i) + \frac{1}{N} \sum_{i=1}^{N_k} l(f(x_i), y_i) + \frac{1}{N} l(f(x_0), y_0). \]  

(38)

Let us further partition the region \( R_k \) into some disjoint convex subregions such that a subregion contains the single point \( x_0 \), and other samples are contained in the remaining subregions. Regions other than \( R_k \) remain intact. The minimal possible value of loss function \( f \) is zero, so there exists \( \tilde{y}_n \) such that \( l(f_\hat{x}(x_0), \tilde{y}_n) = 0 \). Then, the single sample \( x_0 \) can be fit by a linear predictor \( \tilde{f}(x_0) = A x_0 + \tilde{b} \), where \( A \in \mathbb{R}^{d \times d} \), and \( \tilde{b} \in \mathbb{R}^d \) are parameters of \( \tilde{f} \). Because there always exists \( (\tilde{A}, \tilde{b}) \) satisfying \( \tilde{A} x_0 + \tilde{b} = \tilde{y}_n \), we have

\[ l(\tilde{f}(x_0), y_0) = 0. \]  

(39)

The remaining samples in region \( R_k \) have been partitioned into subgroups and each subgroup is again optimally fit by linear predictors. Suppose that for the \( j \)th subgroup (excluding the one containing \( x_0 \)) of region \( R_k \), the new optimal linear predictor is \( \tilde{f}_j \) and there are \( N_{kj} \) samples in it; then,

\[ \sum_{i=1}^{N_{kj}} l(\tilde{f}_j(x_i), y_i) \leq \sum_{i=1}^{N_{kj}} l(f(x_i), y_i). \]  

(40)

Finally, the combination of (38)–(40) yields

\[ \hat{R} = \frac{1}{N} \sum_{j=1}^{P} \sum_{i=1}^{N_j} l(f_j(x_i), y_i) + \frac{1}{N} \sum_{j=1}^{N_k} l(f_j(x_i), y_i) + \frac{1}{N} \sum_{j=1}^{N_{kj}} l(\tilde{f}_j(x_i), y_i) \]

\[ + \frac{1}{N} l(f(x_n), y_n) = R \]  

(41)

where the fact \( l(f(x_n), y_n) > 0 \) has been used for the inequality. This indicates that lower empirical risks may be obtained with finer partitions, and hence, \( W = (W^1, \ldots, W^L) \) is a spurious local minimum. Whether finer partitions with lower empirical risks exist depends on data and current partition of samples.

7) Size of Deep Fully Connected Networks With Spurious Local Minima: For a CPWL fitting with two linear pieces \( (f_1, f_2) \), at most one minimization or one maximization operation is required for each component of \( f_1 \) and \( f_2 \), and thus, a ReLU network with two hidden layers (one is for producing affine functions \( f_1 \) and \( f_2 \) and the other is for minimization or maximization) and 2 \( d_e \) neurons in each hidden layer (for each component, the width of minimization or maximization module is 2, as shown in Fig. 4) is enough to represent this CPWL function. For data samples satisfying Assumption 1, i.e., they cannot be fitted by linear models, compared with fitting by a single linear predictor (which apparently can be produced by the same network), there always exist two-piece CPWL fittings with lower empirical risks. Therefore, spurious local minima exist for deep ReLU networks with as few as two hidden layers and 2 \( d_e \) neurons in each hidden layer and datasets that cannot be fit by linear models.

For a CPWL fitting with \( n > 2 \) linear pieces, there are at most \( n \) affine functions in each \( \Psi_i \), and thus, the network can include at most 4 \( \lceil \log_2 n \rceil \) hidden layers, with 2 \( \lceil \log_2 n \rceil \) layers for the hierarchical minimization required by each \( \Psi_i \) and another 2 \( \lceil \log_2 n \rceil \) layers for the hierarchical maximization over \( \{ \Psi_i, i \in [n] \} \). The width of a hidden layer is at most \( n^2 d_e \), due to the fact that hierarchical maximization requires a width of at most \( n \) neurons for each \( \Psi_i \) and each component.

For bigger deep ReLU networks, more finer partitions and fittings can be generated, and consequently, more spurious local minima exist, leading to their prevalence. The actual widths of networks possessing spurious local minima can be much less than \( n^2 d_e \), see Remark 1 later in this section.

For deep networks with arbitrary CPWL activations, since three hidden neurons are required to implement the minimization or maximization operation, spurious local minima exist for networks with as few as two hidden layers and 3 \( d_e \) neurons in each hidden layer.
B. Convolutional Neural Networks

The main differences between CNNs and fully connected networks come from the convolutional layers and pooling layers. In this section, it will be proved that spurious local minima are common for CNNs with convolutional layers, average-pooling layers, and fully connected layers.

