Constructing the Basis Path Set by Eliminating the Path Dependency∗

ZHU Juanping · MENG Qi · CHEN Wei · WANG Yue · MA Zhiming

DOI: 10.1007/s11424-022-0208-7
Received: 31 August 2020 / Revised: 5 July 2021
© The Editorial Office of JSSC & Springer-Verlag GmbH Germany 2022

Abstract The newly appeared G-SGD algorithm can only heuristically find the basis path set in a simple neural network, so its generalization to a more practical network is hindered. From the perspective of graph theory, the BasisPathSetSearching problem is formulated to find the basis path set in a complicated fully connected neural network. This paper proposes algorithm DEAH to hierarchically solve the BasisPathSetSearching problem by eliminating the path dependencies. For this purpose, the authors discover the underlying cause of the path dependency between two independent substructures. The path subdivision chain is proposed to effectively eliminate the path dependency, both inside the chain and between chains. The theoretical proofs and the analysis of time complexity are presented for Algorithm DEAH. This paper therefore provides one methodology to find the basis path set in a general and practical neural network.

Keywords Basis path, neural network, path dependency, path subdivision chain, substructure path.

1 Introduction

Neural networks with ReLU activation function have been developed for a variety of tasks, such as distribution estimation in statistics, machine translation and language modeling, etc.[1, 2]. The specific way that an ReLU neural network is trained in terms of weights in an appropriate space, affects the network performance and efficiency. In order to handle the mismatch during the optimization of neural networks in the positively scale-invariant space[3, 4], Meng, et al.[5] proposed to optimize the values of basis path set in the G-space of ReLU neural networks by...
the stochastic gradient descent algorithm (SGD)\cite{9}. According to the experimental results, this novel $G$-SGD algorithm\cite{5} with attractively low dimensionality of basis path set significantly outperformed conventional SGD algorithm. In addition, the performance superiority was recently approved in language modeling and machine translation, by adopting the basis path set in transformer networks\cite{2}.

Despite the experimental success, the $G$-SGD algorithm needs further theoretical investigation on how the inner mechanism of basis paths works in neural networks. For instance, how to analyze the structure relationship between the basis path set and the network and how the different network structure affects the algorithm searching for basis path set. On the other hand, the generalization of $G$-SGD algorithm is currently hindered, because the implementation of $G$-SGD algorithm\cite{5} depends on the structure of the neural network: The current $G$-SGD algorithm can only heuristically find the basis path set in a simple fully connected network, which demands (a) the widths of all layers to be the same, and (b) not to be any edge-skipping over layers. However, there are varieties of structures in practical networks (such as ResNet and DenseNet) with different combinations of edge-skipping over layers and non-uniform layer width. It would be desirable for the $G$-SGD algorithm to be generalized to more practical neural networks, so that it could be applied widely in applications.

Recently many efforts have been invested into explaining and understanding the overwhelming success of neural network learning methods\cite{7–11}. We resort to graph theory for the theoretical support, to understand basis path set similar to the way that neural networks are explained in various perspectives, such as group operations and functional\cite{12–14}. In the perspective of graph theory, Zhu, et al.\cite{15} defined the basis path and proposed an hierarchical algorithm to find basis path set in each independent substructure. This hierarchical algorithm\cite{15} can handle the network with non-uniform layer width and edge-skipping but requires that there are no shared layers between any two independent substructure paths, which is a restrictive assumption for practical neural networks.

This paper considers the graph theory based BasisPathSetSearching problem of how to find a maximal independent path set in a regular fully connected neural network with non-uniform layer widths and edge-skipping, without the restriction assumed in \cite{15}. First, we investigate the combinatoric possibility that two independent substructures will bring up the path dependency, when they share some common layers. Then we propose Algorithm DEAH to hierarchically solve the BasisPathSetSearching problem by eliminating the path dependency. This hierarchical idea decomposes the complicated network into maximally independent substructures and finds a basis path set for each independent substructure in parallel. Then the algorithm eliminates the paths which would cause the path dependency when we combine these basis path sets together. To avoid the enumeration of all possible substructure path pairs with shared layers, we take advantage of path subdivision chains in the algorithm design. Lemmas and theorems are provided to guarantee that the algorithm can solve BasisPathSetSearching problem in polynomial time.

The contributions of this paper are the following: (i) The paper formulates the problem of finding the basis path set in more practical neural networks as the BasisPathSetSearching problem.
problem from graph theory perspective, discovers the structure relationship of two independent substructures, and provides one polynomial algorithm; (ii) This paper provides one methodology to find the basis path set in more general neural networks, and the research can offer the theoretical and algorithmic support for the application of G-SGD algorithm in more practical scenarios.

2 Preliminary

A regular fully connected neural network is an $L$-layer multi-layer perceptron with weighted edges that can skip over different layers\cite{19}. We denote the $i$-th node in the $l$-th layer as $O_i^l$ and the node set of the $l$-th layer as $P_l$. $(O_i^l, O_{i+j}^l)$ is denoted as the directed edge from layer $l$ to layer $l + j$, which can skip layer $l + 1$ till layer $l + j - 1$, where $1 \leq j \leq L - l$, $1 \leq i \leq |O_l|$ and $1 \leq i' \leq |O_{l+j}|$. Within the same layers, the edges are fully connected. Since graph theory provides one effective platform to investigate the paths in the graph both intuitively and theoretically\cite{16,17,18}, the paper would interpret a neural network as a neural network $G = (V, E)$, where the finite node set $V = O_1 \cup \cdots \cup O_l \cdots \cup O_L$ comprising all nodes in neural network and finite edge set $E = \{(O_i^l, O_{i+j}^l)|0 \leq l \leq L - 1, 1 \leq j \leq L - l\}$ consisting of all directed fully connected edges between different layers. $m = |E|$ is the number of edges in graph $G$. Network $G$ satisfies that there must be directed edge $(O_i^l, O_{i+j}^l)$ for all $1 \leq i \leq |O_l|$ and $1 \leq i' \leq |O_{l+j}|$, if there is one edge between layer $l$ and layer $l + j$ for $1 \leq j \leq L - l$. Let set $P = \{(O_0^1, O_{i_1}^l, \cdots, O_{j_1}^l, \cdots, O_{i_L}^L)|0 < i_1 \cdots < j_1 \cdots < L\}$ consist of all paths from the input layer to the output layer in network $G^{16,17}$, where $O_0^i$ is denoted as some node without specified position in the $l$-th layer for simplicity.

**Definition 2.1** (see [15]) (path addition to a graph) Given a graph $H$ and a path $p$, we denote the path addition by $H + p$ with $V(H + p) = V(H) \cup V(p)$ and $E(H + p)$ being the disjoint union of $E(G)$ and $E(p)$ (Parallel edges may arise).

**Definition 2.2** (see [15]) (path removal from a graph) Given a graph $H$ and one path $p \subseteq E(H)$, the removal of the path $p$ from the graph $H$ is defined as $H - p$ with $E(H - p) = E(H) \setminus E(p)$ and $V(H - p) = V(H) \setminus \{v \in V(H)|v$ is an isolated vertex after $E(H) \setminus E(p)\}$.

**Definition 2.3** (see [15]) (structure path) Given network $G$, if all paths in $P$ from the input layer to the output layer pass through the same layers consecutively, any path $p \in P$ can be called the structure path of network $G$, since it can express the structure information of $G$.

**Definition 2.4** (see [15]) (independent path set) Given path set $B \subseteq P$ of neural network $G$, if there exist one path $p \in B$ and another path $q \in B \setminus \{p\}$ such that we can reach path $p$ from path $q$ through finite steps of path addition and path removal within $B$, we call path set $B$ independent. Otherwise, we call path set $B$ dependent.

**Definition 2.5** (see [15]) (maximal independent path set) A path set $B \subseteq P$ of network $G$ is maximally independent, if including any other path $p^* \in P \setminus B$ would make $B \cup p^*$ dependent. Hence, a basis path set $B$ of network $G$ is a maximal independent subset of $P$.

In Figure 1, path $p_1 = (O_0^1, O_1^1, O_2^1)$, path $p_2 = (O_0^2, O_1^1, O_2^1)$, $p_3 = (O_0^3, O_1^1, O_2^1)$, and...
\( p_4 = (O_0, O_1, O_2) \). Let path set \( B = \{p_1, p_2, p_3\} \), then \( B \) is an independent path set. However, if \( B = \{p_1, p_2, p_3, p_4\} \), \( B \) is a dependent path set for \( p_4 = p_1 + p_2 + p_3 \), which means we can reach path \( p_4 \) from \( p_1 \in B \). Obviously, \( \{p_1, p_2, p_3\} \) is the maximal independent path set in this graph.

**Definition 2.6** (substructure path) Given neural network \( G \) and induced sub-graph \( G' \) with \( V(G') = (O^0, O^1, \cdots, O^{i'}, \cdots, O^L) \), where \( 1 \leq i' \leq L - 1 \). Let \( P(G') \) be the path set from the input layer to the output layer in \( G' \). If all paths in \( P(G') \) from the input to the output pass through the same layers homogeneously, then any path \( p \in P(G') \) can be called the substructure path of network \( G \). Substructure path \( p \) can express the structure information about sub-graph \( G' \) in network \( G \).

