Mathematical Foundations of AIM

3

Yonina C. Eldar, Yuelong Li, and Jong Chul Ye

Contents

Abstract

With the tremendous success of deep learning in recent years, the field of medical imaging has undergone unprecedented changes. Despite the great success of deep learning in medical imaging, these recent developments are largely empirical. Our goal in this chapter is to provide an overview of some of the key mathematical foundations of deep learning to the medical imaging community. In particular, we will consider ties with traditional machine learning methods, unrolling techniques which connect deep learning to iterative algorithms, and

Y. C. Eldar (✉)
Department of Math and Computer Science, Weizmann Institute of Science, Rehovot, Israel
e-mail: yonina.eldar@weizmann.ac.il

Y. Li
Amazon, San Jose, CA, USA

J. C. Ye
Department Bio and Brain Engineering & Department Mathematical Sciences, Korea Advanced Institute of Science & Technology (KAIST), Daejeon, Republic of Korea
e-mail: jong.ye@kaist.ac.kr

N. Lidströmer, H. Ashrafian (eds.), *Artificial Intelligence in Medicine*,
https://doi.org/10.1007/978-3-030-64573-1_333

geometric interpretations of modern deep networks.

Keywords

Machine learning · Deep learning · Neural network · Representation power · Hierarchical feature extraction

Introduction

Since groundbreaking performance improvements were first demonstrated by AlexNet [1] at the ImageNet challenge, deep learning has provided significant gains over classical approaches in various fields of artificial intelligence and data science. Availability of large-scale training datasets and advances in neural network research such as development of effective network architectures and efficient training algorithms have resulted in unprecedented successes of deep learning in innumerable medical imaging applications such as disease diagnosis, image segmentation, and image reconstruction.

In contrast to classical shallow machine learning approaches, which require "feature engineering" to extract features to feed into simple classifiers for diagnosis and recovery, one of the most important advantages of deep learning is that deep neural networks automatically discover the features and design appropriate classifiers and recovery methods in a data-driven way. This greatly simplifies the workflow of machine learning algorithm development and deployment. More importantly, the features are optimized toward specific tasks, which generally offer enhanced representation of the underlying data. In particular, for classification tasks the learned features can be much more discriminative than handcrafted features, whereas for reconstruction problems the learned features may better preserve the details and enable a more faithful reconstruction. As data representation plays a critical role, better features typically lead to superior performance in practice. Therefore, deep learning has become an increasingly important and versatile tool for medical imaging.

Nonetheless, the success of deep learning largely remains a mystery. From an architecture perspective, deep neural networks are typically composed of a series of convolution, pooling, and nonlinearity layers, which from a mathematical point of view are regarded as primitive tools. Interestingly, with abundant training samples available, the cascaded connection of these primitive tools results in superior performance over traditional approaches which are carefully designed and tailored toward specific applications.

A popular explanation for the success of deep neural networks is that neural networks are developed by mimicking the human brain. One of the most famous numerical experiments is the emergence of hierarchical features from a deep neural network when it is trained to classify human faces [2]. This phenomenon is similarly observed in human brains, where hierarchical features of the objects emerge during visual information processing. However, when asked why, it is surprising to find out that neuroscientists usually rely on numerical simulations with artificial neural networks to explain how hierarchical properties arise in the brain [3].

To understand this fundamental question, one can go back to classical approaches to understand the similarities and differences from modern deep neural network methods. Recent studies have shown that there is a close relationship between deep learning approaches and sparse representations [4–8]. Specifically, neural networks have been interpreted as unrolled versions of sparse recovery, where each unfolded block is learned from the training data [4, 9–11]. The authors in [5, 6] showed that a deep neural network can be interpreted as a piecewise linear representation, whose data-driven basis is learned from training data and automatically adapted to various input signals [12]. In this chapter, we will consider these and other mathematical underpinnings of deep networks in order to offer insights into their unprecedented performance.

We begin in section "Classical Machine Learning and Its Limitations" by reviewing typical machine learning models and explain their limitations in representation power which motivate the development of modern deep learning techniques. We then review standard modern deep neural networks and illustrate conceptually how they have successfully achieved superior representation power compared to classical approaches in section "Modern Deep Learning Revolution." We next

review algorithm unrolling in section "Algorithm Unrolling: From Iterative Algorithms to Deep Networks" which connects traditional iterative algorithms with modern deep neural networks. To gain further insights into deep learning, we discuss how it can be interpreted, both from a biological and a geometric perspective in section "Interpretations of Deep Learning." Finally, we summarize this chapter in section "Summary and Outlook."

Classical Machine Learning and Its Limitations

In this section, we provide historical context on how deep learning evolved into its current form. We review two relevant classical machine learning models, the perceptron and support vector machine, and illustrate the limitations of these "shallow" models.

The Perceptron Model

One of the earliest machine learning models is the single layer perceptron [13]. As illustrated in Fig. 1, it is built by fully connected neurons at a single hidden layer, where each neuron is formed by an affine transformation of the input vector followed by a nonlinear mapping. Formally speaking, let $\varphi : \mathbb{R} \mapsto \mathbb{R}$ be a nonlinear activation function, and let $\chi \subset \mathbb{R}^n$ denote the input space. Then, a single layer perceptron $f_\Theta : \chi \mapsto \mathbb{R}$ is represented by

$$f_\Theta(x) = \Sigma_{i=1}^{d} v_i \varphi\left(w_i^\top x + b_i\right), \quad x \in \chi \qquad (1)$$

where $w_i \in \mathbb{R}^n$ is a weight vector, $v_i, b_i \in \mathbb{R}$ are real constants, and $\Theta = \{(w_i, v_i, b_i)\}_{i=1}^{d}$ collectively represents the model parameters. To estimate the unknown parameters, a collection of training data $\{(x_i, y_i)\}_{i=1}^{N}$. The model parameters are then estimated by solving the following error criterion:

$$\min_{\Theta} \Sigma_{i=1}^{N} \ell\left(y_i, f_\Theta(x_i)\right) + \lambda R(\Theta) \qquad (2)$$

where $\ell(.,.)$ is a desired loss function, λ is a regularization parameter, and $R(\Theta)$ is a regularization function with respect to the parameter set Θ.

When the single layer perceptron was first introduced, it was not clear how to simultaneously optimize the weights $\{w_i\}_{i=1}^{d}$ for all neurons. Instead, a heuristic algorithm called *the perceptron algorithm* was used to estimate the neuron weights. When the training data set is linearly separable, the perceptron algorithm is guaranteed to find an exact separation in a finite number of steps. Later on, the back-propagation algorithm was introduced [14], which applies gradient-based learning to learn all neuron weights simultaneously.

One of the classical results regarding the representation power of a single layer perceptron is the universal approximation theorem [15], which states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets under mild assumptions on the activation function. The universal approximation theorem promoted research into neural networks as a powerful functional approximation; however, it turned out to be a limitation in the development of machine learning by circumventing construction of deeper neural networks. Although the theorem conceptually justifies that a shallow network with sufficiently many neurons can be a universal approximator, the proof of the theorem does not offer even a loose upper bound on how many neurons are required. Therefore, it is entirely possible that the networks which act as universal approximators are too large to be practical. However, at the time, deeper neural networks were difficult to train. Therefore, the prevalent approach was to increase the width of the network, namely the number of nodes, rather than the depth. Only recently has it been realized that depth matters, i.e., there exists a function that a deep neural network can approximate but a shallow neural network with the same number of parameters cannot [16–19].