The main idea is given as follows. The convolution is a linear operation. Let each ReLU neuron in the first convolutional layer compute a positive value \( \sigma((w^l_i x_p) + b^l_i + c) = (w^l_i x_p) + b^l_i + c \), where \( w^l_i \) is the \( i \)th convolutional filter in the first convolutional layer, \( x_p \) is the \( p \)th input patch, and \( c \) is a large positive constant to ensure a positive result and its effect is canceled at the output layer by subtracting \( c \) from each output neuron. The output of the first convolutional layer is thus a linear function of input \( x \) in the domain of interest. Average pooling is also a linear operation. Therefore, a CNN stacked by convolutional layers, average-pooling layers, and fully connected layers can produce a linear output in the domain of interest. This fact is also true for CNNs with arbitrary CPWL activations, by use of \( c_1 \) and \( \lambda \) (see Section VI-A4). The following description will be focused on CNNs with ReLU activation.

One can partition the data samples into groups as before and each group of samples can be optimally fit using a linear predictor. CNN architectures can be constructed to generate these optimal linear predictors, along with the point-wise minimization and maximization operations over them. Like the case of deep fully connected networks, under small perturbation of parameters, the expression of empirical risk for each group of samples remains constant, leading to the existence of local minima. If finer partitions produce lower fitting errors, then these local minima are spurious.

The CNN construction for producing the desired CPWL predictors is not unique. In general, the output of a CNN is
\[
\hat{y} = W^L L^{L−1}W^{L−1} L^{L−2}W^{L−2} \cdots, W^1
\]
\[
\times L^{l_c−1}W^{l_c−1} W^{l_c−2} A, \ldots, *W^1 * x_i
\] (42)
where \( * \) is the convolution operation and \( A \) is the average-pooling operation. Layers from \( l_c \) to \( L \) are fully connected layers. For convolutional layers, \( l_i \) (\( i \in [l_c − 1]) \) is defined for every neuron in the feature maps. A detailed CNN construction is given as follows to produce the desired CPWL predictors.

The optimal linear predictors for different groups of samples will be obtained independently. Let the linear predictor for the \( j \)th group of samples be \( f_j^* \), with elements being \( f_{j,k}(x_i) = w^l_{j,k} \hat{f} (w^{l−1}_{j,k} * \ldots, A, \ldots, *w^1 * x_i) \) (\( j \in [P], k \in [d], x_i \in R^2 \)), where \( f_{j,k} \) is the \( k \)th component of \( f_j \), \( \{w^l_{j,k}, l \in [l_c − 1] \} \) are convolutional filters, and \( w^1_{j,k} \) is the weight vector for the first fully connected layer. All weights in the \( l \)th (\( l > 1 \)) convolutional layers and the first fully connected layer are fixed to 1, i.e., \( w^l_{j,k} = (1,1,\ldots,1)^T \in \mathbb{R}^{v−1} \) (\( 2 \leq l < l_c \)) and \( w^l_{j,k} = (1,1,\ldots,1)^T \). The biases in these layers are fixed to 0. Therefore, only the filters and biases in \( w^l_{j,k} \) will be optimized. The reason why the first fully connected layer is used to produce the predictors is that each neuron in the first fully connected layer has a receptive field as large as the whole image, so all components of \( x \) can be utilized in the predictors.

Each optimal linear predictor \( f^*_j \) is then obtained by minimizing the empirical loss \( \sum_{i=1}^{N} l(f(x_i), y_i), (j \in [P], x_i \in R^2) \) through tuning the filters and biases in \( w^l_{j,k} \). After that, the filters and biases from different regions and components are concatenated to form parameter matrix \( W^l (1 \leq l \leq l_c) \) for each convolutional layer and the first fully connected layer. In (42), each \( l_i \) (\( i \in [l_c] \)) for these layers is equal to an identity matrix since all involved ReLU neurons are active due to the introduction of bias \( c \). Finally, the fully connected layers above layer \( l_c \) are used to compute the maximization over minimization of linear pieces \( \{f^*_j \} \).

For a dataset partitioned into two groups, the above CNN construction needs at least one convolutional layer with \( 2 d_y \) convolutional filters, two fully connected hidden layers (the first fully connected layer is used in computing optimal linear predictors and the second is used for computing \( \min(f_{1,k}, f_{2,k}) \) or \( \max(f_{1,k}, f_{2,k}) \)) with \( 2 d_y \) neurons in each fully connected hidden layer (or \( 3 d_y \) neurons if piecewise linear activations other than ReLU are used). For datasets that cannot be fit by linear models, there always exist two-piece CPWL fittings with empirical risks lower than that of fitting all samples with a single linear predictor. Therefore, spurious local minima exist for such CNNs. For bigger CNNs, finer partitions and fittings can be generated, and consequently, more spurious local minima exist. The number of possible fitting patterns depends on data and the number of linear pieces a network can produce.

