A homotopy training algorithm for fully connected neural networks

Qipin Chen and Wenrui Hao

Department of Mathematics, Pennsylvania State University, University Park, PA 16802, USA

In this paper, we present a homotopy training algorithm (HTA) to solve optimization problems arising from fully connected neural networks with complicated structures. The HTA dynamically builds the neural network starting from a simplified version and ending with the fully connected network via adding layers and nodes adaptively. Therefore, the corresponding optimization problem is easy to solve at the beginning and connects to the original model via a continuous path guided by the HTA, which provides a high probability of obtaining a global minimum. By gradually increasing the complexity of the model along the continuous path, the HTA provides a rather good solution to the original loss function. This is confirmed by various numerical results including VGG models on CIFAR-10. For example, on the VGG13 model with batch normalization, HTA reduces the error rate by 11.86% on the test dataset compared with the traditional method. Moreover, the HTA also allows us to find the optimal structure for a fully connected neural network by building the neural network adaptively.

1. Introduction

The deep neural network (DNN) model has been experiencing an extraordinary resurgence in many important artificial intelligence applications since the late 2000s. In particular, it has been able to produce state-of-the-art accuracy in computer vision [1], video analysis [2], natural language processing [3] and speech recognition [4]. In the annual contest ImageNet Large Scale Visual Recognition Challenge (ILSVRC), the deep convolutional neural network (CNN) model has achieved the best classification accuracy since 2012 and has exceeded human ability on such tasks since 2015 [5].
The success of learning through neural networks with large model size, i.e. deep learning, is widely believed to be the result of being able to adjust millions to hundreds of millions of parameters to achieve close approximations to the target function. The approximation is usually obtained by minimizing its output error over a training set consisting of a significantly large amount of samples. Deep learning methods, as the rising star among all machine learning methods in recent years, have already had great success in many applications. Many advancements \([6–9]\) in deep learning have been made in the last few years. However, as the size of new state-of-the-art models continues to grow larger, they rely more heavily on efficient algorithms for training and making inferences from such models. This clearly places strong limitations on the application scenarios of DNN models for robotics \([10]\), auto-pilot automobiles \([11]\) and aerial systems \([12]\). At present, there are two big challenges in fundamentally understanding DNNs:

— How to efficiently solve the highly nonlinear and non-convex optimization problems that arise during the training of a deep learning model.
— How to design a DNN structure for specific problems.

In order to solve these challenges, in this paper, we will present a new training algorithm based on the homotopy continuation method \([13–15]\), which has been successfully used to study nonlinear problems such as nonlinear differential equations \([16–18]\), hyperbolic conservation laws \([19,20]\), data-driven optimization \([21,22]\), physical systems \([23]\) and some more complex free boundary problems arising from biology \([24,25]\). In order to tackle the nonlinear optimization problem in DNN, the homotopy training algorithm (HTA) is designed and shows efficiency and feasibility for fully connected neural networks with complex structures. The HTA also provides a new way to design a deep fully connected neural network with an optimal structure. This homotopy set-up presented in this paper can also be extended to other neural networks such as CNN and RNN. In this paper, we focus on fully connected DNNs only. The rest of this paper is organized as follows. We first introduce the HTA in \(\S2\) and then discuss the theoretical analysis in \(\S3\). Several numerical examples are given in \(\S4\) to demonstrate the accuracy and efficiency of the HTA. Finally, the applications of HTA to computer vision is given in \(\S5\).

2. Homotopy training algorithm

The basic idea of HTA is to train a simple model at the beginning, then adaptively increase the structure’s complexity, and eventually to train the original model. We will illustrate the idea of the homotopy set-up by using a fully connected neural network with \(x = (x_1, \ldots, x_n)^T\) as the input and \(y = (y_1, \ldots, y_m)^T\) as the output. More specifically, for a single-hidden layer, the neural network (figure 1) can be written as

\[
y = f(x) = W^T_2 \sigma(W^T_1 x + \beta_1) + \beta_2,
\]

where \(\sigma\) is the activation function (e.g. ReLU), \(W_1 \in R^{n \times d_1}\) and \(W_2 \in R^{d_1 \times m}\) are parameter matrices representing the weighted summation, \(\beta_1 \in R^{d_1}\) and \(\beta_2 \in R^m\) are vectors representing bias, and \(d_1\) is the number of nodes of the single-hidden layer, namely, the width. Similarly, a fully connected neural network with two hidden layers (figure 2) is written as

\[
y = f(x) = W^T_3 \sigma(W^T_2 \sigma(W^T_1 x + \beta_1) + \beta_2) + \beta_3,
\]

where \(W_1 \in R^{n \times d_1}\), \(W_2 \in R^{d_1 \times d_2}\), \(W_3 \in R^{d_2 \times m}\), \(\beta_1 \in R^{d_1}\), \(\beta_2 \in R^{d_2}\), \(\beta_3 \in R^m\), and \(d_2\) is the width of the second layer.

Then, the homotopy continuation method is introduced to track the minimizer of (2.1) to the minimizer of (2.2) by setting

\[
y(t) = H_1(x; W_j, \beta_j, t) = (1 - t)[W^T_2 \sigma(W^T_1 x + \beta_1) + \beta_2]
+ t[W^T_3 \sigma(W^T_2 \sigma(W^T_1 x + \beta_1) + \beta_2) + \beta_3] \quad j = 1, 2, 3.
\]
Then, an optima of (2.2) will be obtained by tracking the homotopy parameter \( t \) from 0 to 1. The idea is that the model of (2.1) is easier to train than that of (2.2). Moreover, the homotopy set-up will follow the universal approximation theory [26,27] to find an approximation trajectory to reveal the real nonlinear relationship between the input \( x \) and the output \( y \). Similarly, we can extend this homotopy idea to any two layers

\[
H_i(x; \theta, t) = (1 - t)y_i(x; \theta) + ty_{i+1}(x; \theta),
\]

where \( y_i(x; \theta) \) is the approximation of a fully connected neural network with \( i \) layers and \( \theta \) represents parameters that are weights of the neural network. In this case, we can train a fully connected neural network ‘node-by-node’ and ‘layer-by-layer’. This computational algorithm can significantly reduce computational costs of deep learning, which is used on large-scale data and complex problems.

Using the homotopy set-up, we are able to rewrite the ANN, CNN and RNN in terms of a specific start system such as (2.1). After designing a proper homotopy, we need to train this model with some datasets. In the homotopy set-up, we need to solve the following optimization problem:

\[
\theta(t) = \arg \min_{\theta} \sum_{j=1}^{N} \| H_j(X_{ij}; \theta, t) - Y_{ij} \|^2_{U_j}
\]
and the stochastic gradient descent method, instead of solving nonlinear equations (2.7) directly. However, in practice, we will use some advanced optimization methods for solving (2.5), such as solve it by using (2.4), which is equivalent to solving the following nonlinear equations: 

$$\text{Hom}(\theta, t) := t \nabla_{\theta} f_2(\theta) + (1 - t) \left( \nabla_{\theta} \tilde{f}_2(\theta) - \nabla_{\theta} \tilde{f}_2(\theta^0) \right) = 0,$$  

(2.7)

where $\tilde{f}_2$ is the objective function with the activation function of the second hidden layer as the identity and $\theta^0$ is constructed as 

$$W^0_1 = W^*_1, \quad W^0_2 = [W^*_2, 0, 0], \quad W^0_3 = [1, 0, 0]^T$$

and 

$$\beta^0_1 = \beta^*_1, \quad \beta^0_2 = [\beta^*_2, 0, 0]^T, \quad \beta^0_3 = 0.$$ 

By noticing that $\nabla_{\theta} f_2(\theta^0) = \nabla_{\theta} f_1(\theta^*_1) = 0$, we have that $\text{Hom}(\theta^0, 0) = 0$, which implies that the neural network with a single layer can be rewritten as a special form of the neural network with two layers. Then, we can solve the optimization problem $\min f_2(\theta)$ by tracking (2.7) with respect to $t$ from 0 to 1. This homotopy technique is quite often used in solving nonlinear equations [16–18]. However, in practice, we will use some advanced optimization methods for solving (2.5), such as the stochastic gradient descent method, instead of solving nonlinear equations (2.7) directly.