As shown in Figure 1, path \( p_1 = (O^0, O^1, O^2) \) is a substructure path in this network. So is path \( p_2, p_3 \) or \( p_4 \).

**Definition 2.7** (substructure path set) Define one induced sub-graph \( G^S = (V^S, E^S) \) of neural network \( G \), where \( V^S = \{ \) random \( O^l_i, \in O^l | l = 0, 1, \cdots, L \} \) and \( E^S = \{ (O^l_i, O^k_j) | E | O^l_i, \in V^S, O^k_j, \in V^S, 0 \leq l < k \leq L \} \). By breath first search, we can get all substructure paths starting from node \( O^0 \) to node \( O^k \) in \( G^S \). Denote this substructure path set as \( P^S \), which can represent all substructure information of network \( G \).

**Definition 2.8** (maximal independent substructure path set) Given neural network \( G \) and substructure path set \( P^S \), the substructure path set \( P^S_{ind} \subset P^S \) is called dependent, if there is one substructure path \( p \in P^S_{ind} \) and another path \( q \in P^S_{ind} \setminus \{p\} \) such that we can reach path \( p \) from path \( q \) through finite steps of path addition and path removal within \( P^S_{ind} \). Otherwise, we call \( P^S_{ind} \) independent. A substructure path set \( P^S_{ind} \) of \( P^S \) is maximally independent, if including any other path \( p^* \in P^S \setminus P^S_{ind} \) would make \( P^S_{ind} \cup p^* \) dependent.

Actually subgraph \( G^S \) is a simplified network with only one node in each layer. There possibly exists one special structure relationship between two independent substructure paths, i.e., path subdivision, which is the adaptation of graph subdivision[19–21] to the path.

**Definition 2.9** (path subdivision) Given a substructure path set \( P^S \) and one substructure path \( p \in P^S \) with one edge \( e = (u, v) \in E(p) \). If there exists one substructure path \( p' \in P^S \) with sub-path \( (u, x_1, x_2, \cdots, x_k, v) \), this sub-path is called the edge subdivision of edge

![Figure 1 Example of the independent path set](image)
We call $e$ the subdivided edge and $x_1, x_2, \cdots, x_k$ the subdivision vertices. Furthermore, if $p - e = p' - (u, x_1, x_2, \cdots, x_k, v)$, we call path $p'$ the path subdivision of path $p$ and path $p$ the subdivided path. Especially, we denote by $p'$ the path obtained from $p$ by subdividing the edges $e_1, \cdots, e_t, \cdots, e_T$, where each edge $e_t \in E(p)$ is subdivided once for $t \in \{1, 2, \cdots, T\}$.

There is no edge subdivision in $P_S$ in Figure 2(a). In Figure 2(b), edge $(O^0, O^2)$ is a subdivided edge, because there exists sub-path $(O^0, O^1, O^2)$ between $O^0$ and $O^2$. Let $p_1 = (O^0, O^1, O^2, O^3, O^4)$, $p_2 = (O^0, O^2, O^3, O^4)$, $p_3 = (O^0, O^1, O^2, O^4)$ and $p_4 = (O^0, O^2, O^4)$. Substructure path $p_1$ is the path subdivision of paths $p_2, p_3, \text{ and } p_4$. Paths $p_2$ and $p_3$ are the path subdivision of path $p_1$ but $p_2$ and $p_3$ are not path subdivision of each other. Note that we use the layer to represent the random node in the layer when discussing the substructure path, which is different from the regular path expression.

Figure 2  Example of the path subdivision

**Definition 2.10**  (underlying substructure path) Given the substructure path set $P_S$, subset $P' \subset P_S$ and one substructure path $p_0 \in P'$, we call $p_0$ the underlying substructure path of $P'$ if $p_0$ is the path subdivision of any substructure path $p \in P' \setminus \{p_0\}$.

### 3 Problem Statement

#### 3.1 BasisPathSetSearching Problem

To solve the challenge of how to find basis path set in a more general neural network mentioned in section Introduction, this paper focuses on BasisPathSetSearching Problem, which is formulated as follows:

Given a regular fully connected neural network $G = (V, E)$, where $|O^l|$ is not necessarily the same for all $0 \leq l \leq L$ and there must be at least one path from $O^0$ to $O^L$. We aim to find one maximal independent path set (basis path set) $B$ in graph $G$.

In other words, the regular fully connected neural network $G$ can be a network with unbalanced layers (the widths of different layers are not equal) and with edge-skipping over different layers.
3.2 Path Dependency Between Two Independent Substructures

In graph theory, any path $p \in P$ in network $G$ from the input layer to the output layer can be represented by the basis path set $B$ with smaller cardinality. One hierarchical algorithm has been proposed to decompose graph $G$ into maximal independent substructures $P_{ind}^S$, where each sub-graph $G_r$ induced from $p_r \in P_{ind}^S$ can be treated as a fully connected neural network without any edge-skipping. Here $G_r$ is the sub-graph of $G$ by taking all edges from $G$ with the same layers as substructure path $p_r$, and Algorithm Subroutine (in Appendix) can find basis path set $B_r$ accordingly in $G_r$. It is well-known that the hierarchical idea usually decomposes the complicated combinatorial optimization problem into several independent and simpler sub-optimization problems and solves each sub-optimization problem separately. Since $P_{ind}^S$ is the maximal independent substructure set and each independent substructure $p_r \in P_{ind}^S$ has unique structure, intuitively we would consider to simply combine these basis path sets $B_r$ together to form basis path set $B$ for network $G$. However, the basis paths from two independent substructures $p_r$ and $p_s$ in network $G$ couldn’t guarantee to be path dependent, though their structures are unique. The underlying cause of path dependency is $E(p_r) \cap E(p_s) \neq \emptyset$ ($r \neq s$), i.e., there exist shared edges (or layers) between different substructure paths $p_r$ and $p_s$. Assume $B_r$ is the basis path set from the induced sub-graph $G_r$ in terms of $p_r$ and $B_s$ is the basis path set from $G_s$ in terms of $p_s$. The shared edges offer the chance for basis paths from $B_r$ and $B_s$ to exchange the locations of the shared layer $s$ between two substructures and to cancel same unshared layers within the unique substructure. There are three typical cases to illustrate this combinatoric possibility, and we also notice that the basis paths from each substructure must appear in pairs to cancel the unique unshared layers.

**Case 1** In Figure 3, substructure paths $p_1$ and $p_2$ are independent, and $p_1$ and $p_2$ share layers of $S$ and $S'$. $p_2$ is the path subdivision of $p_1$ in the layers of $I$. Given basis paths $p_{1,1}, p_{1,2} \in B_1$ and $p_{2,1}, p_{2,2} \in B_2$, any path can be represented by the remaining three paths, such as $p_{1,1} = p_{2,1} - p_{2,2} + p_{1,2}$. The same sub-path $I'$ in $p_{1,1}$ and $p_{1,2}$ in the unshared layers can be canceled inside substructure $p_{1,1}$, and the same sub-path $I$ in $p_{2,1}$ and $p_{2,2}$ can be canceled within substructure $p_{2,2}$ too. In the shared layers, sub-path $S$ of $p_{1,1}$ and $p_{2,1}$ can be swapped between independent substructures, and so can sub-path $S'$ of $p_{1,2}$ and $p_{2,2}$.

![Figure 3](image.png)

**Figure 3** Case 1 of the path dependency

**Case 2** In Figure 4, independent substructure paths $p_2$ and $p_3$ share layers at $S$ and $S'$. $p_2$ and $p_3$ have the edge subdivision in each other but neither of them is the path subdivision.
of each other. Given basis paths $p_{2,1}, p_{2,2} \in B_2$ and $p_{3,1}, p_{3,2} \in B_3$, sub-path $S'$ in $p_{2,1}$ and $p_{3,1}$ and $S$ in $p_{2,2}$ and $p_{3,2}$ can be swapped between substructures. For the unshared layers, sub-path $I' + J'$ in $p_{2,1}$ and $p_{2,2}$ and $I + J$ in $p_{3,1}$ and $p_{3,2}$ can be canceled within the substructure.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Fig5.png}
\caption{Case 2 of the path dependency}
\end{figure}

**Case 3** In Figure 5, substructure paths $p_2$ and $p_1$ have layers of $S$ and $S'$ in common but have no edge subdivision in each other. In the unshared layers, same sub-path $I$ in $p_{1,1}$ and $p_{1,2}$ can be canceled from each other, and sub-path $I'$ in $p_{2,1}$ and $p_{2,2}$ can be canceled too. In the shared layers, sub-path $S$ in $p_{1,1}$ and $p_{2,1}$ and $S'$ in $p_{1,2}$ and $p_{2,2}$ can be swapped.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Fig6.png}
\caption{Case 3 of the path dependency}
\end{figure}

**Property 1** Given independent substructure paths $p_r$ and $p_s$ ($r \neq s$) in network $G$, if $E(p_r) \cap E(p_s) \neq \emptyset$, then the path set $B_r \cup B_s$ is not path independent.