Support Vector Machine

Another typical example of a "wide" model is Support Vector Machine (SVM). Similar to the neuron model, it classifies the data samples by learning an optimal separating hyperplane. However, in order to ensure that all data samples are far apart from the hyperplane, it employs a maximal

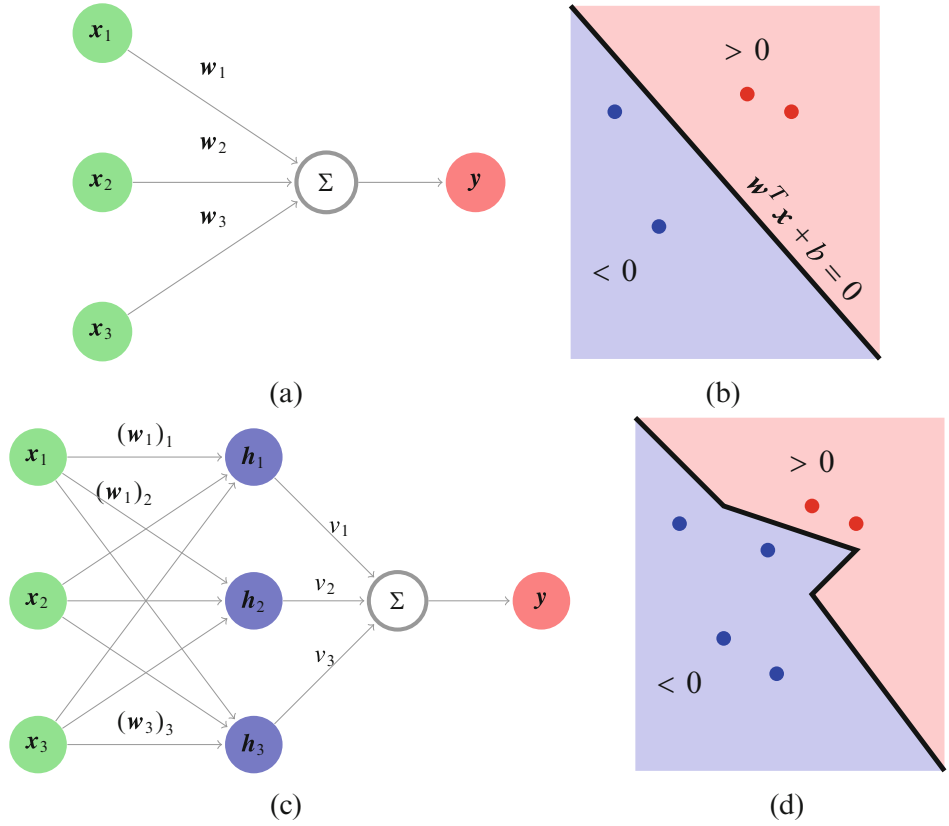

Fig. 1 Geometric interpretation of the perceptron model. (**a**) Architecture of the perceptron model for a three dimensional input. A neuron first applies an affine mapping of the input x and then performs thresholding to determine the predicted class y; (**b**) geometrically, a neuron divides the input space into two decision regions through a separating hyperplane; (**c**) the perceptron model is formed through a combination of multiple neurons; and hence (**d**) its decision boundary corresponds to a piecewise linear surface

margin loss function for training the weights of the separating hyperplane. Specifically, given training samples $\{x_n, y_n\}_{n=1}^N$, it solves the following minimax problem [20]:

$$\max_{w,b} \left\{ \frac{1}{\|w\|} \min_n \left[y_n \left(w^T x_n + b \right) \right] \right\} \tag{3}$$

which can be interpreted as maximizing the distances (margin) from the closest data samples (support vectors) to the separating hyperplane. A visual illustration of this criterion is given in Fig. 2a.

In practice, it is possible that the data samples cannot be classified by a single separating hyperplane, i.e., the data samples are not linearly separable. In this scenario, SVM employs a nonlinear mapping φ which embeds the data manifold into a high-dimensional feature space [21], where the data samples are assumed linearly separable. In practice, instead of explicitly designing the embedding φ, one can seek a *kernel function,* which characterizes the inner product $\langle ., . \rangle$ in the underlying feature space:

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle. \tag{4}$$

The aforementioned SVM technique is then employed in the feature space. In this way, the explicit form of ϕ does not need to be specified and only the kernel function has to be designed.

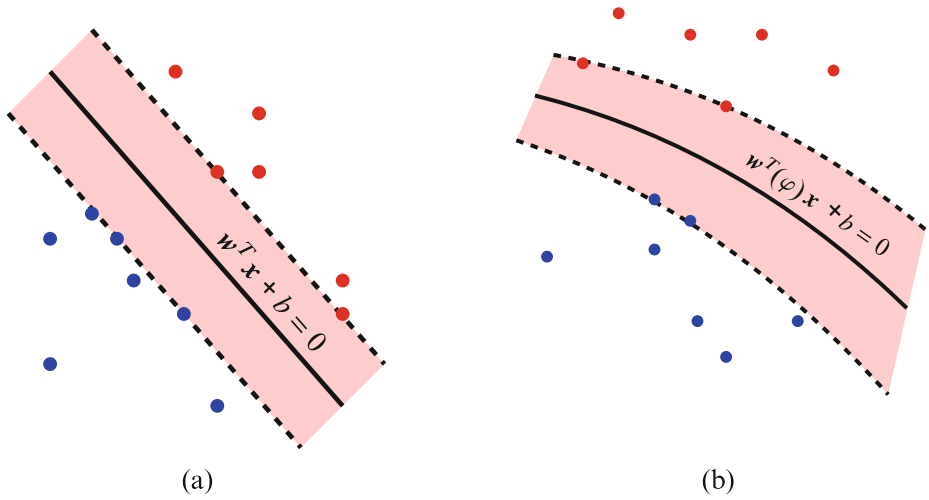

Fig. 2 The decision regions of SVM. (**a**) Linear SVM forms its separating hyperplane by maximizing the margins of data samples; (**b**) through the kernel trick, SVM is able to form nonlinear decision boundaries and classify samples that are not linearly separable

This technique is commonly called the *kernel trick* [21]. In order for the kernel function to correspond to a valid inner product, Mercer's theorem [22] requires φ that the kernel function is symmetric and positive definite. As depicted in Fig. 2b, the kernel trick enables SVM to form nonlinear decision boundaries. As the feature space is often of higher dimensionality compared to the ambient space, kernel SVM can be regarded as increasing the "width" of the classification model in order to enhance its capacity, instead of employing hierarchical architectures which follow the "depth" dimension.

In principle, any symmetric positive definite kernel function can be associated to an inner product. Therefore, the kernel generates a wide variation of functions within the feature space. Indeed, one of the main research thrusts in classical machine learning approaches is to find appropriate kernels that are suitable for specific applications. That said, kernel methods still have fundamental limitations. First, the kernel function is typically handcrafted instead of learned from data. Second, once the kernel machine is trained, the parameters are fixed, and it is not possible to adjust them at test phase. These drawbacks lead to fundamental limitations of expressivity of kernel-based learning models [21].

Modern Deep Learning Revolution

One of the main reasons for the success of deep learning is its significantly extended expressivity by learning hierarchical features through deep neural networks. Before we delve deep into this claim, we first provide a brief review of modern deep network architectures.

Architectures of Modern Deep Neural Networks

In early neural network research, the Multi-Layer Perceptrons (MLP) was a popular choice. A diagram illustration of this architecture is given in Fig. 3. As can be observed, this model can be regarded as the extension of a single layer perceptron shown in Fig. 1c, by introducing multiple hidden layers. This scheme can be viewed as adopting a hierarchical feature representation and increasing its "depth."