3. Theoretical analysis

In this section, we analyse the convergence of HTA between any two layers, namely, from the $i$th layer to the $i + 1$th layer. Assuming that we have a known minimizer of a fully connected neural network with $i$ layers, we prove that we can get a minimizer by adding the $i + 1$th layer through the HTA. First, we consider the expectation of the loss function as 

$$E_{\xi}(\mathcal{L}(H_i(x_{\xi}; \theta, t), y_{\xi})), \quad (3.1)$$

where $\mathcal{L}$ is the categorical cross-entropy loss function [30,31] and is defined as $\mathcal{L}(x, y) = -x(y) + \log(\sum_i e^{x[i]})$, and $\xi$ is a random variable due to random algorithms for solving the optimization problem for each given $t$. 

---

**Algorithm 1.** The HTA algorithm for a neural network with two hidden layers.

1: Solve the optimization problem $\mathcal{L}(H_1(x; \theta, 0), y)$ and denote the solution as $\theta^*_1$;
2: Set an initial guess as $\theta^0 = \theta^*_1 \cup \theta^0_2$ and $N = \frac{1}{\delta t}$, where $\delta t$ is the homotopy stepsize;
3: for $i = 1, \ldots, N$ do
4: \hspace{1em} Solve $\theta^i = \arg\min \mathcal{L}(H_1(x; \theta, i\delta t), y)$ by using $\theta^{i-1}$ as the initial guess;
5: end for

where $X^t$ and $Y^t$ represent data points and $N$ is the number of data points in a mini-batch. In this optimization, the homotopy set-up tracks the optima from a simpler optimization problem to a more complex one. The loss function in (2.5) could be changed to other types of entropy functions [28,29].

A simple illustration. We consider a simple neural network with two hidden layers to approximate a scalar function $y = f(x)$ (the width of two hidden layers are 2 and 3, respectively). The detailed HTA algorithm for training this neural network is listed in algorithm 1. Then, the neural network with a single layer in (2.1) gives us that $W_1 \in R^{1 \times 2}$, $W_2 \in R^{2 \times 1}$, $\beta_1 \in R^{2 \times 1}$ and $\beta_2 \in R$. By denoting all the weights $W_1, W_2, \beta_1, \beta_2$ as $\theta_1$, the optimization problem (2.5) for $t = 0$ is formulated as $\min f_1(\theta_1)$. Then, a minimizer $\theta^*_1 = (W^*_1, W^*_2, \beta^*_1)$ satisfies the necessary condition

$$\nabla_{\theta} f_1(\theta^*_1) = 0.$$

Similarly, for the neural network with two hidden layers, we have that, in (2.2), $W_2 \in R^{2 \times 3}$ and $\beta_2 \in R^{3 \times 1}$ are changed, $W_3 \in R^{3 \times 1}$ and $\beta_3 \in R$. Then, all the new variables introduced by the second hidden layer ($W_3, \beta_3$ and part of $W_2$ and $\beta_2$) are denoted as $\theta_2$. The total variables $\theta = \theta_1 \cup \theta_2$ formulate the optimization problem of two hidden layers, namely, (2.5) for $t = 1$, as $\min f_2(\theta)$ and solve it by using (2.4), which is equivalent to solving the following nonlinear equations:

$$\text{Hom}(\theta, t) := t \nabla_{\theta} f_2(\theta) + (1 - t) \left( \nabla_{\theta} \tilde{f}_2(\theta) - \nabla_{\theta} \tilde{f}_2(\theta^0) \right) = 0,$$  

(2.7)
For simplicity, we denote
\[
F(\theta; \xi, t) := \mathcal{L}(H_i(x_\xi; \theta, t), y_\xi),
\]
(3.2)
\[
f(\theta; t) := \mathbb{E}_\xi[F(\theta; \xi, t)]
\]
(3.3)
and
\[
G(\theta; \xi, t) := \nabla_\theta F(\theta; \xi, t),
\]
(3.4)
where the index \( i \) does not contribute to the analysis and therefore is ignored in our notation.

By denoting \( \theta^*_t := \arg\min_{\theta} f(\theta; t) \), we define our stochastic gradient scheme for any given \( t \):
\[
\theta_{k+1} = \theta_k - \gamma_k G(\theta_k; \xi_k, t).
\]
(3.5)
First, we have the following convergence theorem for any given \( t \) with the sigmoid activation function.

**Theorem 3.1 (Non-convex Convergence).** If \( \nabla_\theta H_i(x_\xi; \theta, t) \) is bounded for a given \( t \), namely, \( |\nabla_\theta H_i(x_\xi; \theta, t)| \leq M_l \), and \( \{\theta_k\} \) is contained in a bounded open set, supposing that (3.18) is run with a step-size sequence satisfying
\[
\sum_{k=1}^{\infty} \gamma_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \gamma_k^2 < \infty,
\]
(3.6)
then we have
\[
\mathbb{E} \left[ \frac{1}{A_k} \sum_{k=1}^{K} \gamma_k \|\nabla f(\theta_k; t)\|^2_2 \right] \to 0 \quad \text{as} \quad K \to \infty \quad \text{with} \quad A_k := \sum_{k=1}^{K} \gamma_k.
\]
(3.7)
Proof. First, we prove the Lipschitz-continuous objective gradients condition [32], which means that \( f(\theta; t) \) is \( C^1 \) and \( \nabla f(\theta; t) \) is Lipschitz continuous with respect to \( \theta \):

— \( f(\theta; t) \) is \( C^1 \). Since \( y_i(x_\xi; \theta) \in C^1 \) for \( \theta \), we have \( H(x_\xi; \theta, t) \in C^1 \). Moreover, since \( \mathcal{L}(., y) \) is \( C^1 \), we have that \( F(\theta; \xi, t) \in C^1 \) or that \( \nabla_\theta F(\theta; \xi, t) \) is continuous. Considering
\[
\nabla_\theta f(\theta; t) = \nabla_\theta \mathbb{E}_\xi[F(\theta; \xi, t)] = \mathbb{E}_\xi(\nabla_\theta F(\theta; \xi, t)),
\]
(3.8)
we have that \( \nabla_\theta f(\theta; t) \) is continuous or that \( f(\theta; t) \in C^1 \).