**Claim 1** Given two independent substructure paths $p_1$ and $p_2$ with the shared layers $S^* = E(p_1) \cap E(p_2)$ in network $G$, basis paths $p_{1,1}, p_{1,2} \in B_1$ with $E(S) \subset E(p_{1,1})$ and $E(S') \subset E(p_{1,2})$ at the shared layers $S^*$. If $p_{1,1} - \sum_{e \in S} e = p_{1,2} - \sum_{e \in S'} e$, then there must exist basis paths $p_{2,1}, p_{2,2} \in B_2$ such that $E(S) \subset E(p_{2,1})$ and $E(S') \subset E(p_{2,2})$ at the layers of $S^*$ and $p_{2,1} - \sum_{e \in S} e = p_{2,2} - \sum_{e \in S'} e$.

**Proof** When unshared layers $I$ appear at the top of shared layers (in Figure 5), let $I = p_{1,1} - \sum_{e \in S} e = p_{1,2} - \sum_{e \in S'} e$. According to the construction rule for the basis path set when applying Algorithm Subroutine\cite{18}, the lowest edge in $I$ must be a direct path, because it accepts two different sub-paths $S$ and $S'$ from its incident node $O_j^*$. Moreover, there must exist one direct path starting from $O_j^*$ in $B_2$ regarding $p_2$. Pick up one sub-path starting with
this direct path, denoted as \( I' \). Here we demand the same rule for selecting the random paths in Algorithm Subroutine. So sub-paths \( S \) and \( S' \) ending at \( O_j^2 \) must be included in some basis paths of \( B_2 \) and \( I' \) will accept both \( S \) and \( S' \) in \( p_{2,1} \) and \( p_{2,2} \) as shown in Figure 5.

If the shared layers is at the top of unshared layers of \( I \) (in Figure 3), \( S \) and \( S' \) are two different sub-paths starting from \( O_j^2 \) in \( B_1 \). Applying the same rule to select random paths, \( S \) and \( S' \) must exist as layers of two basis paths in \( B_2 \). We can pick up the direct path ending at \( O_j^2 \) which can be concatenated by \( S \) and \( S' \) at \( O_j^2 \).

Other locations of shared layers are the combination of these two cases.

Claim 2 The path dependency couldn’t happen among more than two independent substructures, if we eliminate the paths which cause the path dependency when combing the basis path sets of two independent substructures.

Proof The analysis of the path dependency indicates that basis paths from one substructure must appear in pairs to cancel unshared layers for the structural uniqueness. Suppose the path dependency happens among three independent substructures, i.e., \( p_1, p_2, \) and \( p_3 \) as shown in Figure 6. Assume \( p_{1,1} - p_{1,2} + p_{2,1} - p_{2,2} = p_{3,1} - p_{3,2} \), where \( p_{1,1}, p_{1,2} \in B_1, p_{2,1}, p_{2,2} \in B_2, \) and \( p_{3,1}, p_{3,2} \in B_3 \). The common layers \( E(S) \subset E(p_{1,1}) \) and \( E(S) \subset E(p_{2,1}) \) must appear in pairs for swapping. The same for \( E(S') \subset E(p_{2,2}) \) and \( E(S') \subset E(p_{3,1}) \), and \( E(S'') \subset E(p_{1,2}) \) and \( E(S'') \subset E(p_{3,2}) \). The unshared layers \( p_{1,1} = \sum_{e \in S} e \) and \( p_{1,2} = \sum_{e \in S'} e \) must be the same to cancel the unique structure. So do \( p_{2,1} = \sum_{e \in S} e \) and \( p_{2,2} = \sum_{e \in S'} e \) pair and \( p_{3,1} = \sum_{e \in S''} e \) and \( p_{3,2} = \sum_{e \in S''} e \) pair. When considering to eliminate the paths which cause path dependency between two independent substructures, we must discard at least one basis path from the corresponding pair. For example, if \( p_{1,1} \) and \( p_{3,2} \) stay, \( p_{1,2} \) must be discarded when considering \( p_1 \) and \( p_3 \). Therefore, it is impossible to either make swapping for the shared layers or cancel the unshared layers for the basis path from the third substructure, i.e., \( p_2 \). So, the assumption couldn’t happen and Claim 2 holds.

\[
\begin{align*}
\text{Figure 6} & \quad \text{Elimination of the path dependency}
\end{align*}
\]

Claim 2 indicates that we only need to consider the path dependency between any two independent substructures. However, it would be expensive to enumerate all two substructure path pairs with the shared layers. In Case 2, the maximal independent substructure path set could be \( \{p_0, p_1, p_2, p_3\} \) in Figure 4(a) and it can also be \( \{p_0, p_4, p_5, p_6\} \) as in Figure 4(c), where \( p_5 \) is the path subdivision of \( p_4 \), \( p_6 \) is the path subdivision of \( p_5 \) and the underlying substructure path \( p_0 \) is the path subdivision of \( p_4, p_5 \) and \( p_6 \). Consequently, \( \{p_0, p_4, p_5, p_6\} \) forms one chain. Motivated by this, the path subdivision chain is proposed to avoid the complicated enumeration.
Definition 3.1 (path subdivision set) Given the substructure path subset $P' \subset P^S$ of network $G$, the path subdivision set $U_r$ of substructure path $p_r \in P'$ is defined as $\{p \in P' | p$ is the subdivision of $p_r\}$. According to the set containment relationship, we can get $T$ path subdivision chains over $\{U_r | r = 1, 2, \ldots, |P'|\}$, i.e., the $t$-th chain $U_{t_1} \supset U_{t_2} \cdots \supset U_{t_j} \cdots \supset U_{t_{st}}$, where $U_{t_j}$ is the $j$-th set of the $t$-th chain and $\sum_{t=1}^{T} s_t \geq |P'|$.

Hence, substructure path $p_r$ corresponds to basis path set $B_r$ in its induced sub-graph $G_r$ and path subdivision set $U_r$. Figure 4(c) can form one path subdivision chain $U_1 \supset U_5 \supset U_6 \supset U_0$ but Figure 4(a) needs 3 chains, i.e., $U_1 \supset U_0$, $U_2 \supset U_0$ and $U_3 \supset U_0$, though $p_1$, $p_2$ and $p_3$ have subdivision layers in each other. Obviously, path subdivision set $U_0 = \emptyset$.

4 Dependency Eliminating Algorithm Based on Hierarchical Idea (DEAH)

Algorithm HBPS\cite{15} initiated an inspiring hierarchical idea to decompose the complicated layered graph $G$ into several independent substructures, but the restriction is that there must be no shared layers between any two maximal independent substructures. To solve BasisPathSet-Searching problem in more practical network $G$, Algorithm DEAH is proposed to overcome this restriction by eliminating the path dependency. In order to avoid the enumeration of all independent substructure path pairs with some possible shared layers, Algorithm DEAH tactically manipulates multiple path subdivision chains among the maximal independent substructure path set. The crucial point is to stretch the path subdivision chain as long as possible, as shown in Figure 4(c). Therefore, Algorithm DEAH tries to avoid Case 2 (in Figure 4(a)) to happen when searching the maximal independent substructure path set. According to Claim 2, the algorithm only considers the elimination of the path dependency between: a) The neighboring substructure pairs along each path subdivision chain, and b) the first substructure path in one chain and the last substructure path in another chain which shares common layers with this first substructure path.

To find the maximal independent substructure path set $P^S_{ind}$ from substructure path set $P^S$, we borrow the concept of adjacent matrix from graph theory\cite{16–18} and define adjacent matrix $M_r$ for path $p_r \in P^S$ as

$$M_r(j,l) = \begin{cases} 1, & \text{if there is an edge from layer } j \text{ to layer } l, \\ 0, & \text{otherwise,} \end{cases}$$

where $j = 0, 1, \ldots, L$ and $l = 0, 1, \ldots, L$. Then, each $p_r$ is expressed as a 0-1 substructure path vector $\alpha_r = [M_r(0,:), \cdots, M_r(j,:), \cdots, M_r(L,:)]$. Since $G^S$ is the simplest network with only one node in each layer, it is easy to calculate the maximal independent path set $P^S_{ind}$ from $P^S$ by implementing numerical linear algebra method, but it is infeasible to get basis path set $B$ in $P$ in the same way. To find the path subdivision and edge subdivision between two independent
substructure paths, straightforward $(L+1)$-dimensional incident vector $\beta_r$ is defined as

$$\beta_r(l) = \begin{cases} 
1, & \text{if } p_r \text{ passes through layer } l, \\
0, & \text{otherwise},
\end{cases}$$

where $l = 0, 1, \ldots , L$. Let $X = \beta_r - \beta_t$. If $X$ contains 1 and $-1$, then $p_r$ and $p_t$ are not path subdivision to each other. If $X$ contains only 0 and $-1$, $p_r$ is the path subdivision of $p_t$.