Remark 1: The width of \( 2 d_y \) in the proposed architecture is obtained by producing the CPWL predictor independently for each output component. For some typical CNN architectures, such as AlexNet [1] and VGG [2] for ImageNet classification, this width requirement is satisfied by fully connected layers (both AlexNet and VGG have two fully connected hidden layers, and each has a width of 4096). However, it must be pointed out that the width of \( 2 d_y \) is unnecessary in practice because the parameters of all layers can be jointly optimized to fit all components simultaneously.

VII. CONCLUSION

In this work, the loss landscapes of deep fully connected neural networks and average-pooling CNNs with piecewise linear activations have been studied from a theoretical perspective. It is shown that spurious local minima are common for these networks. Each output neuron of deep fully connected networks and CNNs with piecewise linear activations produces a CPWL function. When minimizing the empirical risk, different pieces of CPWL output can optimally fit disjoint groups of data samples. Each fitting pattern corresponds to a local minimum. Distinct fitting patterns usually result in different levels of empirical risk, and thus, spurious local minima are common. CPWL predictors are represented as maximization over minimization of linear pieces, and deep fully connected networks and average-pooling CNNs with piecewise linear activations are constructed to produce them.
In future work, it is interesting to explore the gap between empirical risks of different spurious local minima. It is also worthwhile to investigate how gradient-based optimization converges to spurious local minima.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent., 2015, pp. 1–14.

[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 9, pp. 1137–1149, Jun. 2017.

[4] E. Shelhamer, J. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 640–651, Apr. 2017.

[5] Y. Ji, H. Zhang, Z. Zhang, and M. Liu, “CNN-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances,” Inf. Sci., vol. 546, pp. 835–857, Feb. 2021.

[6] Z. Zhang et al., “Dense residual network: Enhancing global dense feature flow for character recognition,” NeuroD Svg., vol. 139, pp. 77–85, Jul. 2021.

[7] F. He and D. Tao, “Recent advances in deep learning theory,” 2020, arXiv:2012.10931.

[8] R. Sun, D. Li, S. Liang, T. Ding, and R. Srikant, “The global landscape of neural networks: An overview,” IEEE Signal Process. Mag., vol. 37, no. 5, pp. 95–108, Sep. 2020.

[9] I. Safran and O. Shamir, “Spurious local minima are common in two-layer ReLU neural networks,” in Proc. 35th Int. Conf. Learn. Represent., 2018, pp. 4433–4441.

[10] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, “Local minima in training of neural networks,” 2016, arXiv:1611.06310.

[11] C. Yun, S. Sra, and A. Jadbabaie, “Small nonlinearities in activation functions create bad local minima in neural networks,” in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–13.

[12] Y. Zhou and Y. Liang, “Critical points of neural networks: Analytical forms and landscape properties,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–18.

[13] A. Shariﬁnassab, S. Salehkaleybar, and S. J. Golestani, “Bounds on over-parametrization for guaranteed existence of descent paths in shallow ReLU networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–25.

[14] F. He, B. Wang, and D. Tao, “Piecewise linear activations substantially affect the behaviour of the optimizer,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 1–11.

[15] K. Kawaguchi and Y. Bengio, “Depth creates no bad local minima,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 1–9.

[16] L. Zhang, “Depth creates no more spurious local minima,” 2019, arXiv:1901.09827.

[17] M. Sohtnluko t a bi, A. Javanmard, and J. D. Lee, “Theoretical insights into the optimization landscape of over-parameterized shallow shallow networks,” IEEE Trans. Inf. Theory, vol. 65, no. 2, pp. 742–769, Feb. 2019.

[18] S. S. Du and J. D. Lee, “On the power of over-parametrization in neural networks with quadratic activation,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 1329–1338.

[19] M. Hardt and T. Ma, “Identity matters in deep learning,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–20.

[20] A. Kazempour, B. W. Larsen, and S. Druckmann, “Avoiding spurious local minima in deep quadratic networks,” 2020, arXiv:2001.00098.

[21] D. Soudry and Y. Carmon, “No bad local minima: Data independent training error guarantees for multilayer neural networks,” 2016, arXiv:1605.08361.

[22] T. Laurent and J. H. von Brecht, “The multilinear structure of ReLU networks,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 2908–2916.

[23] I. Safran and O. Shamir, “On the quality of the initial basin in overspecified neural networks,” in Proc. 33rd Int. Conf. Mach. Learn., 2016, pp. 774–782.

[24] D. Soudry and E. Hoffer, “Exponentially vanishing sub-optimal local minima in multilayer neural networks,” 2017, arXiv:1702.05777.