— \( \nabla f(\theta; t) \) is Lipschitz continuous. Since
\[
\nabla_\theta f(\theta; t) = \mathbb{E}_\xi[\nabla_x \mathcal{L}(H(x_\xi; \theta, t), y_\xi)\nabla_\theta H(x_\xi; \theta, t)],
\]
(3.9)
we will prove that both \( \nabla_x \mathcal{L}(H(x_\xi; \theta, t), y_\xi) \) and \( \nabla_\theta H(x_\xi; \theta, t) \) are bounded and Lipschitz continuous. Because both \( \sigma(x) = 1/(1 + e^{-x}) \) and \( \sigma'(x) = \sigma(x)(1 - \sigma(x)) \) are Lipschitz continuous and \( \{\theta_k\} \) is bounded (assumption of theorem 3.1), \( \nabla_\theta H(x_\xi; \theta, t) \) is Lipschitz continuous. \( (x_\xi \text{ is bounded because the size of our dataset is finite.}) \)

By differentiating \( \mathcal{L}(x, y) \), we have
\[
\nabla_x \mathcal{L}(x, y) = \left( -\delta_y^1 + \frac{e^{x_i}}{\sum_j e^{x_j}}, \ldots, -\delta_y^n + \frac{e^{x_n}}{\sum_j e^{x_j}} \right),
\]
(3.10)
where \( \delta \) is the Kronecker delta. Since
\[
\frac{\partial}{\partial x_k} \sum_j e^{x_j} = \begin{cases} \frac{e^{x_k} \sum_j e^{x_j} - (e^{x_k})^2}{(\sum_j e^{x_j})^2} & k = i, \\ -\frac{e^{x_i} e^{x_k}}{(\sum_j e^{x_j})^2} & k \neq i, \end{cases}
\]
(3.11)
which implies that \(|(\partial/\partial x_k)(e^{x_k}/\sum_j e^{x_j})| \leq 2 \), we see that \( \nabla_x \mathcal{L}(., y) \) is Lipschitz continuous and bounded. Therefore, \( \nabla_x \mathcal{L}(H(x_\xi; \theta, t), y_\xi) \) is Lipschitz continuous and bounded. Thus, \( \nabla f(\theta; t) \) is Lipschitz continuous.

Second, we prove the first- and second-moment limits condition [32]:
(a) according to our theorem’s assumption, \( \{ \theta_k \} \) is contained in an open set that is bounded. Since \( f \) is continuous, \( f \) is bounded;
(b) since \( G(\theta_k; \xi_k, t) = \nabla \phi F(\theta_k; \xi_k, t) \) is continuous, we have
\[
\mathbb{E}_{\xi_k}[G(\theta_k; \xi_k, t)] = \nabla \phi F(\theta_k; \xi_k, t) = \nabla \phi(\theta_k; t).
\] (3.12)
Therefore,
\[
\nabla \phi(\theta_k; t)^T \mathbb{E}_{\xi_k}[G(\theta_k; \xi_k, t)] = \nabla \phi(\theta_k; t)^T \nabla \phi(\theta_k; t) = \| \nabla \phi(\theta_k; t) \|^2 \geq u \| \nabla \phi(\theta_k; t) \|^2
\] (3.13)
for \( 0 < u \leq 1 \).
On the other hand, we have
\[
\| \mathbb{E}_{\xi_k}[G(\theta_k; \xi_k, t)] \|_2 = \| \nabla \phi(\theta_k; t) \|_2 \leq u_G \| \nabla \phi(\theta_k; t) \|_2
\] (3.14)
for \( u_G \geq 1 \).
(c) Since \( \nabla F(\theta_k; \xi_k, t) \) is bounded for a given \( t \), we have \( \mathbb{E}_{\xi_k}[\| \nabla F(\theta_k; \xi_k, t) \|_2^2] \) is also bounded. Thus,
\[
\mathbb{E}_{\xi_k}[G(\theta_k; \xi_k, t)] := \mathbb{E}_{\xi_k}[\| G(\theta_k; \xi_k, t) \|_2^2] - \| \mathbb{E}_{\xi_k}[G(\theta_k; \xi_k, t)] \|_2^2 \leq \mathbb{E}_{\xi_k}[\| G(\theta_k; \xi_k, t) \|_2^2],
\] (3.15)
which implies that \( \mathbb{E}_{\xi_k}[G(\theta_k; \xi_k, t)] \) is bounded.

We have checked Assumptions 4.1 and 4.3 in [32]. By Theorem 4.10 in [32], with the diminishing step-size, namely,
\[
\sum_{k=1}^{\infty} \gamma_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \gamma_k^2 < \infty,
\] (3.16)
the following convergence is obtained:
\[
\mathbb{E} \left[ \frac{1}{A_k} \sum_{k=1}^{K} \gamma_k \| \nabla \phi(\theta_k; t) \|_2^2 \right] \rightarrow 0 \quad \text{as} \ K \rightarrow \infty.
\] (3.17)

Second, we theoretically explore the existence of solution path \( \theta(t) \) when \( t \) varies from 0 to 1 for the convex case. The solution path of \( \theta(t) \) might be complex for the non-convex case, i.e. bifurcations, and is hard to analyse theoretically. Therefore, we analyse the HTA theoretically on the convex case only but apply it to non-convex cases in the numerical experiments. We redefine our stochastic gradient scheme for the homotopy process as
\[
\theta_{k+1} = \theta_k - \gamma_k G(\theta_k; \xi_k, t_k),
\] (3.18)
where \( \gamma_k \) is the learning rate and \( t_0 = 0, t_k \neq 1 \). Instead of considering the local convergence of the HTA in a neighbourhood of the global minimum, we proved the following theorem in a more general assumption, namely, \( f \) is a convex and differentiable objective function with a bounded gradient.

**Theorem 3.2 (Existence of solution path \( \theta(t) \)).** Assume that \( f(\cdot, \cdot) \) is convex and differentiable and that \( \| G(\theta; \xi, t) \| \leq M \). Then for stochastic gradient scheme (3.18), with a finite partition for \( t \) between \([0,1]\), we have
\[
\lim_{n \to \infty} \mathbb{E}[f(\tilde{\theta}_n, \tilde{t}_n)] = f(\theta^*_1, 1),
\] (3.19)
where \( \tilde{\theta}_n = \sum_{k=0}^{n} \gamma_k \theta_k / \sum_{k=0}^{n} \gamma_k \) and \( \tilde{t}_n = \sum_{k=0}^{n} \gamma_k t_k / \sum_{k=0}^{n} \gamma_k \).
Proof.

\[ \mathbb{E}[\|\theta_{k+1} - \theta^*_a\|_2^2] = \mathbb{E}[\|\theta_k - \gamma_k G(\theta_k; \xi_k, t_k) - \theta^*_a\|_2^2] \\
- 2\mathbb{E}[\langle \theta_k - \gamma_k G(\theta_k; \xi_k, t_k) - \theta^*_a, \theta_{k+1}^* - \theta^*_a \rangle] \\
+ \mathbb{E}[\|\theta_{k+1}^* - \theta^*_a\|_2^2]. \quad (3.20) \]

By defining

\[ A_k = -2\mathbb{E}[\langle \theta_k - \gamma_k G(\theta_k; \xi_k, t_k) - \theta^*_a, \theta_{k+1}^* - \theta^*_a \rangle] + \mathbb{E}[\|\theta_{k+1}^* - \theta^*_a\|_2^2], \quad (3.21) \]

we have \( \sum_{k=0}^n A_k \leq A < \infty \) since \( t \in [0, 1] \) has a finite partition. Therefore, we obtain

\[ \mathbb{E}[\|\theta_{k+1} - \theta^*_a\|_2^2] = \mathbb{E}[\|\theta_k - \gamma_k G(\theta_k; \xi_k, t_k) - \theta^*_a\|_2^2] + A_k \\
= \mathbb{E}[\|\theta_k - \theta^*_a\|_2^2] - 2\gamma_k \mathbb{E}[\langle G(\theta_k; \xi_k, t_k), \theta_k - \theta^*_a \rangle] + \gamma_k^2 \mathbb{E}[\|G(\theta_k; \xi_k, t_k)\|_2^2] + A_k \\
\leq \mathbb{E}[\|\theta_k - \theta^*_a\|_2^2] - 2\gamma_k \mathbb{E}[\langle G(\theta_k; \xi_k, t_k), \theta_k - \theta^*_a \rangle] + \gamma_k^2 M^2 + A_k. \]