Based on the definition of incident vector $\beta_r$, Algorithm SDVChain, called by Algorithm DEAH, is designed to eliminate the path dependency between every neighboring independent substructure pair along the path subdivision chain and can also be called to eliminate the path dependency between any two independent substructures.

\textbf{Algorithm 1 SDVChain}($\{B_r\}^*_r=1, \{B_r\}^*_r=1, \{P_r\}^*_r=1, \{\beta_l\}^*_r=1, B_0, p_0, \beta_0, G$)

\begin{algorithmic}
\State \textbf{Input} Shrunk set $B'_r$, basis path set $B_r$, and $\beta_r$, basis path set $B_0$ and $\beta_0$ in network $G$
\State \textbf{Output} Updated shrunk set $\{B'_r\}^*_r=2$ and discarded path set $D'_0$.
\State Let $B'_{r+1} = B_0, p_{r+1} = p_0$ and $\beta_{r+1} = \beta_0$.
\For {$r = 1 : s$ do}
\State Search shared layers $\{(O'_n, O''_n)\}^k_{k=1}$ for $p_r$ and $p_{r+1}$, according to $\beta_r$ and $\beta_{r+1}$. Find unshared layers $\{Ep_{r,j}\}$ in $B_r$. Let $\text{DiscardPath} = \emptyset$.
\State Find shared layers set $UCP_j$ in $B_r$ with $Ep_{r,j}$ as the unshared layers. Let $UCP'_j$ be the element in $UCP_j$ with the most frequent occurrence./ Find the most frequent edge set in the shared layers./
\State Find the unshared layers set $IEp_{j}$ in $B'_r$ with the shared layers from $UCP_j$. Delete the element with frequency $< 2$ from $IEp_{j}$. /No dependency between path pair without shared layers./
\State Let $\text{DiscardPath} = \text{DiscardPath} \cup \{ p_{r+1,i} \in B'_{r+1} | p_{r+1,i}$'s unshared layers are from $IEp_{j}, i \in \{1,2,\cdots,|IEp_{j}|\}$. /Discard the path according to the rule./
\EndFor
\State Update $B'_{r+1} = B'_{r+1} \setminus \text{DiscardPath}$ and set $D'_{r+1} = \text{DiscardPath}$.
\State Set $D'_0 = D_{s+1}$.
\end{algorithmic}

\textbf{Lemma 4.1} Given path subdivision chain $U_{t_1} \supset U_{t_2} \cdots \supset U_{t_j} \cdots \supset U_{t_{s+1}} \supset U_0$ in network $G$, where path subdivision set $U_{t_j}$ and original basis path set $B_{t_j}$ correspond to $p_{t_j} \in P^S_{\text{ind}}$. $p_0$ is the underlying substructure path for $P^S_{\text{ind}}$. Let the input $B'_{t_j}$ be original $B_{t_j}$ $(j = 1, 2, \cdots , s_{t})$ and $B'_0$ be $B_0$. Then, $B_{t_1} \cup B'_{t_2} \cup \cdots B'_{t_j} \cup B'_{s_{t}+1}$ output from Algorithm SDVChain is path independent, where $B'_{t_j}$ is the shrunk set of $B_{t_j}$ after the path discarding.

\textbf{Proof} We use induction on the $j$-th subdivided substructure path to prove that $B_{t_1} \cup B'_{t_2} \cup \cdots B'_{t_j} \cup \cdots B'_{t_{s_{t}+1}}$ is path independent.

Basis step: $j = 1$ and $U_{t_1} \supset U_{t_2}$. Algorithm SDVChain first picks $UCP'_j$ with the most frequent occurrence in $UCP_j$ for each unique $Ep_{t_{1,j}}$ of $B_{t_1}$. The trick here is that the edge between every two common layers with the highest frequency must be the direct path according to the construction rule[15] for basis paths. According to Claim 1, only the paths with multiple
repetitions at unshared layers can cause path dependency. The algorithm then searches the unshared layers set \( IEP_j \) in \( B_{t_2} \) which has the shared layers from \( UCP_j \) and discards the element with frequency less than 2 from \( IEP_j \). Next, the algorithm outputs \( B'_{t_2} \) by discarding the paths from \( B_{t_2} \) whose unshared layers are from \( IEP_j \) and shared layers are not \( UCP_j^\ast \). Hence, any path \( p \in B_{t_1} \cup B'_{t_2} \) couldn’t be represented by \( B_{t_1} \cup B'_{t_2} \setminus \{p\} \) and any path \( pt_{2,u} \in B_{t_2} \setminus B'_{t_2} \) can be represented by \( B_{t_1} \cup B'_{t_2} \). So, \( B_{t_1} \cup B'_{t_2} \) is path independent for the basic step.

Induction step: \( j \geq 2 \). Suppose that \( B_{t_1} \cup B'_{t_2} \cup \cdots \cup B'_{t_j} \) is an independent path set and any path \( p \in B_{t_{j-1}} \setminus B'_{t_j} \) can be represented by \( B_{t_{j-1}} \cup B'_{t_j} \setminus \{p\} \). Let \( B'_{t_{j+1}} \) be the shrunk set of \( B_{t_{j+1}} \) based on \( B_{t_j} \) after Algorithm SDVChain. We prove that any path \( p \in B_{t_1} \cup B'_{t_2} \cup \cdots \cup B'_{t_j} \cup B'_{t_{j+1}} \) couldn’t be represented by \( B_{t_1} \cup B'_{t_2} \cup \cdots \cup B'_{t_j} \cup B'_{t_{j+1}} \setminus \{p\} \). Assume path \( p' \in B_{t_1} \cup B'_{t_2} \cup \cdots \cup B'_{t_j} \cup B'_{t_{j+1}} \) can be represented by \( B_{t_1} \cup B'_{t_2} \cup \cdots \cup B'_{t_j} \cup B'_{t_{j+1}} \setminus \{p'\} \). According to Claim 2, consider the simplest form that \( p'' = pt_{j+1,1} - pt_{j+1,2} + p'' \), where \( pt_{j+1,1} \in B'_{t_{j+1}} \) and \( pt_{j+1,2} \in B'_{t_{j+1}} \) such that \( p'' \in B_{t_m} \) and \( p'' \in B'_{t_m} \) with \( m \leq j - 1 \). From basic step, any path \( p \in B_{t_j} \cup B'_{t_{j+1}} \) can be represented by \( B_{t_j} \cup B'_{t_{j+1}} \setminus \{p\} \), so it is obvious that \( p', p'' \notin B'_{t_{j+1}} \) and \( p', p'' \notin B'_{t_{j+1}} \). In the specific chain \( U_{t_m} \supset U_{t_j} \supset U_{t_{j+1}} \), substructure path \( pt_{j+2} \) is the path subdivision of \( pt_{j+2} \), and \( pt_{j+2} \) is the path subdivision of \( pt_{m} \). As shown in Figure 7, \( pt_{j+1,1} \) and \( pt_{j+1,2} \) must have same sub-path \( I \) and paths \( p' \) and \( p'' \) must have \( J \) to cancel the unique sub-paths, since \( pt_{j+2} \) is the path subdivision of \( pt_{m} \) at the layers of \( J \). For the shared layers of \( pt_{m} \) and \( pt_{j+1} \), \( p' \) and \( pt_{j+1,2} \) share sub-path \( S \) and \( p'' \) and \( pt_{j+1,2} \) share \( S' \). \( pt_{j} \) should be the path subdivision of \( pt_{m} \) in some part of \( J \), i.e., the dashed lowest part \( J^\ast \) of \( J \). Hence, the dashed parts \( I^\ast \) in \( J^\ast \) of \( pt_{j} \) subdivided by \( pt_{j+1,1} \) and \( pt_{j+1,2} \) are the same. However, the shared layers such as \( I - I^\ast + S \) in \( pt_{j+1,1} \) and \( I - I^\ast + S' \) in \( pt_{j+1,2} \) are different as shown in Figure 7. This contradicts Algorithm SDVChain which keeps only the most frequent shared layers from \( B_{t_j} \) to \( B'_{t_{j+1}} \).

Repeat this induction step till \( j = s_t \).

![Figure 7](image-url)  

**Figure 7** Edge operation in subdivided unshared layers

The crucial point of Lemma 4.1 is that we can construct path subdivision chain to avoid the enumeration within the chain, so we only have to successively eliminate the path dependency between the neighboring substructures along the chain. Furthermore, if \( U_{t_j} \) in the \( t \)-th chain belongs to more than one chain, i.e., \( U_{t_j} \) is the direct path subdivision of \( U_{t_j^\prime} (U_{t_j^\prime} \supset U_{t_j}) \) in the \( t' \)-th chain, Algorithm SDVChain will shrink \( B'_{t_j} \) iteratively by discarding the paths from
dent substructures, which is essentially different from Algorithm HBPS [15].

\[ B'_{i_{j}} \] based on \( B_{i} \). In this case, it is trivial that path set
\( B_{t_{1}} \cup B'_{i_{j}} \cup \cdots B'_{i_{j}} \cdots \cup B'_{i_{j}} \) is an independent
path set and any path \( p \in B_{t_{j}} \setminus B'_{i_{j}} \) can be represented by either
\( B_{t_{j-1}} \cup B'_{i_{j}} \) or \( B_{t_{j}} \cup B'_{i_{j}} \).