Similar to perceptron, each neuron in MLP is fully connected to every neuron in the previous layer, except for the input layer. The layers are thus commonly called *fully connected* layers. Analytically, in the l-th layer, the

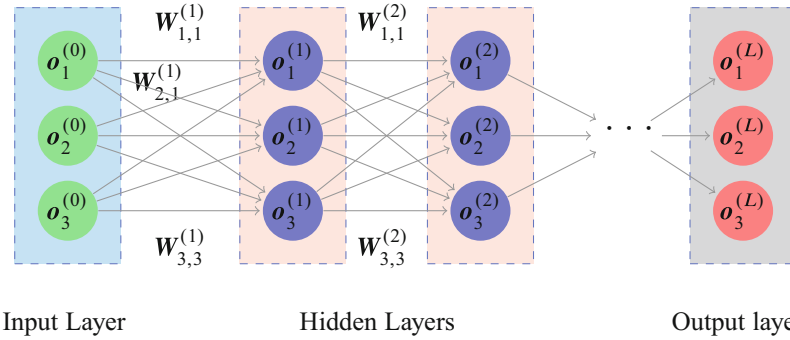

Fig. 3 Architecture of MLP. The input vector passes through a few hidden layers and reaches the output layer. Each layer comprises an affine transformation followed by a nonlinear activation function. We omit drawing the activation functions for brevity

relationship between the neurons $o_j^{(l)}$ and $o_i^{(l+1)}$ is expressed as

$$o_i^{(i+1)} = \sigma\left(\sum_j w_{ij}^{(l+1)} o_j^{(l)} + b_i^{(l+1)}\right) \tag{5}$$

where $W^{(l+1)}$ and $b^{(l+1)}$ are the *weights* and *biases*, respectively, and σ is a nonlinear *activation function*. Popular choices of activation functions include the logistic function and the hyperbolic tangent function. In recent years, they have been superseded by Rectified Linear Units (ReLU) [23] defined by

$$\mathrm{ReLU}(x) = \max\{x, 0\}. \tag{6}$$

The W s and b's are generally trainable parameters that are learned from datasets, using backpropagation [24] for gradient computation.

Nowadays, MLPs are rarely seen in practical medical applications. The fully connected nature of MLPs contributes to a rapid increase in their parameters, making them difficult to train. To address this limitation, Fukushima et al. [25] designed a neural network by mimicking the visual nervous system [26]. The neuron connections are restricted to local neighbors only, and weights are shared across different spatial locations. The affine transformations then become convolutions (or correlations in a strict sense), and the resulting networks are commonly called Convolutional Neural Networks (CNN). A visual illustration of a CNN can be seen in Fig. 4. With significantly reduced parameter dimensionality, training deeper networks becomes much easier.

While CNNs were first applied to digit recognition, their translation invariance is a desirable property for analyzing image features and are broadly applied in various medical problems, including medical image retrieval [27], segmentation [28], and reconstruction [4], to name a few. In reality, the architecture of CNN can be more sophisticated than illustrated in Fig. 4. There may be advanced types of convolutions, such as stridden convolutions which reduce spatial resolution [29], transposed convolutions which perform up-sampling [30], and more. There may also be other layers, such as pooling layers which perform spatial aggregation [29], batch normalization layers which stabilize training [31], and dropout layers which perform network assembling to reduce overfitting [32].

In some medical applications, the data may exhibit certain sequential forms. A concrete example is video processing, where video frames have temporal dependencies [33]. In such scenarios, Recurrent Neural Networks (RNN) [14] are a popular choice. RNNs explicitly model the sequential data dependence in different time steps in the sequence and scale well to sequences with varying lengths. A visual depiction of RNNs is provided in Fig. 5. Given the previous hidden state $s^{(l-1)}$ and input variable $x^{(l)}$, the next hidden state $s^{(l)}$ is computed as

$$s^{(l)} = \sigma_1\left(Ws^{(i-1)} + Ux^{(l)} + b\right) \tag{7}$$

while the output variable $o^{(l)}$ is generated by

$$o^{(i)} = \sigma^2\left(Vs^{(l)} + b\right)$$

Here, ***U, V, W, and b*** are trainable network parameters and σ_1, and σ_2 are activation

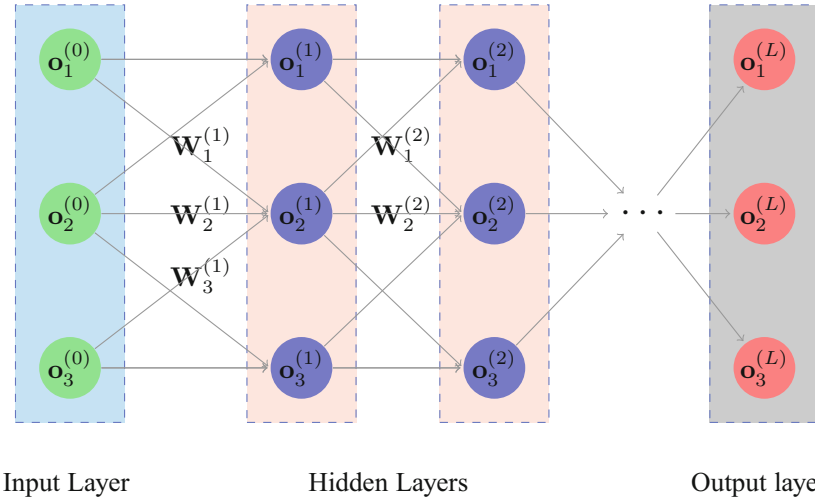

Fig. 4 Architecture of CNN. Instead of connecting all neurons in adjacent layers, CNN only connects each neuron to its spatial neighbors in the previous layer. Furthermore, the weights are shared across different spatial locations. The output neurons in each layer can then be generated by convolving the input neurons with weights, followed by a nonlinear activation function. We omit drawing the activation functions for brevity

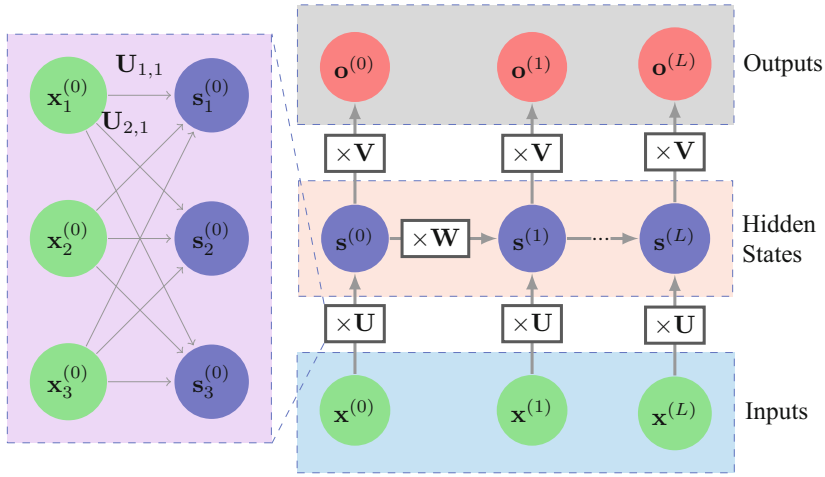

Fig. 5 Architecture of RNN. The inputs $\{x^{(l)}\}_1$ are fed into the network in a sequential order. The state variables $\{s^{(l)}\}_1$ evolve according to a linear transition model, thus capturing sequential dependencies. The outputs $\{o^{(l)}\}_1$ are generated by combining the state variables and the inputs. Parameters U, V, and W are shared across different time steps

functions. We again omit the activation functions and biases in Fig. 5. In contrast to MLPs and CNNs where the layer operations are applied recursively in a hierarchical representation fashion, RNNs apply the recursive operations as the time step evolves. A distinctive property of RNNs is that the parameters ***U, V, and W*** are shared across all time steps, rather than varying from layer to layer. Training RNNs can thus be difficult as the gradients of the parameters may either explode or vanish [34].

In practice, the state-space relation (7) suffers from a few limitations: It does not favor long-term dependencies which is crucial for modeling long sequences, and it brings about difficulties in training by introducing the gradient vanishing and exploding phenomena. To address these issues,

more advanced architectures have been suggested. Long-Short-Term-Memory [35] and Gated Recurrent Network [36] employ gating units and allow information to flow freely both in the forward pass and during back-propagation, which effectively mitigates the aforementioned limitations of vanilla RNN. Recently, self-attention mechanisms have become popular as an effective structure to reduce computational complexity and enable parallel computations, and better capture long-term dependencies [37]. However, the basic prototype remains the same: The input sequence is first encoded into state vectors, and then decoded into an output sequence. The decoder models the temporal dependencies between the current and previous timestamps, which can be regarded as an auto-regressive model.