Since

\[ \mathbb{E}[\langle G(\theta_k; \xi_k, t_k), \theta_k - \theta^*_a \rangle] = \mathbb{E}_{\xi_0, \ldots, \xi_{k-1}}[\mathbb{E}_{\xi_k}[\langle G(\theta_k; \xi_k, t_k), \theta_k - \theta^*_a \rangle|\xi_0, \ldots, \xi_{k-1}]] \\
= \mathbb{E}_{\xi_0, \ldots, \xi_{k-1}}[\langle \nabla f(\theta_k; t_k), \theta_k - \theta^*_a \rangle|\xi_0, \ldots, \xi_{k-1}] \\
= \mathbb{E}[\langle \nabla f(\theta_k; t_k), \theta_k - \theta^*_a \rangle], \quad (3.22) \]

we have

\[ \mathbb{E}[\|\theta_{k+1} - \theta^*_a\|_2^2] \leq \mathbb{E}[\|\theta_k - \theta^*_a\|_2^2] - 2\gamma_k \mathbb{E}[\langle \nabla f(\theta_k; t_k), \theta_k - \theta^*_a \rangle] + \gamma_k^2 M^2 + A_k. \quad (3.23) \]

Due to the convexity of \( f(., t_k) \), namely,

\[ \langle \nabla f(\theta_k; t_k), \theta_k - \theta^*_a \rangle \geq f(\theta_k; t_k) - f(\theta^*_a; t_k), \]

we conclude that

\[ \mathbb{E}[\|\theta_{k+1} - \theta^*_a\|_2^2] \leq \mathbb{E}[\|\theta_k - \theta^*_a\|_2^2] - 2\gamma_k \mathbb{E}[f(\theta_k; t_k) - f(\theta^*_a; t_k)] + \gamma_k^2 M^2 + A_k \]

or

\[ 2\gamma_k \mathbb{E}[f(\theta_k; t_k) - f(\theta^*_a; t_k)] \leq -\mathbb{E}[\|\theta_{k+1} - \theta^*_a\|_2^2 - \|\theta_k - \theta^*_a\|_2^2] + \gamma_k^2 M^2 + A_k. \]

By summing up \( k \) from 0 to \( n \),

\[ 2 \sum_{k=0}^n \gamma_k \mathbb{E}[f(\theta_k; t_k) - f(\theta^*_a; t_k)] \\
\leq -\mathbb{E}[\|\theta_{k+1} - \theta^*_a\|_2^2 - \|\theta_0 - \theta^*_a\|_2^2] + M^2 \sum_{k=0}^n \gamma_k^2 + \sum_{k=0}^n A_k \\
\leq D^2 + M^2 \sum_{k=0}^n \gamma_k^2 + \sum_{k=0}^n A_k, \quad (3.26) \]

where \( D = \|\theta_0 - \theta^*_a\| \). Dividing \( 2 \sum_{k=0}^n \gamma_k \) on both sides, we have

\[ \frac{1}{2 \sum_{k=0}^n \gamma_k} \sum_{k=0}^n \gamma_k \mathbb{E}[f(\theta_k; t_k) - f(\theta^*_a; t_k)] \leq \frac{D^2 + M^2 \sum_{k=0}^n \gamma_k^2 + \sum_{k=0}^n A_k}{2 \sum_{k=0}^n \gamma_k} \\
\leq \frac{D^2 + M^2 \sum_{k=0}^n \gamma_k^2 + A}{2 \sum_{k=0}^n \gamma_k}. \]
According to the convexity of $f(\cdot, \cdot)$ and Jensen’s inequality [33]

$$\frac{1}{\sum_{k=0}^{n} y_k} \sum_{k=0}^{n} y_k \mathbb{E}[f(\theta_k; t_k)] \geq \mathbb{E}[f(\tilde{\theta}_n; \tilde{t}_n)],$$  \hspace{1cm} (3.27)

where $\tilde{\theta}_n = \sum_{k=0}^{n} y_k \theta_k / \sum_{k=0}^{n} y_k$ and $\tilde{t}_n = \sum_{k=0}^{n} y_k t_k / \sum_{k=0}^{n} y_k$.

Then, we have

$$\mathbb{E}[f(\tilde{\theta}_n; \tilde{t}_n)] - \frac{\sum_{k=0}^{n} y_k f(\theta_{k1}; t_k)}{\sum_{k=0}^{n} y_k} \leq \frac{D^2 + M^2 \sum_{k=0}^{n} y_k^2 + A}{2 \sum_{k=0}^{n} y_k}. \hspace{1cm} (3.28)$$

We choose $y_k$ such that $\sum_{k=0}^{n} y_k = \infty$ and $\sum_{k=0}^{n} y_k^2 < \infty$, for example, $y_k = 1/k$. Taking $n$ to infinite, we have

$$\lim_{n \to \infty} \mathbb{E}[f(\tilde{\theta}_n, \tilde{t}_n)] - \lim_{n \to \infty} \frac{\sum_{k=0}^{n} y_k f(\theta_{k1}; t_k)}{\sum_{k=0}^{n} y_k} \leq 0. \hspace{1cm} (3.29)$$

Since $t_k \neq 1$, $f(\cdot, \cdot)$ is continuous and $\sum_{k=0}^{n} y_k = \infty$, then we have

$$\lim_{n \to \infty} \frac{\sum_{k=0}^{n} y_k f(\theta_{k1}; t_k)}{\sum_{k=0}^{n} y_k} = f(\theta_{11}^*, 1), \hspace{1cm} (3.30)$$

which implies that

$$\lim_{n \to \infty} \mathbb{E}[f(\tilde{\theta}_n, \tilde{t}_n)] \leq f(\theta_{11}^*, 1). \hspace{1cm} (3.31)$$

Since $\tilde{t}_n \to 1$, we have

$$\mathbb{E}[f(\lim_{n \to \infty} \tilde{\theta}_n, 1)] \leq f(\theta_{11}^*, 1). \hspace{1cm} (3.32)$$

On the other hand, $\theta_{11}^*$ is the global minimum due to the convexity of $f$, and we have $\mathbb{E}[f(\lim_{n \to \infty} \tilde{\theta}_n, 1)] \geq f(\theta_{11}^*, 1)$. Thus, $\mathbb{E}[f(\lim_{n \to \infty} \tilde{\theta}_n, 1)] = f(\theta_{11}^*, 1)$ holds.

4. Numerical results

In this section, we demonstrate the efficiency and the feasibility of the HTA by comparing it with the traditional method, the stochastic gradient descent method. Although we have shown the convergence of the non-convex case with a $C^1$ activation function for any given $t$ in §3, in the numerical computation, we applied the HTA to more realistic cases with ReLU functions to demonstrate the generalizability. For both methods, we used the same hyper parameters, such as learning rate (0.05), batch size (128), and the number of epochs (380) on the same neural network for various problems. Due to the non-convexity of objective functions, both methods may get stuck at local optimas. We also ran the training process 15 times with different random initial guesses for both methods and reported the best results for each method.