**Lemma 4.2** Given the first set \( U_{t_{1}} \) in the \( t \)-th chain and the set \( U'_{i_{j}} \) with \( p_{i_{j}} \) as the
last substructure path in the \( i \)-th chain to share layers with \( p_{i_{j}}(t' \neq t \text{ and } i' \neq i). \) Let the inputs
of Algorithm SDVChain be \( B'_{i_{j}} \), \( B'_{i_{j}} \), \( p_{i_{j}} \), \( p_{i_{j}} \), \( B_{t_{1}} \), \( p_{t_{1}} \), \( B_{t_{1}} \).
Algorithm SDVChain outputs the shrunk set \( B'_{i_{j}} \) by deleting the paths from \( B_{t_{1}} \) based on \( U'_{i_{j}} \). Then \( B'_{i_{j}} \cup B'_{i_{j}} \) is path
independent, for \( i \in \{1, 2, \cdots , i_{j}\} \).

**Proof** Suppose the \( t \)-th path subdivision chain be \( U'_{i_{j}} \supset \cdots U'_{i_{j-1}} \supset U'_{i_{j}} \supset \cdots \supset U'_{i_{j}} \).
Assume there exist paths \( p_{i_{j}}, p_{i_{j}} \in B'_{i_{j}} \) (i.e. \( i_{j} \)) and paths \( p_{t_{1}}, p_{t_{2}} \in B_{t_{1}} \) such that
\( p_{i_{j}} > p_{i_{j}} \) and \( p_{t_{1}} \). According to Claim 1, there must exist \( p_{i_{j}} \) and \( p_{i_{j}} \) in \( B'_{i_{j}} \),
\( i_{j} > p_{t_{1}} \) such that \( p_{i_{j}} > p_{i_{j}} \) and \( p_{t_{1}} \) shares layers with \( p_{t_{1}} \). Algorithm SDVChain
discards \( p_{t_{1}} \) or \( p_{t_{2}} \) based on \( B'_{i_{j}} \). This contradicts the assumption. Hence, \( B'_{i_{j}} \cup B'_{i_{j}} \) is path
independent, and any path \( p \in B_{t_{1}} \setminus B'_{i_{j}} \) can be represented by \( B'_{i_{j}} \) and \( B'_{i_{j}} \). It is trivial that
\( B'_{i_{j}} \cup B'_{i_{j}} \) is path independent.

Lemma 4.2 indicates that no path dependency would be produced for any \( U'_{i_{j}} \) if we
update \( B'_{i_{j}} \) only from \( B'_{i_{j}} \). This is the second trick to avoid the enumeration when considering
multiple chains. Lemmas 4.1 and 4.2 guarantee the excellent properties when Algorithm DEAH
calls Algorithm SDVChain. This results in three steps for Algorithm DEAH to find the basis
path set in a complicated network.

Step 1 hierarchically decomposes the complicated network \( G \) into \( |P_{i_{j}}| \) maximal independent
substructures, which is essentially different from Algorithm HBPS\[15\]. Compute path
subdivision set \( U'_{i_{j}} \) for each \( p_{i_{j}} \in P_{i_{j}} \) and sort \( \{U'_{i_{j}} \mid i = 0, 1 \cdots |P_{i_{j}}| - 1 \} \) with \( |U'_{i_{j}}| \) in
descending order and \( U_{i_{j}} = U_{0_{j}} \). Starting from \( i = 0 \), find the first \( |P_{i_{j}}| \) independent sub-
structure paths. We must pay attention to \( |U_{i_{j}}| = |U_{i_{j}}| \) if \( \text{Rank}(\{p_{0_{j}}, p_{1_{j}}, \cdots, p_{i_{j}}\}) > \text{Rank}(\{p_{0_{j}}, p_{1_{j}}, \cdots, p_{i_{j}}\}) \). In this case, if there exists some \( U_{i_{j}}(i < i - 1) \) such that \( U_{i_{j}} \subset U_{i_{j}} \) and \( p_{i_{j}} \) and \( p_{i_{j}} \) have the edge subdivision in each other, we should skip \( U_{i_{j}} \) and
go to \( U_{i_{j}} \). This is because the chain with \( U_{i_{j}} \) would split into two chains, but we can
manipulate to form one chain by simply skipping \( U_{i_{j}} \), as described in Figure 4.

For each \( p_{i_{j}} \in P_{i_{j}} \), Step 2 finds basis path set \( B_{i_{j}} \) by calling Algorithm Subroutine in the
corresponding sub-graph \( G_{i_{j}} \) in parallel, which can be treated as a simple network without edge-
skipping. Step 3 tries to stretch the \( t \)-th chain \( U_{t_{1}} \cdots U_{t_{j}} \cdots U_{t_{j}} \cdots U_{t_{j}} \) as long as possible
and starts one new chain from \( U_{t_{j}} \) if \( U_{t_{j}} \) couldn’t be put in any ready chain. Divide these \( |P_{i_{j}}| \)
path subdivision sets into two groups. The first group includes ones which are not the first set
in any chain. The second group consists of the first set in every chain. For each set \( U_{i_{j}} \),
in \( t \)-th chain from the first group, call Algorithm SDVChain to discard the paths from path set \( B'_{i_{j}} \)
based on original basis path set \( B_{i_{j}} \). Here we use \( B'_{i_{j}} \) instead of original \( B_{i_{j}} \), because \( U_{i_{j}} \)
may appear in different chains. For \( U_{0_{j}} \), we keep discarding paths from \( B'_{i_{j}} \) iteratively based on
\( U_{i_{j}} \) for all \( t \)-th chain. For \( U_{i_{j}} \) from the second group, enumerate all \( t \)-th chain to find the last
set \( U_{i_{j}} \) such that \( p_{i_{j}} \) shares layers with \( p_{t_{j}} \) and discard the paths from \( B'_{i_{j}} \) based on

\( \odot \) Springer
original $B_{t+1}'$, by calling Algorithm SDVChain.

In Algorithm DEAH, set $Q_t$ rules out the substructure paths: 1) The paths after $p_{t+1}'$ in the $t'$-th chain, because they don’t share layers with $p_{t+1}$. 2) The paths from the $t$-th chain because of Lemma 4.1. 3) Path $p_{t+1}' (t' > t)$ which shares edges with $p_{t+1}$, because the algorithm eliminates the path dependency between chains only from one direction.

**Theorem 4.3** Given neural network $G$ and $T$ path subdivision chains from Algorithm DEAH, i.e., $U_{t_1} \supset U_{t_2} \supset \cdots \supset U_{t_T} \supset U_0$ with $t = 1, 2, \cdots, T$. $B_{t_j}'$ output from Algorithm DEAH is the shrunk set of original basis path set $B_{t_j}$ of substructure path $p_{t_j}$. Then, the set $B = \bigcup_{i=1}^{T}(\bigcup_{j=1}^{t_i} B_{t_j}'_i) \cup B_0'$ is path independent and any path $p \in B \setminus B$ or $p \in B_0 \setminus B$ can be represented by $B$.

**Proof** We study the path independence between $p_{t_j}$ and $p_{t_j}' (t \neq t')$ from different path subdivision chains.

If $j \neq 1$ and $i \neq 1$, $U_{t_j}$ and $U_{t_j}'$ are not the first sets in the chains. We prove that $B_{t_j}' \cup B_{t_j}'$ is path independent. No path dependency exists inside the same chain after Algorithm SDVChain, according to Lemma 4.1. Assume path $p_{t_1,1} = p_{t_1}'_1 - p_{t_1,2} + p_{t_1,2}$, where $p_{t_1,1} - p_{t_1,2} \in B_{t_1}'$ and $p_{t_1,1}' - p_{t_1,2}' \in B_{t_j}'$. Apparently, neither $U_{t_j}'$ nor $U_{t_j}$ could take $U_0$. As shown in Figure 4(b), let the layers $I'$ of $p_{t_1}'$ be the edge subdivision of the layers $I$ of $p_{t_1}$, and let the layers of $J$ in $p_{t_1}$ be the edge subdivision of $J'$ in $p_{t_1}'$. As for the structure uniqueness, paths $p_{t_1,1}'$ and $p_{t_1,2}'$ must have the same layers $I'$ and $J'$ to cancel each other, and paths $p_{t_2,1}$ and $p_{t_2,2}$ have the same layers $I$ and $J$ too. For the shared layers, $p_{t_1,1}'$ and $p_{t_1,2}'$ must have $S'$ and $p_{t_2,1}'$ and $p_{t_2,2}'$ have $S$ to swap between different substructures. No overlap between $I$ and $J$. Exchange $J'$ and $J$ in $p_{t_1,1}'$ and $p_{t_2,1}'$, and exchange $J'$ and $J$ in $p_{t_1}'$ and $p_{t_2}$. Interestingly, we can get two new substructure paths $p1$ and $p2$ (in Figure 8). However, we notice that $p1, p_{t_j}$ and $p_{t_j}'$ (in Figure 4) are the path subdivisions of $p2$. It contradicts the way we select the maximal independent substructure path set.