Representation Power of Deep Neural Networks

In order to understand the superior representation power of modern deep neural networks over classical machine learning models, we will explore the decision regions of MLP by analyzing its per-layer mapping (5). Without loss of generality, we omit the bias term. For the nonlinear activation function σ, we adopt the ReLU function defined in (6), and let

$$o^{(l)} = \sigma\left(g^{(l)}\right) \ , \ \mathrm{g}^{(l)} \coloneqq W^{(l)} o^{(l-1)}. \tag{8}$$

For an L-layer MLP, the neural network output for a given input x can be represented by

$$f_\Theta(x) \coloneqq \left(\sigma \circ g^{(L)} \circ \sigma \circ g^{(L-1)} \cdots \circ g^{(1)}\right)(x) = B(x)^\top x \tag{9}$$

where $0 = [\mathrm{W}^{(1)} \cdots \mathrm{W}(^{\mathrm{L}})]$ and

$$B(x) = W^{(1)} \Lambda^{(1)}(x) W^{(2)} \Lambda^{(2)}(x) \cdots \Lambda^{(L-1)}(x) W^{(L)} \tag{10}$$

Here, $\Lambda^{(l)}(x)$ is a diagonal matrix with 0 and 1 elements indicating the ReLU activation patterns.

Note that the matrix $B(x)$ in (10) depends on the ReLU activation patterns $\Lambda^{(l)}, l = 1, \ldots, L - 1$, which are determined by the input x. In fact, this ReLU activation-dependent diagonal matrix plays a key role in enabling inductive learning and emergence of hierarchical features. Specifically, the nonlinearity is applied after the linear operation, so the on-and-off activation pattern of each ReLU determines a binary partition of the feature space at each layer across the hyperplane that is determined by the weight matrix. Accordingly, in deep neural networks, the input space is partitioned into multiple nonoverlapping regions so that input images for each region share the same linear representation, but not across the partition. This implies that two different input images are automatically switched to two distinct linear representations that are different from each other depending on the partition as shown in Fig. 6 [38].

This leads to an important insight: Deep neural networks search for the distinct linear representation for each input. However, in contrast to classical optimization-based approaches, deep neural networks do not solve the optimization problem for a new input, rather they switch to different linear representations by changing the ReLU activation patterns. This is an important advance over the classical signal-processing approach.

To quantify the representation power of modern deep networks such as CNN, researchers are conducting rigorous theoretical analysis to extend the classical universal approximation theorem. Zhou et al. [39] prove that deep CNNs can approximate continuous functions supported on a compact space of any accuracy as long as its depth is sufficiently high. They also analyze the approximation property of deep CNNs over a family of functions in Sobolev space, which, loosely speaking, comprises smooth functions that are differentiable to a certain order. Similarly, Yarotsky et al. [19] analyze the approximation property of deep neural networks with ReLU activation functions on the Sobolev space. In addition, they provide upper and lower bounds of network complexity, i.e., the number of networks layers, weights, and neurons required to reach a given approximation error.

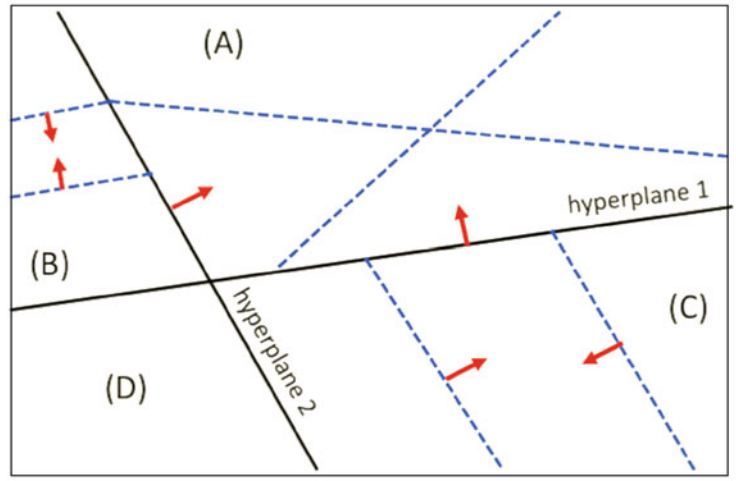

Fig. 6 Piecewise linear representation by a deep neural network

Other Properties of Deep Neural Networks

Splines, or piecewise polynomials, are another form of function approximators. It is therefore interesting to see how splines are connected to deep neural networks. Unser et al. [12] establish a conceptual connection between deep neural networks and piecewise linear functions, i.e., first-order uniform splines. Suppose we are given an MLP of the following form:

$$f\left(x;\left\{W^{(l)},\sigma^{(l)}\right\}_l\right) := \left(\sigma^{(L)}\circ g^{(L)}\circ\sigma^{(L-1)}\circ g^{(L-1)}\cdots\circ g^{(1)}\right)(x)$$

where $g^{(l)} := W^{(l)}o^{(l-1)}$ and $\sigma^{(l)}$'s are arbitrary nonlinear activation functions.

Given a collection of training samples $\{(\mathbf{x_m}, \boldsymbol{y_m})\}_m$, we train the network $\boldsymbol{f}$ through the following optimization problem:

$$\min_{\left\{W^{(l)},\sigma^{(l)}\right\}} \Sigma_{m=1}^{M}\ell\left(y_m, f\left(x_m;\left\{W^{(l)},\sigma^{(l)}\right\}_l\right)\right) + \lambda\mathcal{R}\left(\left\{W^{(l)}\right\}_l\right) + \mu\mathcal{T}\left(\left\{\sigma^{(l)}\right\}_l\right) s\mathcal{T} \qquad (11)$$

where ℓ is a general convex loss function, $\mathcal{R}$ is some arbitrary convex regularization function over the network weights, $\mathcal{T}$ is a regularization over $\sigma^{(l)}$, s which promotes sparsity of their derivatives, and λ and μ are positive regularization parameters. The authors prove that, in order for f to be an optimal solution to the problem (11), it must be the case that $\sigma^{(l)}$'s are piecewise linear. In other words, the original infinite-dimensional optimization becomes finite-dimensional because $\sigma^{(l)}$ now adopts a parametric spline form.

Another intriguing property of deep neural networks is their amazing generalization capability, which seems mysterious from the perspective of classic machine learning. In particular, the number of trainable parameters in deep neural networks is often greater than the training data set, this situation being notorious for overfitting from the point of view of classical statistical learning theory. However, empirical results have shown that a deep neural network generalizes well in the testing phase, resulting in high performance for the unseen data.

This apparent contradiction has raised questions about the mathematical foundations of machine learning and their relevance to practitioners. Recently, the authors in [40, 41] have suggested how to reconcile classical understanding and modern practice in a unified framework. In classical machine learning theory, models with exceedingly high capacity are subject to overfitting and exhibit high test errors due to the fundamental bias-variance trade-off. However, the authors argue that, once the model capacity increases beyond a certain point called interpolation point, its performance starts improving in the test phase. Increasing the functional class capacity to the overparameterized area thus improves the generalization performance of the resulting

classifiers. The authors further justify their claim empirically through experiments.

Algorithm Unrolling: From Iterative Algorithms to Deep Networks

Parallel to the revolution of machine learning models, another thread of research constructs deep networks from iterative algorithms by mapping each iteration to a network layer. This line of research centers around an important technique called *algorithm unrolling,* which originated from Gregor et al.'s seminal technique on learnable sparse coding, called Learned Iterative Shrinkage and Thresholding Algorithm (LISTA) [7]. We first review this method and related theoretical findings. We then discuss general formulations of algorithm unrolling and provide practical examples.