(a) Function approximations

Example 4.1 (Single-hidden layer). The first example we considered is using a single-hidden-layer connected neural network to approximate function

$$f(x) = \sin(x_1 + x_2 + \cdots + x_n), \hspace{1cm} (4.1)$$

where $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$. The width of the single-hidden layer NN is 20 and the width of the hidden layer of the initial state of HTA is set to be 10. Then, the homotopy set-up is written as

$$H(x; \theta, t) = (1 - t)y_1(x; \theta) + ty_2(x; \theta),$$

where $y_1$ and $y_2$ are the fully connected neural networks (NNs) with 10 and 20 as their width of hidden layers, respectively. In particular, we have

$$y_1(x; \theta) = W_{21} \cdot r(W_{11} \cdot x + b_{11}) + b_{21} \hspace{1cm} (4.2)$$
and

\[ y_2(x; \theta) = W_2 \cdot r(W_1 \cdot x + b_1) + b_{21}, \tag{4.3} \]

where \( x \in \mathbb{R}^{n \times 1} \), \( W_{11} \in \mathbb{R}^{10 \times n} \), \( b_{11} \in \mathbb{R}^{10 \times 1} \), \( W_{21} \in \mathbb{R}^{1 \times 10} \), \( b_{21} \in \mathbb{R}^{1 \times 1} \), \( W_1 = \begin{pmatrix} W_{11} \\ W_{12} \end{pmatrix} \in \mathbb{R}^{20 \times n} \), \( b_1 = \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix} \in \mathbb{R}^{20 \times 1} \) and \( W_2 = (W_{21}, W_{22}) \in \mathbb{R}^{1 \times 20} \). We use the ReLU function \( r(x) = \max\{0, x\} \) as our activation function. For \( n \leq 3 \), we used the uniform grid points, where the sample points are the Cartesian products of uniformly sampled points of each dimension. Then, the size of the training dataset is \( 10^2 n \). For \( n \geq 4 \), we employed the sparse grid \([34,35]\) with level 6 as sample points. For each \( n \), 90% of the dataset is used for training while 10% is used for testing. The loss curves of one-dimensional and two-dimensional cases are shown in figures 3 and 4. By choosing \( \Delta t = 0.5 \), the testing loss of HTA (for \( t = 1 \)) is lower than that of the traditional training algorithm. Figure 5 shows the comparison between the traditional method and HTA for the one-dimensional case while figure 6 shows the comparison of the two-dimensional case by using contour curves. All the results of up to \( n = 5 \) are summarized in table 1, which lists the test loss between the HTA and the traditional training algorithm. It shows clearly that the HTA method is more efficient than the traditional method.
Figure 5. Approximation results of \( \sin(x) \). (a) Approximation results of the traditional method. (b) Approximation results of the HTA. (Online version in colour.)

Figure 6. Approximation results of \( \sin(x_1 + x_2) \). (a) Approximation results of the traditional method. (b) Approximation results of the HTA. (Online version in colour.)

Table 1. Testing loss of one-hidden-layer NNs.

| dimensions \((n)\) | test loss | test loss with HTA | number of grid points |
|----------------|-----------|--------------------|----------------------|
| 1              | 0.078     | 0.005              | \(10^2\) (uniform grid) |
| 2              | 0.195     | 0.152              | \(10^4\) (uniform grid) |
| 3              | 0.347     | 0.213              | \(10^6\) (uniform grid) |
| 4              | 0.393     | 0.299              | 2300 (sparse grid) |
| 5              | 0.493     | 0.352              | 5503 (sparse grid) |

Example 4.2 (Multiple hidden layers). The second example is using a two-hidden-layer fully connected neural network to approximate the same function in example 4.1 for the multi-dimensional case. Since the approximation of the neural network with a single hidden layer is not effective for \( n > 3 \) (table 1), we use a two-hidden-layer fully connected neural network with 20 nodes for each layer. Then, we use the following homotopy set-up to increase the width of each layer from 10 to 20

\[
H_1(x; \theta, t) = (1 - t)y_1(x; \theta) + ty_2(x; \theta) \quad \text{and} \quad H_2(x; \theta, t) = (1 - t)y_2(x; \theta) + ty_3(x; \theta),
\]  

\[(4.4)\]
Figure 7. Testing loss of approximating \( \sin(x_1 + x_2 + \cdots + x_n) \). (a) Testing loss of \( n = 5 \). (b) Testing loss of \( n = 6 \). (Online version in colour.)

Table 2. Testing loss of two-hidden-layer NNs.

| dimensions (\( n \)) | test loss | test loss with HTA | number of grid points |
|----------------------|-----------|--------------------|----------------------|
| 5                    | 0.307     | 0.074              | 5503 (sparse grid)   |
| 6                    | 0.340     | 0.194              | 10625 (sparse grid)  |
| 7                    | 0.105     | 0.029              | 18943 (sparse grid)  |
| 8                    | 0.079     | 0.022              | 31745 (sparse grid)  |

where \( y_1, y_2 \) and \( y_3 \) represent neural networks with width (10,10), (10,20) and (20,20), respectively. The rationale is that the first homotopy function, \( H_1(x; \theta, t) \), increases the width of the first layer while the second homotopy function, \( H_2(x; \theta, t) \), increases the width of the second layer. The size of the training data and the strategy of choosing \( \Delta t \) is the same as in example 4.1. Table 2 shows the results of the approximation, and figure 7 shows the testing curves for \( n = 5 \) and \( n = 6 \). The HTA achieves higher accuracy than the traditional method.

(b) Parameter estimation

Parameter estimation often requires a tremendous number of model evaluations to obtain the solution information on the parameter space \([21,36,37]\). However, this large number of model evaluations becomes very difficult and even impossible for large-scale computational models \([38,39]\). Then, a surrogate model needs to be built in order to approximate the parameter space. Neural networks provide an effective way to build the surrogate model. But an efficient training algorithm of neural networks is needed to obtain an effective approximation especially for limited sample data on parameter space. We will use the Van der Pol equation as an example to illustrate the efficiency of HTA on the parameter estimation.

Example 4.3. We applied the HTA to estimate the parameters of the Van der Pol equation

\[
y'' - \mu(k - y^2)y' + y = 0 \quad \text{with} \quad y(0) = 2 \quad \text{and} \quad y'(0) = 0.
\]

(4.5)

In order to estimate the parameters \( \mu \) and \( k \) for a given data \( \tilde{y}(t; \mu, k) \), we first use a single-hidden-layer fully connected neural network to build a surrogate model with \( \mu \) and \( k \) as inputs and \( y(1) \) as the output. Our training dataset is chosen on \( 1 \leq \mu \leq 10 \) and \( 1 \leq k \leq 10 \) with 8281 (with 0.1 as the mesh size). This neural network is trained by both the traditional method and the HTA. The
The testing dataset is 961 uniform grid points on $11 \leq \mu_i \leq 14$ and $11 \leq k_i \leq 14$ with 0.1 as the mesh size for both $\mu$ and $k$. The testing loss curves of the traditional method and HTA are shown in figure 8: after $5 \times 10^4$ steps, the testing loss is 0.007 for HTA and 0.220 for the traditional method. We also compared these two surrogate models (traditional method and HTA) with the numerical ODE solution $y(1; \mu, k)$ in figure 9. This comparison shows that the approximation of the HTA is closer to the ODE model than the traditional method.