![Figure 8](image-url) Illustration of the path exchange

If $j = 1$ and $i \neq 1$, $U_{t_j}$ is the first set in the $t$-th chain and $U_{t_j}'$ is inside the $t'$-th chain. $B_{t_j}'$ is the shrunk set of $B_{t_j}'$ based on $B_{t-1}'$ and $B_{t_j}'$ is shrunk from $B_{t_1}$. We prove that $B_{t_j}' \cup B_{t_j}'$ is path independent. Let $p_{t_j}$ be the last substructure path in the $t'$-th chain to share layers with $p_{t_1}$. If $i \leq i'$, $B_{t_j}' \cup B_{t_j}'$ is path independent, according to Lemma 4.2. If $i > i'$, $B_{t_j}' \cup B_{t_j}'$ is path independent, because $p_{t_1}$ and $p_{t_1}'$ don’t share common edges.

If $j = 1$ and $i = 1$, it is obvious that $B_{t_j}' \cup B_{t_j}'$ is path independent according to Lemma 4.2.
Algorithm 2 Algorithm DEA

Input Fully connected neural network $G = (V, E)$ with $L + 1$ layers

Output Path set $B$ of network $G$

1: Step 1. The upper level /
2: Select a random node $O^i_t$ in layer $l$. Set node set $V^S = \{O^i_t | l = 0, 1, \cdots, L\}$ and edge set $E^S = \{(O^i_t, O^i_j) \in E | O^i_j \in V^S, O^i_t \in V^S\}$. Let $P^S$ be the path set from $O^i_0$ to $O^i_L$ by breadth-first searching in $G^S = (V^S, E^S)$.

3: For each path $p_r \in P^S$, construct vectors $\alpha_r$ and $\beta_r$. Calculate $R = \text{Rank}(|p_r| = 1, 2, \cdots, |P^S|)$ by numerical linear algebra method.

4: for $r = 1: |P^S|$ / do
5: Let $U_r = \emptyset$. For each $t \in \{1, 2, \cdots, |P^S|\}$, let $X = \beta_r - \beta_t$. Let $U_r = U_r \cup \{t\}$, if $X$ doesn’t contain $1$.

6: end for
7: Pick $p_r \in P^S$ with $U_r = \emptyset$. Sort $|U_r_1| \geq |U_r_2| \geq |U_r_{r-1}| \cdots \geq |U_r_{|P^S|}| \geq |U_r_0|$. 

8: Set $i = 1$ and $A = \{\alpha_r\}$.

9: while $|A| < R$ do
10: If $\text{Rank}(A \cup \{\alpha_r\}) > \text{Rank}(A)$ and $|U_r_{r-1}| \neq |U_r_r|$, let $A = A \cup \{\alpha_r\}$. 

11: If $\text{Rank}(A \cup \{\alpha_r\}) > \text{Rank}(A)$ but $|U_r_{r-1}| = |U_r_r|$, let $X = \beta_{r-1} - \beta_r$. If there is no 0 between any 1 and -1 in $X$ or there is no $U_r_j (j < i - 1)$ such that $r_i, r_{i-1} \in U_r_j$, let $A = A \cup \{\alpha_r\}$.

12: Let $i = i + 1$. Select maximal independent substructures. /

13: end while

14: According to $A$, output $U_r$, $\{\beta_r\}$ and $P_{ind}$ set $p_r$ with $|U_r|$ in descending order. /$r = 0, 1, \cdots, R - 1$ / 

15: Step 2. The lower level /
16: for each $p_r \in P_{ind}$ do
17: Let $V_r = \{O^j \in V | p_r \in V^j \}$ and $E_r = \{(O^j, O^l) \in E | O^j \in V_r, O^l \in V_r\}$. Set $G_r = (V_r, E_r)$. / Induce $G_r$. 

18: Call Subroutine $(G_r)$ and output basis path set $B_r$.

19: end for

20: Step 3. Eliminate the path dependency. /
21: Set $X = \emptyset$ and $t = 0$.

22: for $k = 1: R$ do
23: Form multiple chains. /

24: If $k \notin X$, let $t = t + 1$ and $U_{t1} = U_k$. Stretch chain $U_{t1} \supset U_{t2} \cdots \supset U_{tj} \cdots \supset U_{t_t} \supset U_0$ by searching $\{U_t\}$ from the $(k+1)$-th set. Let $Y_t = \{t_1, t_2, \cdots, t_s\}$ and $X = X \cup Y_t$. / Start one new chain. /

25: end for

26: Let $T = \emptyset$.

27: for $t = 1: T$ do
28: Let $B_{t} = B_{t}$ for all $j$. / Initialization for Algorithm SDVChain. /

29: Let $Q_t = \{t_1, v \in E(p_{t_i}) \cap E(p_{t_{t'}}) \neq \emptyset, t' < t\}$ and $Q_t = Q_t \setminus Y_t$. / Form the second group. /

30: end for

31: for $t = 1: T$ do
32: Call SDVChain($\{B_{t_1}\}_{j=1}^{s}, \{B_{t_2}\}_{j=1}^{s}, \{p_{t_{j}}\}_{j=1}^{s}, \{\beta_{t_{j}}\}_{j=1}^{s}, B_0, p_0, \beta_0, G$), and output $\{B_{t_1}\}_{j=1}^{s}$ and $D_0$. Let $B'_{t} = B'_{t} \setminus D_0$. / Eliminate the path dependency along the $t$-th chain. /

33: For each $k \in \{1, 2, \cdots, |Q_t|\}$, call SDVChain($B'_{Q_t(k)}, B_{Q_t(k)}, p_{Q_t(k)}, \beta_{Q_t(k)}, B_{t_1}, p_{t_1}, \beta_{t_1}, G$), output discarded path set $D_{t_1}$ and let $B_{t_1} = B_{t_1} \setminus D_{t_1}$. / Eliminate the path dependency between two chains. /

34: end for

35: Output path set $B = \cup_{t=1}^{T} (\cup_{j=1}^{s} B_{t_1}) \cup B_0$. 

Scope: 1957
According to Algorithm SDVChain, $D'_0$ is the path set which is discarded from $B'_0$, based on original $B_t$ in the $t$-th chain. $B'_0$ is initialized as original $B_0$ and is updated as $B'_0 = B_0 \setminus B'_0$ iteratively for all $t = 1, 2, \cdots, T$. So, any $p \in B_0 \setminus B'_0$ can be represented by $B'_0 \cup B_t$, for some $t$. Since $U_t \supset U_t \cdots \supset U_t \cdots \supset U_t \supset U_0$ is path subdivision chain, so $B'_0 \cup B_t$ is path independent for $j = 1, 2, \cdots, s_t$ in the $t$-th chain. Therefore, $B = \bigcup_{t=1}^T (\bigcup_{j=1}^{s_t} B'_t) \cup B'_0$ is path independent based upon Claim 1 and Claim 2.

Note that set $U_t (j \neq 1)$ can appear in multiple chains but set $U_{t_j}$ can only appear in one $t$-th chain. Lemma 4.1 proves that any path $p \in B_{t_j} \setminus B'_{t_j}$ can be represented by $B_{t_{j-1}} \cup B'_{t_j}$ recursively up to $B_t$. If $U_t$ belongs to the $t'$-th chain and the $t$-th chain, i.e., $U_t \supset \cdots \supset U_{t'} \supset U_t$, $p \in B_{t_j} \setminus B'_{t_j}$ can be represented either by $B_{t_{j-1}} \cup B'_{t_j}$ or by $B_{t'} \cup B'_{t_j}$. Hence, any path $p \in B_{t_j} \setminus B'_{t_j}$ can be represented by $B$. Moreover, any path $p \in B_{t_j} \setminus B'_{t_j}$ can be represented by $B$, because Lemma 4.2 concludes $p$ can be represented by some $B_{t'}, \cup B'_{t_j}$ and any path in $B_{t'}$ can be represented by $B$. Furthermore, any $p \in B_0 \setminus B'_0$ can be represented by $B$ for $p$ can be represented by $B'_0 \cup B_{t_j}$ for some $t$.

**Theorem 4.4** Given neural network $G$ and independent path set $B = \bigcup_{t=1}^T (\bigcup_{j=1}^{s_t} B'_t) \cup B'_0$ output from Algorithm DEAH, where $T$ is the total number of path subdivision chains and $B'_t$ is the shrunk set of $B_t$ of induced subgraph $G_{t_j}$. Then any path $p \in P$ from the input layer to the output layer can be represented by $B$.