Learned Iterative Shrinkage and Thresholding Algorithm

LISTA aims to approximately solve the *sparse coding* problem at a higher efficiency than iterative methods, by learning an unknown dictionary from real data. Specifically, given an observation vector $y \in \mathbb{R}^m$, we seek a vector $x \in \mathbb{R}^n$ such that $\mathbf{y} \approx \boldsymbol{Wx}$ and encourage as many coefficients in $\boldsymbol{x}$ to be zero (or small in magnitude) as possible [42]. A typical approach to achieve this is by solving an unconstrained convex minimization problem:

$$\min_{x \in \mathbb{R}^m} \frac{1}{2}\|y - Wx\|_2^2 + \lambda||\mathrm{x}||1, \qquad (12)$$

where $\boldsymbol{\lambda > 0}$ is a regularization parameter that controls the sparsity of the solution. A well-known class of methods for solving (12) are proximal methods such as Iterative Shrinkage and Thresholding Algorithm (ISTA) [43], which perform the following iterations:

$$x^{l=1=s_\lambda}\left\{\left(I - \frac{1}{\mu}W^T W\right)x^1 + \frac{1}{\mu}W^T y\right\}, 1 = 0, 1, \cdots \qquad (13)$$

Here, $\mathbf{I} \in \mathbb{R}^{\boldsymbol{nxn}}$ *is* the identity matrix, μ is a positive parameter that controls the iteration step size, $\boldsymbol{S\lambda(\cdot)}$ is the soft-thresholding operator defined as for a scalar *x,* and *Sλ(−)* operates element-wise on vectors and matrices.

$$S\lambda(x) = \mathrm{sign(x)}\ \max\{|\mathrm{x}| - \lambda, 0\}, \qquad (14)$$

The slow convergence rate of ISTA can be problematic in real-time applications. Furthermore, the matrix W may not be known exactly. In their seminal work, Gregor and Lecun [7] propose a highly efficient learning-based method that computes good approximations of optimal sparse codes in a fixed amount of time, with the help of W learned optimally from real data [7]. Specifically, iteration (13) can be recast into a single network layer as depicted in Fig. 7. This layer comprises a series of analytical operations (matrix-vector multiplication, summation, and soft-thresholding), which is of the same nature as a neural network. A diagram representation of one iteration step reveals its resemblance to a single network layer. Executing ISTA *L* times can be interpreted as cascading *L* such layers, which essentially forms an L-layer deep network. Note that, in the unrolled network, an implicit substitution of parameters has been made: $W_t = I - \frac{1}{\mu}W^{\mathrm{T}}W$ and $W_e = \frac{1}{\mu}W^T$.

After unrolling ISTA into a network, named Learned ISTA (LISTA), the network is trained through back-propagation using real datasets to optimize the parameters W_t, $\mathrm{W_e}$, and λ. Training is performed in a supervised manner, meaning that for every input vector $y^t \in \mathbb{R}$, t = 1, . . , T, its corresponding sparse output $x^{*t} \in \mathbb{R}^n$, t = 1, . . , T is known. The sparse codes $x^{*\mathrm{t}}$ can be determined, for example, by executing ISTA when W is known. Feeding vector y^t into the network results in a predicted output$\widehat{x}^t(y^t; W_t, W_e, \lambda)$. The network-training loss function is formed by comparing the prediction with the known sparse output $x^{*\mathrm{t}}$:

$$\ell(W_t, W_e, \lambda) = \frac{1}{T}\Sigma_{t=1}^T \left\|\widehat{x}^t(y^t; W_t, W_{e,}\lambda) - x^{*t}\right\|_2^2 \qquad (15)$$

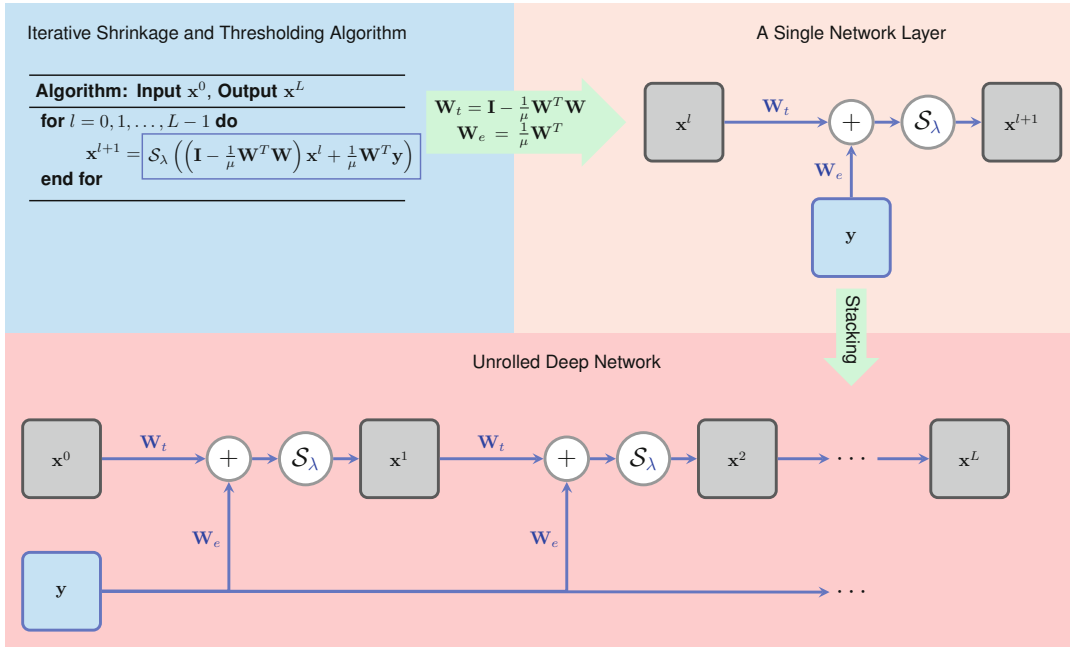

Fig. 7 Illustration of LISTA: One iteration of ISTA executes a linear and then a nonlinear operation and thus can be recast into a network layer; by stacking the layers, a deep network is formed. The network is subsequently trained using paired inputs and outputs by back-propagation to optimize the parameters $\boldsymbol{W_e}$, $\boldsymbol{W_t}$, and $\boldsymbol{\lambda}$. The trainable parameters in the network are colored in blue

The network is trained through loss minimization, using popular gradient-based learning techniques such as stochastic gradient descent, to learn the unknown parameters W_t, W_e, and λ [24]. With optimized parameters, LISTA may achieve higher efficiency compared to ISTA. Indeed, it has been shown empirically that the number of layers L in (trained) LISTA can be an order of magnitude smaller than the number of iterations required for ISTA to achieve convergence corresponding to a new observed input, thus dramatically boosting the sparse coding efficiency [7]. Furthermore, when the dictionary $\boldsymbol{W_t}$ **and** $\mathbf{W_e}$ are unknown or hard to determine analytically, they can be learned from real datasets. In practice, $\boldsymbol{W_t}$, $\boldsymbol{W_e}$, and $\boldsymbol{\lambda}$ can be untied and vary in each layer.

In addition to empirically observed superior efficiency, researchers have recently confirmed the faster convergence speed of LISTA over ISTA by conducting rigorous theoretical analysis. In particular, recent studies reveal the convergence rate of LISTA, and relevant techniques, and characterize the optimality conditions. For instance, Xin et al. [44] study the unrolled Iterative Hard Thresholding (IHT) algorithm, which has been widely applied in various sparsity-constrained estimation problems. IHT largely resembles ISTA except that an ℓ^0 norm is employed instead of the ℓ^1 norm. The authors prove that, in order for the unrolled IHT network to recover a maximally sparse solution (i.e., a vector with minimal ℓ° norm), a weight coupling scheme must be satisfied; furthermore, under the weight-coupling constraint and certain additional Restricted Isometry Property (RIP) conditions [11], a linear convergence rate can be deduced. Compared to classical IHT, the learned version poses a much milder requirement on the RIP condition, meaning that the unrolled network is capable of recovering sparse signals from a much broader family of dictionaries.

Chen et al. [45] observe similar behaviors of the LISTA network with layer-specific parameters. They prove that, in order for LISTA to

recover the underlying sparse solutions, a similar weight-coupling constraint must be satisfied asymptotically. They further introduce a so-called support-selection scheme, which, together with weight coupling and a few other mild conditions, ensures linear convergence rate of the unrolled network. As a follow-up, Liu et al. [46] introduce certain mutual coherence conditions and analytically characterize optimal network parameters based on those conditions. Similar to the networks with trained weights, networks adopting analytic weights converge at a linear rate, which implies that analytic weights can be as efficient as learned weights. In addition, analytic weights are of much lower dimensionality compared to trained weights. However, determining the analytic weights can be a nontrivial task as they are solutions to another dedicated optimization problem.