[25] M. Nguyen, M. C. Makkamala, and M. Heun, “On the loss landscape of a class of deep neural networks with no bad local valleys,” in Proc. Int. Conf. Learn. Represent., 2019.

[26] S. Liang, R. Sun, Y. Li, and R. Srikant, “Understanding the loss surface of neural networks for binary classification,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1–20.

[27] S. Liang, R. Sun, J. D. Lee, and R. Srikant, “Adding one neuron can eliminate all bad local minima,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 1–11.

[28] K. Kawaguchi and L. Pack Kaelbling, “Elimination of all bad local minima in deep learning,” 2019, arXiv:1901.00279.

[29] R. Ge, J. D. Lee, and T. Ma, “Learning one-hidden-layer neural networks with landscape design,” 2017, arXiv:1711.00501.

[30] W. Gao, A. Vardhan Makkuka, S. Oh, and P. Viswanath, “Learning one-hidden-layer neural networks under general input distributions,” 2018, arXiv:1810.04133.

[31] S. Feizi, H. Javadi, J. Zhang, and D. Tse, “Percupine neural networks: Almost all local optima are global,” 2017, arXiv:1710.02196.

[32] L. Chen, L. Jin, X. Du, S. Li, and M. Liu, “Deforming the loss surface to affect the behaviour of the optimizer,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 1–11.

[33] T. Milne, “Piecewise strong convexity of neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 1–10.

[34] J. Pennington and T. M. Darrell, “Geometry of network loss surfaces via random matrix theory,” in Proc. 34th Int. Conf. Mach. Learn., 2017, pp. 2798–2806.

[35] S. Mei, Y. Bai, and A. Montanari, “The landscape of empirical risk for non-convex losses,” 2016, arXiv:1607.06534.

[36] T. Scholten, K. Irpul, and E. Bengio, “Empirical risk landscape analysis for understanding deep neural networks,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–60.

[37] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, “Local minima in training of neural networks,” 2016, arXiv:1611.06310.

[38] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, “Local minima in training of neural networks,” 2016, arXiv:1611.06310.

[39] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, “Local minima in training of neural networks,” 2016, arXiv:1611.06310.

[40] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, “Local minima in training of neural networks,” 2016, arXiv:1611.06310.

[41] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, “Local minima in training of neural networks,” 2016, arXiv:1611.06310.
[55] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.

[56] C. D. Freeman and J. Bruna, “Topology and geometry of half-rectified network optimization,” in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–22.

[57] J. I. Goodfellow, O. Vinyals, and A. M. Saxe, “Qualitatively characterizing neural network optimization problems,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–20.

[58] Q. Liao and T. Poggio, “Theory II: Landscape of the empirical risk in deep learning,” 2017, arXiv:1703.09833.

[59] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the loss landscape of neural nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–11.

[60] S. Fort and S. Jastrzebski, “Large scale structure of neural network loss landscapes,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–9.

[61] C. H. A. Mehmeti-Göpel, D. Hartmann, and M. Wand, “Ringing ReLUs: Harmonic distortion analysis of nonlinear feedforward networks,” in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–22.

[62] F. Draxler, K. Veschinini, M. Salmhofer, and F. A. Hamprecht, “Essentially no barriers in neural network energy landscapes,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1309–1318.

[63] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, “Loss surfaces, mode connectivity, and fast ensembling of DNNs,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–10.

[64] I. Anokhin and D. Varotsos, “Low-loss connection of weight vectors: Distribution-based approaches,” in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 335–344.

[65] Y. Tian, “An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3404–3413.

[66] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, “Recovery guarantees for one-hidden-layer neural networks,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 4410–4419.

[67] Y. Li and Y. Yuan, “Convergence analysis of two-layer neural networks with ReLU activation,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[68] Y. Zhou, J. Yang, H. Zhang, Y. Liang, and V. Tarokh, “SGD converges to global minimum in deep learning via star-convex path,” in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–14.

[69] X. Zhang, Y. Yu, L. Wang, and Q. Gu, “Learning one-hidden-layer ReLU networks via gradient descent,” 2018, arXiv:1806.07808.

[70] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, “Gradient descent finds global minima of deep neural networks,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1675–1685.

[71] Z. Allen-Zhu, Y. Li, and Z. Song, “A convergence theory for deep learning via over-parameterization,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1–11.

[72] D. Zou, Y. Cao, D. Zhou, and Q. Gu, “Gradient descent optimizes over-parameterized deep ReLU networks,” *Mach. Learn.*, vol. 109, pp. 467–492, Oct. 2020.

[73] Q. Nguyen, “On the proof of global convergence of gradient descent for deep ReLU networks with linear widths,” in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 8056–8062.

[74] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–10.