Proof Any path $p \in P$ can be represented in the hierarchical way, first at the substructure level and then at the basis path level. If the path $p \notin B$, but with the structure of $p_r \in P^{S}_{ind}$, it is trivial that path $p$ can be represented by $B_r$. If the structure of $p$ is out of the structure range of $P^{S}_{ind}$, the structure of $p$ can be expressed as $p_{r_1} \cdots + p_{r_j} \cdots + p_{r_d} - p_{s_1} \cdots - p_{s_h} \cdots - p_{s_m}$, where $r_1, \cdots, r_d, s_1, \cdots, s_m$ are distinct, $p_{r_j} \in P^{S}_{ind}$ and $p_{s_h} \in P^{S}_{ind} (1 \leq j \leq d, 1 \leq h \leq m)$. Suppose $V^S = \{O^0_{i_1}, \cdots, O^L_{i_1}, \cdots, O^L_{i_L}\}$ and the target path $p$ passes through $O^V_{i_1}, \cdots, O^L_{i_1}, \cdots, O^L_{i_L}$ sequentially. Define new node set $V' = \{O^0_{i_0}, \cdots, O^L_{i_0}, \cdots, O^L_{i_L}\}$, where $l = 0, 1, \cdots, L$. If $p$ skips over the $l$-th layer, set $O^L_{i_0} = O^L_{i'_{l}}$. If $p$ passes through the $l$-th layer and $O^L_{i_0} \neq O^L_{i'_{l}}$, set $O^L_{i_0} = O^L_{i'_{l}}$. For each $p_{r_j}$ and $p_{s_h}$, construct new paths $p'_{r_j}$ and $p'_{s_h}$ passing through node set $V'$ but $p'_{r_j} \text{and } p'_{s_h}$ keep the same structures as $p_{r_j} \text{and } p_{s_h}$ under $V^S$. $V'$ covers all nodes of path $p$, and node $O^L_{i_0} \in V^S$ corresponds to $O^L_{i'_{l}} \in V'$. In original graph $G$, path $p$ therefore can be accordingly expressed as $p = p'_{r_1} \cdots + p'_{r_j} \cdots + p'_{r_d} - p'_{s_1} \cdots - p'_{s_h} \cdots - p'_{s_m}$. Moreover, new path $p'_{r_j}$ can be represented by basis path set $B_{r_j}$ and $p'_{s_h}$ can be represented by $B_{s_h}$. Hence, $p$ can be represented by $B = \bigcup_{t=1}^T (\bigcup_{j=1}^{s_t} B'_t) \cup B'_0$ according to Theorem 4.3.

Therefore, $B$ is a basis path set for network $G$.

Theorem 4.3 and Theorem 4.4 prove that Algorithm DEAH can find basis path set $B$ for BasisPathSetSearching problem in complicated network $G$. Next, we prove that Algorithm DEAH can be completed in polynomial time.
Lemma 4.5  The time complexity of Algorithm Subroutine $(G)$ in network $G$ without any edge skipping is $O(L^2W_{\text{max}}^2)$, where $W_{\text{max}} = \max_{0 \leq i \leq L} |O^i|$.  

Proof  Algorithm Subroutine takes at most $O(W_{\text{max}}^2L)$ time to construct all sub-graph $G(k)$ ($k = 0, 1, \ldots, L - 1$). It takes $O(W_{\text{max}})$ time to search direct path set $P_{\text{dir}}^k$ and cross path set $P_{\text{cross}}^k$ in $G(k)$, so the total running time for all $G(k)$ is $O(W_{\text{max}}^2L)$. When updating $P_{\text{dir}}^k$ and $P_{\text{cross}}^k$, the average number of paths from the lower layer is at most $O(kW_{\text{max}})$ for each node $O^k_t$ and there are at most $W_{\text{max}}$ nodes for $k$-th layer, so the total time complexity for the path updating is $O \left( \frac{k(L-k)}{2} W_{\text{max}}^2 \right)$. The time for classifying paths to $(k+1)$-th layer is the same as for updating $P_{\text{dir}}^k$ and $P_{\text{cross}}^k$. So the total running time for Algorithm Subroutine is $O(W_{\text{max}}^2L + W_{\text{max}}^2L + 2 \times \frac{k(L-k)}{2} W_{\text{max}}^2) = O(L^2W_{\text{max}}^2)$.

Theorem 4.6  The time complexity of Algorithm DEAH to solve BasisPathSetSearching problem in neural network $G$ is $O(RL^2W_{\text{max}}^2 + O(\max(R,T^2) \cdot (L + B_{\text{max}}^3)))$, where $R = |P_{\text{ind}}^{|}$, $W_{\text{max}} = \max_{0 \leq i \leq L} |O^i|$ and $B_{\text{max}} = \max_{r \in \{1,2,\ldots,R\}} |B_r|$.

Proof  There are three major steps in Algorithm DEAH.

Step 1 of Algorithm DEAH finds $V_S$ in $O(L)$ time, searches $E_S$ in $O(m)$ time and finds $P_S$ in at most $O(m)$ time. In order to get $P_{\text{ind}}$, we compute all $U_r$ in at most $O(m^2)$ time and compute $\text{Rank}(A)$ by searching $U_{r_1}, U_{r_2}, \ldots, U_{r_{P_S}}$ in at most $O(m)$ time, since $|P_S| \leq m$. So, Step 1 runs in $O(3m + m^2 + L) = O(m^2)$ time.

Step 2 calls Subroutine $(G_r)$ to find basis path set $B_r$ in induced sub-graph $G_r$ in terms of substructure path $p_r$ ($r = 1, 2, \ldots, R$). Lemma 4.5 proves that Algorithm Subroutine $(G)$ runs in time $O(L^2W_{\text{max}}^2)$ in simple network $G$ without any edge-skipping. Let $W_{\text{max}} = \max_{0 \leq i \leq L} |O^i|$ in network $G$. Therefore, the time complexity of Step 2 is $O(RL^2W_{\text{max}}^2)$.

Step 3 consists of two parts to compute the shrunk sets. One part is to compute $B'_{t_{j+1}} (j \neq 0)$. In each $t$-th chain we call SDVChain to compute $B'_{t_{j+1}}$ based on $B_{t_{j+1}}$. It takes $O(L)$ time to find common layers between $p_{t_{j}}$ and $p_{t_{j+1}}$, takes $O(|B_{t_{j}}|)$ time to separate the shared layers and unshared layers in $B_{t_{j}}$ and takes $O(|B_{t_{j+1}}|)$ time to separate the layers of $B_{t_{j+1}}$. It needs $O(|B_{t_{j}}|)$ time to get unique unshared layers $\{E_{p_{t_{j}}} \}$ for each $E_{p_{t_{j}}}$, it takes at most $O(|B_{t_{j}}|)$ time to compute $UCP_{t_{j}}$ and $UCP^*_{t_{j}}$ in $B_{t_{j}}$, and takes $O(|B_{t_{j}}|)$ time to get unshared layers set $I_{E_{p_{t_{j}}}}$ in $B_{t_{j+1}}$. Since there are at most $O(|B_{t_{j}}|)$ elements in $\{E_{p_{t_{j}}} \}$, so this phase needs at most $O(|B_{t_{j}}| \cdot (|B_{t_{j}}| + |B_{t_{j}}|)) = O(B_{t_{j}}^3)$ time, where $B_{t_{j}} = \max_{r \in \{1,2,\ldots,R\}} |B_r|$. Furthermore, there are $T$ chains, each $t$-th chain calls SDVChain for $s_t$ iterations, some $U_{t_{j+1}}$ may appear in most $T$ chains and all path after $U_{t_{j+1}}$ in the chain only needs computing once for the set containment relationship. Hence, we need at most $O(R + T) \cdot O(L + |B_{t_{j}}| + |B_{t_{j+1}}| + |B_{t_{j}}^2 + B_{t_{j+1}}^3|) = O((R + T)(L + B_{t_{j}}^3)) = O(R(L + B_{t_{j}}^3))$ time. For the other part, each $U_{t_{j}}$ enumerates all elements in $Q_t$ to run the simplest SDVChain to update $B'_{t_{j}}$ based on $B_{Q_t}$ iteratively. This simple version of SDVChain would take at most $O(L + B_{t_{j}}^3)$ time for each iteration and there are at most $T$ elements in $B_{Q_t}$, i.e., totally $O(T(L + B_{t_{j}}^3))$ time. There are $T$ chains, so this phase takes $O(T^2(L + B_{t_{j}}^3))$ time. Therefore, Step 3 takes totally $O(R(L + B_{t_{j}}^3)) + O(T^2(L + B_{t_{j}}^3)) = O(\max(R,T^2)(L + B_{t_{j}}^3))$ time.
In sum, the total time complexity of Algorithm DEAH is \( O(m^2) + O(RL^2W_{\text{max}}^2) + O(\max(R, T^2)(L + B_{\text{max}}^3)) = O(RL^2W_{\text{max}}^2) + O(\max(R, T^2)(L + B_{\text{max}}^3)). \)

Theorem 4.6 indicates that the computation complexity of Algorithm DEAH depends heavily on the structure of the network, i.e., the maximal layer width \( W_{\text{max}} \) and edge-skipping over layers. Usually \( B_0 \) is the basis path set taking \( B_{\text{max}} \). Each \( p_r \in P_{\text{std}}^S \) represents one type of substructure information about the edge-skipping over layers in network \( G \), and different types of substructures can be combined in a variety of ways. Thus, there is no constraint about the graph structure levying on Algorithm DEAH, which breaks the bottleneck of the algorithm in [15] and is generalized to more practical networks.