Unrolling Generic Iterative Algorithms

Although the initial focus of Gregor et al.'*s* work [7] was on sparse coding techniques, the underlying principles could be easily generalized. More specifically, provided with a certain iterative algorithm, we can unroll it into a corresponding deep network, following the procedures depicted in Fig. 8 [8]. The first step is to identify the analytic operations per iteration, which we represent abstractly as an h function, and the associated parameters, which we denote collectively as θ^l. The next task is to generalize the functional form of h into a more generic version $\widehat{h}$, and correspondingly expand the parameters θ^l into an enlarged version $\widehat{\theta}^l$ if necessary. For instance, in LISTA, the parameter W is substituted with W_t and W_e through the formula $W_t = I - \frac{1}{\mu}W^T W$ and $W_e = \frac{1}{\mu}W^T$. After this procedure, each iteration can be recast into a network layer in the same spirit as LISTA. By stacking the mapped layers together, we obtain a deep network with undetermined parameters and then obtain optimal parameters through end-to-end training using real-world datasets.

The exact approach to generalize h and $\theta^l s$ toward $\widehat{h}$ and $\widehat{\theta}^l$ s is largely case specific. An extreme scenario is to strictly follow the original functional forms and parameters, i.e., to take $\widehat{h} = h$ and $\widehat{\theta}^l = \theta^l, \forall l$. In this way, the trained network corresponds exactly to the original algorithm with finite truncation and optimal parameters. In addition to efficiency enhancement thanks to training [7], the unrolled networks can aid with estimating structured parameters such as filters [47] or dictionaries [48] which are hard to design either analytically or by handcrafting. Alternatively, some operations may be replaced with a stand-

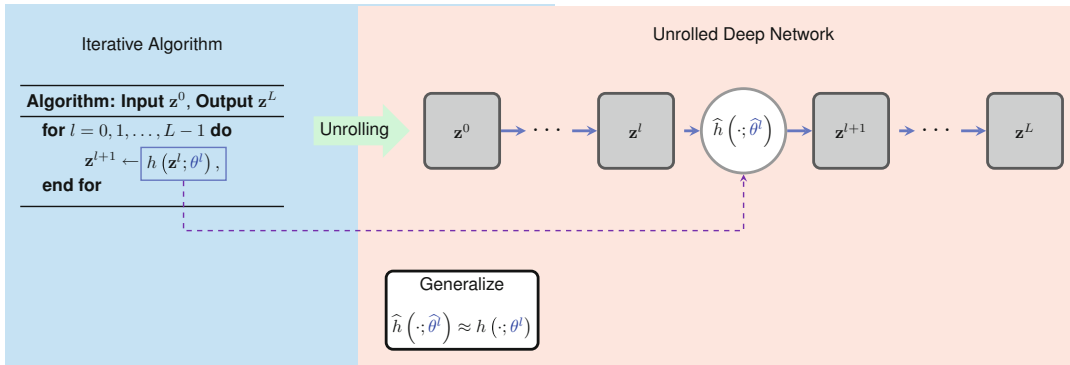

Fig. 8 Illustration of the general idea of algorithm unrolling: given an iterative algorithm, we map one iteration (described as the function h parametrized by θ^l, $l = \theta^l, \ldots, L-1$), into a single network layer, and stack a finite number of layers to form a deep network. Feeding the data forward through an L-layer network is equivalent to executing the iteration L times (finite truncation). The parameters $\theta^l, l = 0, \ldots, L-1$ are learned from real datasets by training the network end-to-end to optimize the performance. They can either be shared across different layers or varying from layer to layer. The trainable parameters are colored in blue

alone deep neural network such as Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN). For instance, in [49], the authors replace a proximal gradient update step with a CNN. In addition, the parameters can be layer specific instead of being shared across different layers. For instance, in [50], the authors plug in a CNN in each iteration step (layer) and allow the network parameters to differ. As it is, networks with shared parameters generally resemble RNN, while those with layer-specific parameters mimic CNN, especially when there are convolutional structures embedded in layer-wise operations. While custom modifications may potentially invalidate the convergence guarantees, they are practically beneficial and critical for performance improvement because the representation capacity of the network can be significantly extended.

In addition to performance and efficiency benefits, unrolled networks can potentially reduce the number of parameters and hence storage footprints. Conventional generic neural networks typically reuse essentially the same architectures across different domains and thus require a large amount of parameters to ensure their representation power. In contrast, unrolled networks generally carry significantly fewer parameters, as they implicitly transfer problem structures (domain knowledge) from iterative algorithms to unrolled networks, and their structures are more specifically tailored toward target applications. These benefits not only ensure higher efficiency, but also provide better generalizability especially under limited training schemes.

Unrolling techniques have been widely used in medical applications. An important imaging modality is ultrasound, which has the advantage of being a radiation-free approach. When used for blood flow depiction, one of the challenges is the fact that the tissue reflections tend to be much stronger than those of the blood, leading to strong clutter resulting from the tissue. Thus, an important task is to separate the tissue from the blood. Various filtering methods have been used in this context such as high-pass filtering, and filtering based on the singular value decomposition. Solomon et al. [51] suggest using a robust Principal Component Analysis (PCA) approach by modeling the received ultrasound movie as a low-rank and sparse matrix where the tissue is low rank and the blood vessels are sparse. The robust PCA problem can be solved via a generalized version of ISTA, which is further unrolled into a deep network, called Convolutional rObust pRincipal cOmpoNent Analysis (CORONA). As the name suggests, they replace matrix multiplications with convolutional layers, effectively converting the network into a CNN-like architecture. Compared with state-of-the-art approaches, CORONA demonstrates vastly improved reconstruction quality and has much fewer parameters than the well-known ResNet [52].

In optical microscopy, a fundamental challenge is to enhance spatial resolution, which is limited by the physics of light. Solomon et al. [47] exploit the sparse nature of the fluorophores distribution and improve the spatial resolution via a sparsity-constrained estimation approach, called SPARsity based super-resolution COrrelation Microscopy (SPARCOM). Recently, Dardikman et al. [53] unroll SPARCOM into a deep network called Learned SPARCOM (LSPARCOM). The structure of the network resembles LISTA, except that they adopt a customized proximal operator. Experimental results show that LSPARCOM can obtain super-resolution images from a small number of high-emitter density frames without knowledge of the optical system, and has clear runtime advantages.

Interpretations of Deep Learning

We have so far covered the origin of deep learning and how it connects to classical machine learning models and traditional iterative algorithms. In this section, we investigate other ways to interpret deep neural networks, in addition to a methodological perspective. Specifically, we first take a biological perspective, where we illustrate how hierarchical features, an essential component of deep neural networks, are also commonly found in biological visual systems. We then switch to a geometric standpoint and contend that deep neural networks effectively capture the low-dimensional data manifolds.

Hierarchical Features in the Visual System

The visual system is a part of the central nervous system that enables organisms to detect and interpret information from visible light to create a representation of the environment. In pursuit of understanding the visual system, Hubel and Wiesel [54] found two classes of functional cells in the primary visual cortex: simple cells and complex cells. More specifically, simple cells at the primary visual cortex at the V1 L4 layer respond best to edge-like stimuli with a certain orientation, position, and phase within their relatively small receptive fields. They realized that such response of the simple cells could be obtained by pooling the activity of a small set of input cells with the same receptive field that is observed in Lateral Geniculate Nucleus (LGN) cells. This observation has been extended to higher areas of the visual cortex to result in a class of object recognition models [3]. Specifically, there is a neuronal connection along this path, which forms a neuronal hierarchy such that neurons become sensitive to more complex inputs. An extreme form or surprising example of this information-processing hierarchy can be found in the discovery of the so-called "Jennifer Aniston Cell" [55], which identified a single neuron that is sensitive to a complex but specific concept or object.