Once we built surrogate models, then we moved to a parameter estimation step for any given data $\tilde{y}(t; \mu, k)$ to solve the following optimization problem:

$$\min_{\mu, k} (S(\mu, k) - \tilde{y}(1))^2,$$

(4.6)

where $S(\mu, k)$ is the surrogate neural network model and $\tilde{y}(1)$ is the data when $t = 1$. In our example, we generated ‘artificial data’ on the testing dataset. We use the SDG to solve the optimization problem with $\rho = k = 11$ as the initial guess. We define the error of the parameter
Table 3. Parameter estimation results.

| sample points ($\mu_i, k_i$) | optima of traditional method ($\mu_i^*, k_i^*$) | optima of HTA ($\mu_i^*, k_i^*$) |
|-------------------------------|-----------------------------------------------|-------------------------------|
| (11.1, 12.9)                 | (11.9, 11.6)                                 | (11.9, 12.6)                 |
| (11.9, 13.1)                 | (11.9, 11.7)                                 | (11.9, 12.7)                 |
| (12.6, 11.4)                 | (11.0, 11.0)                                 | (12.0, 11.4)                 |
| (13.2, 12.8)                 | (11.9, 11.5)                                 | (11.9, 12.5)                 |
| (13.9, 11.1)                 | (11.0, 11.0)                                 | (12.0, 11.2)                 |

estimation below

$$\text{Err}_{PE} = \frac{\sum_{i=1}^{n} \sqrt{(\mu_i^* - \mu_i)^2 + (k_i^* - k_i)^2}}{n},$$

(4.7)

where $(\mu_i, k_i)$ is the sample point and $(\mu_i^*, k_i^*)$ is the optima of (4.6) for a given $\tilde{y}(1)$. Then, the error of HTA is 0.71 while the error of the traditional method is 1.48. We also list some results of parameter estimation for different surrogate models in Table 3. The surrogate model created by the HTA provides smaller errors than the traditional method for parameter estimation.

5. Applications to computer vision

Computer vision is one of the most common applications in the field of machine learning [40,41]. It has diverse applications, from designing navigation systems for self-driving cars [42] to counting the number of people in a crowd [43]. There are many different models that can be used for detection and classification of objects. Since our algorithm focuses on the fully connected neural networks, we will only apply our algorithm to computer vision models with fully connected neural networks. Therefore, in this section, we will use different visual geometry group (VGG) models [1] as an example to illustrate the application of HTA to computer vision. In computer vision, the VGG models use convolutional layers to extract features of the input picture and then flatten the output tensor to be a 512-dimension-long vector. The output long vector will be sent into the fully connected network (see Figure 10 for more details). In order to demonstrate the efficiency of HTA, we will apply it to the fully connected network part of the VGG models.

(a) Three states of HTA

The last stage of VGG models is a fully connected neural network that links convolutional layers of VGG to the classification categories of the Canadian Institute for Advanced Research (CIFAR-10) [44]. Then, the input is the long vector generated by convolutional layers (the width is 512), while the output is the 10 classification categories of CIFAR-10. In order to train this fully connected neural network, we construct a three-states set-up of HTA, which is shown in Figure 11. In this section, we use $x$ to represent the inputs generated by the convolutional layers ($x \in \mathbb{R}^{512}$) and $\theta$ to represent the parameters for each state. The size of $\theta$ may change for different states.

— **State 1.** For the first state, we construct a fully connected network with 2 hidden layers. The width of the $i$th hidden layer is set to be $w_i$. Then, it can be written as

$$y_1(x; \theta) = W_{31} \cdot r(W_{21} \cdot r(W_{11} \cdot x + b_{11}) + b_{21}) + b_3,$$

(5.1)

where $x \in \mathbb{R}^{512 \times 1}$, $W_{11} \in \mathbb{R}^{w_1 \times 512}$, $b_{11} \in \mathbb{R}^{w_1 \times 1}$, $W_{21} \in \mathbb{R}^{w_2 \times w_1}$, $b_{21} \in \mathbb{R}^{w_2 \times 1}$, $W_{31} \in \mathbb{R}^{10 \times w_2}$ and $b_3 \in \mathbb{R}^{1 \times 1}$. We use the ReLU function $r(x) = \max(0, x)$ as our activation function.

— **State 2.** For the second state, we add $(512 - w_1)$ nodes to the first hidden layer to recover the first hidden layer of the original model. Therefore, the formula of the second state
Figure 10. The structure of VGG models that consist of a convolution & pooling neural network and a fully connected neural network. The HTA is applied on the fully connected layers only. (Online version in colour.)

Figure 11. Three states of the VGG with the HTA. (Online version in colour.)

becomes

\[ y_2(x; \theta) = W_{31} \cdot r(\tilde{W}_2 \cdot r(W_1 \cdot x + b_1) + b_{21}) + b_3, \]  

(5.2)

where \( x \in \mathbb{R}^{512 \times 1} \), \( W_1 = (W_{11}, W_{12}) \in \mathbb{R}^{512 \times 512} \), \( b_1 = (b_{11}, b_{12}) \in \mathbb{R}^{512 \times 1} \) and \( \tilde{W}_2 = (W_{21}, W_{22}) \in \mathbb{R}^{w_2 \times 512} \). In particular, if we choose \( W_{12} = 0 \) and \( W_{22} = 0 \), we will recover \( y_1(x; \theta) \) of state 1.

— State 3. Finally, we recover the original structure of the VGG by adding \((512 - w_2)\) nodes to the second hidden layer

\[ y_3(x; \theta) = W_3 \cdot r(W_2 \cdot r(W_1 \cdot x + b_1) + b_2) + b_3, \]  

(5.3)

where \( W_3 = (W_{31}, W_{32}) \in \mathbb{R}^{10 \times 512} \), \( W_2 = \left(\frac{\tilde{W}_2}{\tilde{W}_2}\right) \in \mathbb{R}^{512 \times 512} \), \( b_2 = \left(\frac{b_{21}}{b_{22}}\right) \in \mathbb{R}^{512 \times 1} \). \( y_3(x; \theta) \) will be reduced to \( y_2(x; \theta) \) if \( \tilde{W}_2 = 0 \) and \( W_{32} = 0 \).
(b) Homotopic path

In order to connect these three states, we use two homotopic paths that are defined by the following homotopy functions:

\[ H_i(x; \theta, t) = (1 - t)y_i(x; \theta) + ty_{i+1}(x; \theta), \quad i = 1, 2. \]  

(5.4)

In this homotopy set-up, when \( t = 0 \), we already have an optimal solution \( \theta_i \) for the \( i \)th state and want to find an optimal solution for \( \theta \) for the \( i + 1 \)th state when \( t = 1 \). By tracking \( t \) from 0 to 1, we can discover a solution path \( \theta(t) \) since \( y_i(x; \theta) \) is a special form of \( y_{i+1}(x; \theta) \). Then, the loss functions for the homotopy set-up becomes

\[ L_i(x, y; \theta, t) = L(H_i(x; \theta, t), y), \]  

(5.5)

where \( L(x, y) \) is the loss function.