5 Conclusion

In a regular fully connected neural network \( G \), the shared layers between two independent substructure paths bring up the combinatoric possibility of path dependency, when combining the basis path sets from both substructures. Algorithm DEAH is designed to eliminate such path dependency and the crucial point for the effective elimination is the path subdivision chain. The theoretical proofs guarantee the feasibility of Algorithm DEAH for BasisPathSetSearching problem. The paper generalizes the algorithms and theoretical results from the specific network structure without shared layers between two independent substructures to more practical network structures, and provides one methodology to find basis path set in a more general neural network. This work can help facilitate the theoretic research and applications of \( G \)-SGD algorithm in more practical scenarios.

References

[1] Wu S, Dimakis A G, and Sanghavi S, Learning distributions generated by one-layer ReLU networks, 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 2019.
[2] Wang Y, Liu Y T, and Ma Z M, The scale-invariant space for attention layer in neural network, Neurocomputing, 2020, 392: 1–10.
[3] Neelyshabur B, Salakhutdinov R R, and Srebro N, Path-sgd: Path normalized optimization in deep neural networks, NIPS’15 Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015, 2422–2430.
[4] Zheng S X, Meng Q, Zhang H S, et al., Capacity control of ReLU neural networks by basis-path norm, Thirty-third AAAI Conference on Artificial Intelligence (AAAI2019), 2019.
[5] Meng Q, Zheng S X, Zhang H S, et al., \( G \)-SGD: Optimizing ReLU neural networks in its positively scale-invariant space, International Conference of Learning Representations (ICLR2019), 2019.
[6] Rumelhart D E, Hinton G E, and Williams R J, Learning representations by back-propagating errors, Nature, 1986, 323(6088): 533–536.
[7] Fan F, Xiong J, Li M, et al., On interpretability of artificial neural networks: A survey, *IEEE Transactions on Radiation and Plasma Medical Sciences*, 2021, 5(6): 741–760.

[8] Guan C, Wang X, Zhang Q, et al., Towards a deep and unified understanding of deep neural models in NLP, *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, USA, 2019.

[9] Hooker S, Erhan D, Kidermans P, et al., A Benchmark for Interpretability Methods in Deep Neural Networks, *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019.

[10] Inoue K, Expressive numbers of two or more hidden layer ReLU neural networks, 2019 *Seventh International Symposium on Computing and Networking workshops (CANDARW 2019)*, 2019.

[11] Zhang Q S, Cao R M, Shi F, et al., Interpreting CNN knowledge via an explanatory graph, *The Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, 4454–4463.

[12] Wu M, Wicker M, Ruan W, et al., A game-based approximate verification of deep neural networks with provable guarantees, *Theoretical Computer Science*, 2020, 807: 298–329.

[13] Ensign D, Neville S, Paul A, et al., The complexity of explaining neural networks through (group) invariants, *Theoretical Computer Science*, 2020, 808: 74–85.

[14] Xing R T, Xiao M, Zhang Y Z, et al., Stability and Hopf bifurcation analysis of an $(n + m)$-neuron double-ring neural network model with multiple time delays, *Journal of Systems Science & Complexity*, 2021, DOI: 10.1007/s11424-021-0108-2.

[15] Zhu J P, Meng Q, Chen W, et al., Interpreting basis path set in neural networks, *Journal of Systems Science and Complexity*, 2020, 33(1): 1–13.

[16] Corberan A and Laporte G, *Arc Routing Problems, Methods, and Applications*, Society for Industrial and Applied Mathematics, 2015.

[17] Jensen J B and Gutin G Z, *Digraphs: Theory, Algorithms and Applications* (Second Edition), Springer, New York, 2009.

[18] Korte B and Vygen J, *Combinatorial Optimization, Theory and Algorithm* (Fifth Edition), Springer, New York, 2012.

[19] Babu C S and Diwan A A, Subdivisions of graphs: A generalization of paths and cycles, *Discrete Mathematics*, 2008, 308(19): 4479–4486.

[20] Bondy J A and Murty U S R, *Graph Theory*, Section 10.1, 2008.

[21] Dettlaff M, Raczek J, and Yero I G, Edge subdivision and edge multisubdivision versus some domination related parameters in generalized corona graph, *Opuscula Mathematica*, 2016, 36(5): 575–588.

[22] Chaieb M, Jemai J, and Mellouli K, A hierarchical decomposition framework for modeling combinatorial optimization problems, *Procedia Computer Science*, 2015, 60: 478–487.

[23] Chang Y, Tang H, Cheng Y, et al., Dynamic hierarchical energy efficient method based on combinatorial optimization for wireless sensor networks, *Sensors*, 2017, 17(7): 1665.

[24] Ochiai H, Kanazawa T, Tamura K, et al., Combinatorial optimization method based on hierarchical structure in solution space, *Electronics and communications in Japan*, 2016, 99(18): 25–37.

[25] Racke H, Optimal hierarchical decompositions for congestion minimization in networks, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 2008, 255–264.
Appendix

Algorithm 3 Subroutine(G)\(^{[15]}\)

**Input** Fully connected neural network \(G = (V,E)\) without edge-skipping

**Output** Basis path set \(B\) in graph \(G\).

1. / This subroutine is to find a basis path set on \(G\) /
2. for \(k = 0: L-1\) do
3. \(E^k = \{e \in G | e \) leaves from the \(k\)-th layer and enters the \((k+1)\)-th layer\}.
4. / Step 1. Construct the direct path set /
5. Let sub-graph \(G(k) = (O^k \cup O^{k+1}, E^k)\).
6. if \(|O^k| \geq |O^{k+1}|\) then
7. \(\text{Find } |O^{k+1}| \text{ vertex disjoint paths by depth-first searching and let the path set be } P_{dir}^{(k)}\).
   For each \(v \in O^k \setminus V(P_{dir}^{(k)})\), pick up one node \(O_{vy}^{k+1} \in O^{k+1}\) randomly and construct path \((v, O_{vy}^{k+1})\). Set \(P_{dir}^{(k)} = P_{dir}^{(k)} \cup \{(v, O_{vy}^{k+1})\}\).
8. else
9. \(\text{Find } |O^k| \text{ vertex disjoint paths by depth-first searching and let the path set be } P_{dir}^{(k)}\).
10. end if
11. / Step 2. Construct the cross path set /
12. Set cross path set \(P_{cross}^{(k)} = E^k \setminus E(P_{dir}^{(k)})\).
13. for \(i = 1: |O^k|\) do
14. Let the path set \(P_{dir}^{(k)}|O_i^k = \{p \in P_{dir}^{(k)} | \text{the tail of } p \text{ is node } O_i^k\}\) and \(P_{cross}^{(k)}|O_i^k = \{p \in P_{cross}^{(k)} | \text{the tail of } p \text{ is node } O_i^k\}\). / Classify direct paths and cross paths for \(O_i^k \in O^k\) /
15. end for
16. / Step 3. Concatenate the direct paths and cross paths from the \((k-1)\)-th layer /
17. if \(k \neq 0\) / No concatenation for \(k = 0\) / then
18. for \(i = 1: |O^k|\) do
19. Let \(P_{dir}^{(k)}|O_i^k = \{p_0 + p_1 | p_1 \in P_{dir}^{(k)}|O_i^k, p_0 \in P(O_i^k)\}\) for node \(O_i^k \in O^k\). / Form \(P(O_i^k)\) direct paths /
20. Select one path \(p^* \in P(O_i^k)\) randomly and let \(P_{cross}^{(k)}|O_i^k = \{p^* + p_1 | p_1 \in P_{cross}^{(k)}|O_i^k\}\). / Extend all cross paths /
21. end for
22. Update \(P_{dir}^{(k)} = \cup_{O_i^k \in O^k} P_{dir}^{(k)}|O_i^k\) and \(P_{cross}^{(k)} = \cup_{O_i^k \in O^k} P_{cross}^{(k)}|O_i^k\).
23. end if
24. / Step 4. Classify the paths for the nodes in the \((k+1)\)-th layer /
25. for \(i = 1: |O^{k+1}|\) do
26. Let the path set \(P(O_i^{k+1}) = \{p \in P_{dir}^{(k)} \cup P_{cross}^{(k)} \mid \text{the head of } p \text{ is node } O_i^{k+1}\}\).
27. end for
28. end for
29. Output basis path set \(B = P_{dir}^{(L-1)} \cup P_{cross}^{(L-1)}\).