A similar phenomenon can be observed in the convolution neural network, once it is properly trained. In particular, VGGNet [2] provides very intuitive information that is well correlated with the visual information processing in the brain. For example, Fig. 9 illustrates the input signal that maximizes the filter response at specific channels and layers of VGGNet [2]. Here, an input image

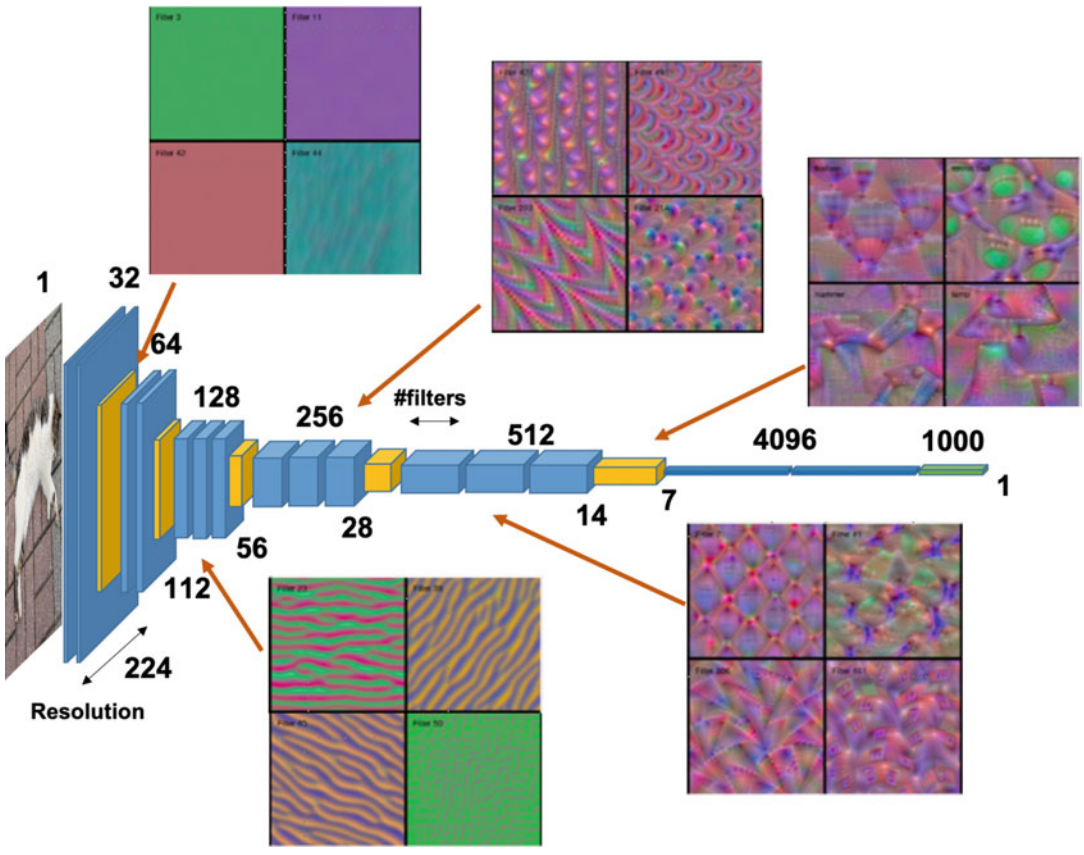

Fig. 9 Input images that maximize filter responses at specific channels and layers of VGGNet

that activates this filter most is displayed for specific channel and layer filters. This is similar to the Hubel and Wiesel experiments where they analyzed the input image that maximizes the neuronal activation. Specifically, at the earlier layers the input signal-maximizing filter response is composed of directional edges similar to the Hubel and Wiesel experiment. As we go deeper into the network, the filters build on each other and learn to code more complex patterns. Finally, in Fig. 9 the input images that maximize the response on the last softmax level in the specific classes correspond to the visualization of the input images that maximize the class categories. The emergence of the hierarchical feature from simple edges to the high-level concept is similar to visual information processing in the brain.

Geometric Understanding of Deep Neural Networks

The manifold structure of real-world data has been heavily exploited in classical machine learning techniques. Structured data are often assumed to lie on a manifold whose dimensionality is much lower than its ambient space. In particular, it has been long recognized that natural images lie on a low-dimensional manifold [34], and the key to success in many machine learning tasks hinges on capturing the underlying manifold structure. As illustrated in Fig. 10, researchers have spent intensive efforts to model the bidirectional mapping between the data manifold and the underlying latent space. The forward mapping φ_α, commonly called the *encoder*, maps each data sample (such as an image) into a low-dimensional latent vector, while the inverse mapping $\varphi_{\alpha-1}$, commonly called *decoder*, generates a sample (such as an image) from a provided latent vector.

In classical machine learning, modeling the encoder and decoder has been a key topic in unsupervised learning. For instance, Principal Component Analysis (PCA), as a widely applied linear dimensionality reduction technique, estimates a linear encoder which projects the data onto a low-dimensional subspace, and a linear decoder which recovers the data approximately from this subspace. However, the subspace usually does not offer an accurate approximation when the data manifold is nonlinear. To improve the approximation accuracy, nonlinear dimensionality techniques have been developed by either modeling the topological relationship among inputs [56], or capturing their metric structure [57]. Nonetheless, these techniques lack a deep hierarchical decomposition of the mapping, φ_α, and hence do not faithfully model the data manifold.

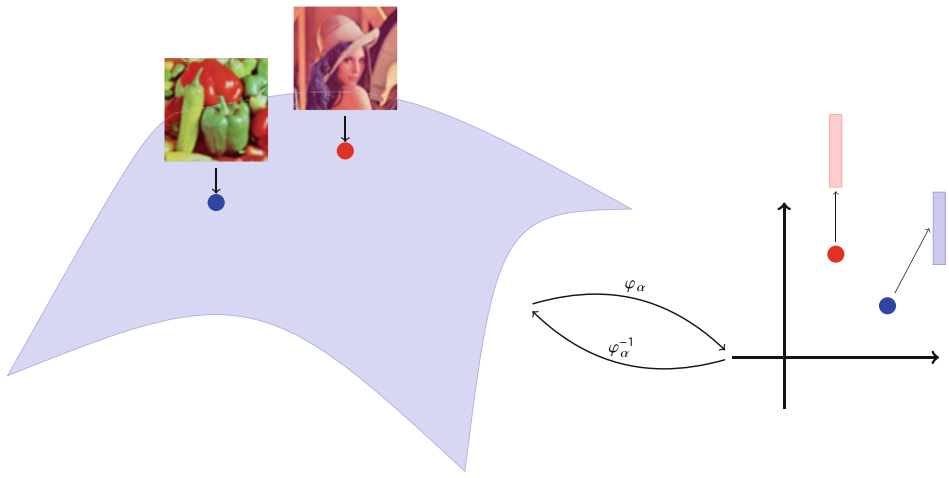

Fig. 10 Manifold structure of natural images. Many unsupervised learning techniques aim to discover the encoding mapping φ_α from the low-dimensional manifold of natural images to its latent space. Generative models capture the inverse mapping, which generates natural image samples given the latent vectors

In recent years, deep unsupervised learning has achieved tremendous progress. The essence of many such techniques is how to learn the encoder/decoder using deep neural networks. Variational Auto Encoders (VAE) [58] jointly learns both the encoding mapping φ_α and the decoding mapping $\varphi_{\alpha-1}$ by integrating auto-encoder with variational Bayes. There are also generative models which effectively learn the mapping $\varphi_{\alpha-1}$ and generate high-quality data samples providing the latent vectors. Typical generative models include Generative Adversarial Networks (GAN) [59] and Normalizing Flows (NF) [60]. The deep hierarchical architecture ensures high capacity in modeling complicated functional mappings, which is the key to success for these approaches.

Summary and Outlook

In this chapter, we reviewed the historical developments of deep learning, following two main threads: the evolution from classical machine learning models to modern deep learning models, and the transition from traditional iterative algorithms to contemporary deep networks. We explained the limitations of traditional machine learning models and algorithms, in terms of expressivity, and discussed how deep learning successfully can overcome these limitations. We also reviewed recent theoretical breakthroughs which justify the superior representation power of deep networks and help in understanding their properties and behaviors.