(c) Training process

We first optimize \( L(y_1(x; \theta_1), y) \) for the first state. The model structure is relatively simple to solve, and it efficiently obtains a local minimum or even a global minimum for the loss function. Then, the second set-up is to optimize \( L_1(x, y; \theta, t) \) by using \( \theta_1 \) as an initial condition for \( t = 0 \). By gradually tracking parameter \( t \) to 1, we obtain an optimal solution \( \theta_2 \) of \( L(y_2(x; \theta), y) \). The third set-up is to optimize \( L_2(x, y; \theta, t) \) by tracking \( t \) from 0 (\( \theta_2 \)) to 1. Then, we obtain an optimal solution, \( \theta_3 \), of \( L(y_3(x; \theta), y) \). Due to the continuous paths, the optimal solution \( \theta_{i+1} \) of \( i + 1 \)th state is connected to \( \theta_i \) of the \( i \)th state by the parameter \( t \). In this way, we can build our complex network adaptively.

(d) Numerical results on CIFAR-10

We tested the HTA with the three-state set-up on the CIFAR-10 dataset. We used VGG11, VGG13, VGG16 and VGG19 with batch normalization [1] as our base models. Figure 12 shows the comparison of validation loss between HTA and the traditional method on the VGG13 model. Using the HTA with VGG13 has a lower error rate (5.14\%) than the traditional method (5.82\%), showing that the HTA with VGG13 improves the error rate by 11.68\%. All of the results for the different models are shown in table 4. It is clearly seen that the HTA is more accurate than the
Table 4. The comparison between the HTA and the traditional method of VGG models on CIFAR-10.

| base model name | original error rate (%) | error rate with HTA (%) | rate of improvement (ROIs) (%) |
|-----------------|-------------------------|-------------------------|-----------------------------|
| VGG11           | 7.83                    | 7.02                    | 10.34                       |
| VGG13           | 5.82                    | 5.14                    | 11.68                       |
| VGG16           | 6.14                    | 5.71                    | 7.00                        |
| VGG19           | 6.35                    | 5.88                    | 7.40                        |

Table 5. The results of the algorithm for finding the optimal structure when applied to CIFAR-10: \( w_i \) stands for the optimal width of \( i \)th hidden layer.

| base model name | original error rate (%) | error rate with OSF (%) | \( w_1 \) | \( w_2 \) |
|-----------------|-------------------------|-------------------------|----------|----------|
| VGG11           | 7.83                    | 7.37                    | 480      | 20       |
| VGG13           | 5.82                    | 5.67                    | 980      | 310      |
| VGG16           | 6.14                    | 5.91                    | 752      | 310      |
| VGG19           | 6.35                    | 6.05                    | 752      | 310      |

traditional method for all of the different models. For example, the HTA with VGG11 results in an error rate of 7.02% while the traditional method results in 7.83% (an improvement of 10.34%). In addition, the HTA with VGG19 has an error rate of 5.88% compared with 6.35% with the traditional model (an improvement of 7.40%), and the HTA with VGG16 has an error rate of 5.71% while the traditional method has an error rate of 6.14% (a 7.00% improvement).

(e) The optimal structure of a fully connected neural network

Since the HTA builds the fully connected neural network adaptively, it also provides a way for us to find the optimal structure of the fully connected neural network; for example, we can find the number of layers and the width for each layer. We designed an algorithm to find the optimal structure based on HTA. First, we began with a minimal model; for example, in the VGG models of CIFAR-10, the minimal width of two hidden layers is 10 because of the 10 classification. Then, we applied the HTA to the first hidden layer by adding ‘node-by-node’, and we optimized the loss function dynamically with respect to the homotopy parameter. If the optimal width of the first hidden layer was found, then the weights of new added nodes were close to zero after optimization. Then, we moved to the second hidden layer and implemented the same process to train ‘layer-by-layer’. When the weights of the new added nodes for the second hidden layer became close to zero, we terminated the process.

**Numerical results on CIFAR-10.** When we applied the algorithm for finding the optimal structure to each VGG base model, we found that the results were more accurate than when we used the base model only. For VGG11 with batch normalization, we set \( \delta t = 1/2 \) and \( n_{\text{epoch}} = 50 \) and found the optimal structure whose widths of the first and second hidden layers are 480 and 20, respectively. The error rate with VGG11 was reduced to 7.37% while the error rate of the base model was 7.83%. In this way, our algorithm can reach higher accuracy but with a simpler structure. The rest of our experimental results for different VGG models is listed in table 5.

6. Conclusion

In this paper, we developed a HTA for the fully connected neural network models. This algorithm starts from a simple neural network and adaptively grows into a fully connected neural network with a complex structure. Then, the complex neural network can be trained by the HTA to attain a higher accuracy. The convergence of the HTA for each \( t \) is proved for the non-convex
optimization that arises from fully connected neural networks with a $C^1$ activation function. Then, the existence of solution path $\theta(t)$ is demonstrated theoretically for the convex case although it exists numerically in the non-convex case. Several numerical examples have been used to demonstrate the efficiency and feasibility of HTA. We also proved the convergence of HTA to the local optima if the optimization problem is convex. The application of HTA to computer vision, using the fully connected part of VGG models on CIFAR-10, provides better accuracy than the traditional method. Moreover, the HTA method provides an alternative way to find the optimal structure to reduce the complexity of a neural network. In this paper, we developed the HTA for fully connected neural networks only, but we vision it as the first step in the development of HTA for general neural networks. In the future, we will design a new way to apply it to more complex neural networks such as CNN and RNN so that the HTA can speed up the training process more efficiently. Since the structures of the CNNs and RNNs are very different from fully connected neural networks, we need to redesign the homotopy objective function in order to incorporate their structures, for instance, by including the dropout technique.

Data accessibility. This article does not contain any additional data.
Authors' contributions. Q.C. implemented the HTA method. W.H. designed the method and wrote the manuscript.
Competing interests. We declare we have no competing interests.
Funding. This research was supported by the American Heart Association (grant no. 17SDG33660722), the National Science Foundation (grant no. DMS-1818769) and the AiCure company.
Acknowledgements. We thank the reviewers for their careful and insightful comments to improve our manuscript.