Although there is already a rich body of research on the mathematical foundation of deep learning, we are still far from understanding the full mystery of deep learning. As it is, so far there is little knowledge on what weights are optimal in order for the networks to approximate a certain function. In addition, currently network training is largely empirical, and network performance largely relies on heuristics in training and hyperparameter tuning. In the context of medical imaging, developing more methods that are interpretable and robust is of key importance. With the many recent exciting advances and the many researchers entering this field, the next decade will surely lead to further insights and methods in these directions.

References

1. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inform Process Syst. 2012;25:1097–1105.
2. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations, 2015.
3. Riesenhuber M, Poggio T. Hierarchical models of object recognition in cortex. Nat Neurosci. 1999;2(11):1019–25.
4. Jin KH, McCann MT, Froustey E, Unser M. Deep convolutional neural network for inverse problems in imaging. IEEE Trans Image Process. 2017;26(9):4509–22.
5. Ye JC, Han Y, Cha E. Deep convolutional framelets: a general deep learning framework for inverse problems. SIAM J Imag Sci. 2018;11(2):991–1048.
6. Ye JC, Sung WK. Understanding geometry of encoder-decoder CNNs. Int Conf Mach Learn, 2019;97:7064–7073.
7. Gregor K, LeCun Y. Learning fast approximations of sparse coding. Int Conf Mach Learn, 2010, p. 399–406.
8. Monga V, Li Y, Eldar YC. Algorithm unrolling: interpretable, efficient deep learning for signal and image processing. IEEE Signal Process Mag. 2021;38(2):18–44.
9. Hammernik K, Klatzer T, Kobler E, Recht MP, Sodickson DK, Pock T, Knoll F. Learning a variational network for reconstruction of accelerated MRI data. Magn Reson Med. 2018;79(6):3055–71.
10. Sun J, Li H, Xu Z et al. Deep ADMM-Net for compressive sensing MRI. Adv Neural Inf Proces Syst, 2016;29:10–18.
11. Eldar YC, Kutyniok G. Compressed sensing: theory and applications. Cambridge: Cambridge University Press; 2012.
12. Unser M. A representer theorem for deep neural networks. J Mach Learn Res. 2019;20(110):1–30.
13. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the Brain. Psychol Rev. 1958;65(6):386.
14. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323(6088):533–6.
15. Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Syst. 1989;2(4):303–14.
16. Telgarsky M. Benefits of depth in neural networks. In: Conference on learning theory. PMLR; 2016, pp. 1517–1539.
17. Eldan R, Shamir O. The power of depth for feedforward neural networks. In: Conference on learning theory. PMLR; 2016. pp. 907–940.
18. Raghu M, Poole B, Kleinberg J, Ganguli S, Sohl-Dickstein J. On the expressive power of deep neural

networks. In: International conference on machine learning. PMLR; 2017. pp. 2847–2854.
19. Yarotsky D. Error bounds for approximations with deep ReLU networks. Neural Netw. 2017;94:103–14.
20. Bishop CM. Pattern recognition and machine learning. New York: Springer; 2006.
21. Schölkopf B, Smola AJ, Bach F, et al. Learning with kernels: support vector machines, regularization, optimization, and beyond. London: MIT Press; 2002.
22. Vapnik V. The nature of statistical learning theory. New York: Springer Science & Business Media; 2013.
23. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings; 2011, pp. 315–323.
24. LeCun YA, Bottou L, Orr GB, Müller K. Efficient BackProp. In: Neural networks: tricks of the trade, Lecture notes in computer science. Berlin, Heidelberg: Springer; 2012. p. 9–48.
25. Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern. 1980;36 (4):193–202.
26. Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol. 1962;160(1):106–54.
27. Qayyum A, Anwar SM, Awais M, Majid M. Medical image retrieval using deep convolutional neural network. Neurocomputing. 2017;266:8–20.
28. Ibtehaz N, Rahman MS. MultiResUNet: rethinking the U-Net architecture for multimodal biomedical image segmentation. Neural Netw. 2019;121:74–87.
29. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hub-bard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989;1(4):541–51.
30. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: IEEE conference on computer vision and pattern recognition, 2015, p. 3431–3440.
31. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics; 2015. p. 315–323.
32. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
33. Hosseini SAH, Yaman B, Moeller S, Hong M, Akçakaya M. Dense recurrent neural networks for accelerated MRI: history-cognizant unrolling of optimization algorithms. IEEE J Select Topics Signal Process. 2020;14(6):1280–91.
34. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge: MIT Press; 2016.
35. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
36. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, December 2014.
37. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Adv Neural Inform Process Syst. 2017, pp. 5998–6008.
38. Cha E, Oh G, Ye JC. Geometric approaches to increase the expressivity of deep neural networks for MR reconstruction. IEEE J Select Topic Signal Process. 2020;14 (6):1292–305.
39. Zhou D-X. Universality of deep convolutional neural networks. Appl Comput Harmon Anal. 2020;48(2): 787–94.
40. Belkin M, Hsu D, Ma S, Mandal S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proc Natl Acad Sci. 2019;116(32): 15849–54.
41. Belkin M, Hsu D, Xu J. Two models of double descent for weak features. SIAM J Math Data Sci. 2020;2(4): 1167–80.
42. Chen SS, Donoho DL, Saunders MA. Atomic decomposition by basis pursuit. SIAM Rev. 2001;43(1):129–59.
43. Daubechies I, Defrise M, De Mol C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun Pure Appl Math. 2004;57(11):1413–57.
44. Xin B, Wang Y, Gao W, Wipf D, Wang B. Maximal sparsity with deep networks? Adv Neural Inf Process Syst, 2016;29:4340–4348.
45. Chen X, Liu J, Wang Z, Yin W. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Red Hook, NY, USA: Curran Associates Inc.; 2018, pp. 9079–9089.
46. Liu J, Chen X, Wang Z, Yin W ALISTA: analytic weights are as good as learned weights in LISTA. In: International conference on learning representations. 2019.
47. Solomon O, Eldar YC, Mutzafi M, Segev M. SPARCOM: sparsity based super-resolution correlation microscopy. SIAM J Imag Sci. 2019;12(1): 392–419.
48. Wang Z, Liu D, Yang J, Han W, Huang T. Deep networks for image super-resolution with sparse prior. In: Proceedings of the IEEE international conference on computer vision. 2015, pp. 370–378.
49. Hauptmann A, Lucka F, Betcke M, Huynh N, Adler J, Cox B, Beard P, Ourselin S, Arridge S. Model-based learning for accelerated, limited-view 3-D photoacoustic tomography. IEEE Trans Med Imaging. 2018;37(6):1382–93.
50. Adler J, Öktem O. Learned primal-dual reconstruction. IEEE Trans Med Imaging. 2018;37(6):1322–32.
51. Solomon O, Cohen R, Zhang Y, Yang Y, He Q, Luo J, van Sloun RJG, Eldar YC. Deep unfolded robust PCA with application to clutter suppression in ultrasound. IEEE Trans Med Imaging. 2020;39(4):1051–63.
52. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE

conference on computer vision and pattern recognition. 2016, pp. 770–778.
53. Dardikman-Yoffe G, Eldar YC. Learned SPARCOM: unfolded deep super-resolution microscopy. Opt Express. 2020;28(19):27736–63.
54. Hubel DH, Wiesel TN. Receptive fields of single neurones in the cat's striate cortex. J Physiol. 1959;148(3):574–91.
55. Quiroga RQ, Reddy L, Kreiman G, Koch C, Fried I. Invariant visual representation by single neurons in the human brain. Nature. 2005;435(7045):1102–7.
56. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;290 (5500):2323–6.
57. Tenenbaum JB, De Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;290(5500):2319–23.
58. Kingma DP, Welling M. Auto-encoding variational Bayes. In: International conference on learning representations. 2014.
59. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. Adv Neural Inform Process Syst. 2014, 27.
60. Rezende D, Mohamed S. Variational inference with normalizing flows. In: International conference on machine learning. PMLR; 2015, pp. 1530–1538.