References

1. Simonyan K, Zisserman A. 2014 Very deep convolutional networks for large-scale image recognition. Comput. Sci. (https://arxiv.org/abs/1409.1556)
2. He K, Zhang X, Ren S, Sun J. 2016 Deep residual learning for image recognition. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, NV, 26 June–1 July, pp. 770–778. Piscataway, NJ: IEEE.
3. Collobert R, Weston J. 2008 A unified architecture for natural language processing: deep neural networks with multitask learning. In Proc. 25th Int. Conf. on Machine Learning, Helsinki, Finland, 5–8 July, pp. 160–167. New York, NY: ACM.
4. Sainath T, Mohamed A, Kingsbury B, Ramabhadran B. 2013 Deep convolutional neural networks for LVCSR. In 2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada, 26–31 May, pp. 8614–8618. Piscataway, NJ: IEEE.
5. Markoff J. 2015 A learning advance in artificial intelligence rivals human abilities. New York Times, 10.
6. Cang Z, Mu L, Wei G. 2018 Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. PLoS Comput. Biol. 14, e1005929. (doi:10.1371/journal.pcbi.1005929)
7. Cang Z, Wei G. 2018 Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction. Int. J. Numer. Methods Biomed. Eng. 34, e2914. (doi:10.1002/cnm.2914)
8. Yin P, Zhang S, Lyu J, Osher S, Qi Y, Xin J. 2018 Binaryrelax: a relaxation approach for training deep neural networks with quantized weights. (https://arxiv.org/abs/1801.06313)
9. Yin P, Zhang S, Qi Y, Xin J. 2016 Quantization and training of low bit-width convolutional neural networks for object detection. (https://arxiv.org/abs/1612.06052)
10. Konda K, Königs A, Schulz H, Schulz D. 2012 Real time interaction with mobile robots using hand gestures. In Proc. 7th Annual ACM/IEEE Int. Conf. on Human-Robot Interaction, Boston, MA, 5–8 March, pp. 177–178. New York, NY: ACM.
11. Hammerla N, Halloran S, Ploetz T. 2016 Deep, convolutional, and recurrent models for human activity recognition using wearables. (https://arxiv.org/abs/1604.08880)
12. Maire F, Mejias L, Hodgson A. 2014 A convolutional neural network for automatic analysis of aerial imagery. In 2014 Int. Conf. on Digital Image Computing: Techniques and Applications (DICTA), Wollongong, Australia, 25–27 November, pp. 1–8. Piscataway, NJ: IEEE.
13. Bates D, Hauenstein J, Sommese A. 2011 Efficient path tracking methods. Numer. Algorithms 58, 451–459. (doi:10.1007/s11075-011-9463-8)
14. Bates D, Hauenstein J, Sommese A, Wampler C. 2008 Adaptive multiprecision path tracking. *SIAM J. Numer. Anal.* 46, 722–746. (doi:10.1137/06068862)
15. Morgan A, Sommese A. 1987 Computing all solutions to polynomial systems using homotopy continuation. *SIAM J. Numer. Anal.* 24, 115–138. (doi:10.1137/096-3003(87)90064-6)
16. Hao W, Hauenstein J, Hu B, Liu Y, Sommese A, Zhang YT. 2012 Bifurcation for a free boundary problem modeling the growth of a tumor with a necrotic core. *Nonlinear Anal. Real World Appl.* 13, 694–709. (doi:10.1016/j.nonrwa.2011.08.010)
17. Hao W, Hauenstein J, Hu B, Sommese A. 2011 A three-dimensional steady-state tumor system. *Appl. Math. Comput.* 218, 2661–2669. (doi:10.1016/j.amc.2011.08.006)
18. Wang Y, Hao W, Lin G. 2018 Two-level spectral methods for nonlinear elliptic equations with multiple solutions. *SIAM J. Sci. Comput.* 40, B1180–B1205. (doi:10.1137/17M113767X)
19. Hao W, Hauenstein J, Shu CW, Sommese A, Xu Z, Zhang YT. 2013 A homotopy method based on WENO schemes for solving steady state problems of hyperbolic conservation laws. *J. Comput. Phys.* 250, 332–346. (doi:10.1016/j.jcp.2013.05.008)
20. Hao W, Yang Y. 2018 Convergence of a homotopy finite element method for computing steady states of Burgers’ equation. *ESAIM: Math. Modell. Numer. Anal.*, 53, 1629–1644. (doi:10.1051/m2an/2018046)
21. Hao W. 2018 A homotopy method for parameter estimation of nonlinear differential equations with multiple optima. *J. Sci. Comput.* 74, 1314–1324. (doi:10.1007/s10915-017-0518-4)
22. Hao W, Harlim J. 2018 An equation-by-equation method for solving the multidimensional moment constrained maximum entropy problem. *Commun. Appl. Math. Comput. Sci.* 13, 189–214. (doi:10.2140/camcos)
23. Hao W, Nepomechie R, Sommese A. 2014 Singular solutions, repeated roots and completeness for higher-spin chains. *J. Stat. Mech: Theory Exp.* 2014, P03024. (doi:10.1088/1742-5468/2014/03/P03024)
24. Hao W, Crouser E, Friedman A. 2014 Mathematical model of sarcoidosis. *Proc. Natl Acad. Sci. USA* 111, 16 065–16 070. (doi:10.1073/pnas.1417789111)
25. Hao W, Friedman A. 2014 The LDL-HDL profile determines the risk of atherosclerosis: a mathematical model. *PLoS ONE* 9, e90497. (doi:10.1371/journal.pone.0090497)
26. Hornik K, Stinchcombe M, White H. 1989 Multilayer feedforward networks are universal approximators. *Neural Netw.* 2, 359–366. (doi:10.1016/0893-6080(89)90020-8)
27. Huang G, Chen L, Siew C. 2006 Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Netw.* 17, 879–892. (doi:10.1109/TNN.2006.875977)
28. Goodfellow I, Bengio Y, Courville A. 2016 *Deep learning*. Cambridge, MA: MIT Press. See http://www.deeplearningbook.org.
29. Vapnik V. 1999 An overview of statistical learning theory. *IEEE Trans. Neural Netw.* 10, 988–999. (doi:10.1109/72.788640)
30. Rao D, McMahan B. 2019 *Natural language processing with PyTorch: build intelligent language applications using deep learning*. Sebastopol, CA: O’Reilly Media, Inc.
31. Bernico M. 2014 Mathematical model of sarcoidosis. *Proc. Natl Acad. Sci. USA* 111, 16 065–16 070. (doi:10.1073/pnas.1417789111)
32. Bottou L, Curtis F, Nocedal J. 2018 Optimization methods for large-scale machine learning. *SIAM Rev.* 60, 223–311. (doi:10.1137/16M1080173)
33. Jensen J. 1906 Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Math.* 30, 175–193. (doi:10.1007/BF02418571)
34. Garcke J et al. 2006 *Sparse grid tutorial*. Canberra, Australia: Mathematical Sciences Institute, Australian National University, p. 7.
35. Smolyak S. 1963 Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Doklady Akademii Nauk*, vol. 148, pp. 1042–1045. Moscow, Russia: Russian Academy of Sciences.
36. Corbetta A, Muntean A, Vafayi K. 2015 Parameter estimation of social forces in pedestrian dynamics models via a probabilistic method. *Math. Biosci. Eng.* 12, 337–356. (doi:10.3934/mbi.2015.12.337)
37. Goodman I. 1983 Applications of fuzzy set theory to parameter estimation and tracking. Technical Report DTIC Document.
38. Haven K, Majda A, Abramov R. 2005 Quantifying predictability through information theory: small sample estimation in a non-Gaussian framework. *J. Comput. Phys.* 206, 334–362. (doi:10.1016/j.jcp.2004.12.008)
39. Masliev I, Somlyody L. 1994 Probabilistic methods for uncertainty analysis and parameter estimation for dissolved oxygen models. *Water Sci. Technol.* **30**, 99–108. (doi:10.2166/wst.1994.0033)

40. Krizhevsky A, Sutskever I, Hinton G. 2012 Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conf. on Neural Information Processing Systems 2012, Lake Tahoe, NV, 3–8 December*, pp. 1097–1105. Curran Associates Inc.

41. Rowley H, Baluja S, Kanade T. 1998 Neural network-based face detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **20**, 23–38. (doi:10.1109/34.655647)

42. Bojarski M *et al.*. 2016 End to end learning for self-driving cars. (https://arxiv.org/abs/1604.07316)

43. Chan A, Liang Z, Vasconcelos N. 2008 Privacy preserving crowd monitoring: counting people without people models or tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pp. 1–7. New York, NY: IEEE.

44. Krizhevsky A, Hinton G. 2009 Learning multiple layers of features from tiny images, vol. 1, no. 4. Technical report. University of Toronto.