Epitopological sparse ultra-deep learning: a brain-network topological theory carves communities in sparse and percolated hyperbolic ANNs

Yingtao Zhang, Jialin Zhao, Wenjing Wu, Alessandro Muscoloni, Carlo Vittorio Cannistraci*
Center for Complex Network Intelligence (CCNI), Tsinghua Laboratory of Brain and Intelligence (THBI)
Department of Computer Science, Tsinghua University
Department of Biomedical Engineering, Tsinghua University
*Corresponding author: Carlo Vittorio Cannistraci (kalokagathos.agon@gmail.com)

Abstract

Sparse training (ST) aims to improve deep learning by replacing fully connected artificial neural networks (ANNs) with sparse ones, akin to the structure of brain networks. Therefore, it might benefit to borrow brain-inspired learning paradigms from complex network intelligence theory. Epitopological learning (EL) is a field of network science that studies how to implement learning on networks by changing the shape of their connectivity structure (epitopological plasticity). One way to implement EL is via link prediction: predicting the existence likelihood of nonobserved links in a network. Cannistraci-Hebb (CH) learning theory inspired the CH3-L3 network automata rule for link prediction which is effective for general-purpose link prediction. Here, starting from CH3-L3 we propose Epitopological Sparse Ultra-deep Learning (ESUL) to apply EL into sparse training. In empirical experiments, we find that ESUL learns ANNs with sparse hyperbolic topology in which emerges a community layer organization that is ultra-deep (meaning that also each layer has an internal depth due to power-law node hierarchy). Furthermore, we discover that ESUL automatically sparse the neurons during training (arriving even to 30% neurons left in hidden layers), this process of node dynamic removal is called percolation. Then we design CH training (CHT), a training methodology that put ESUL at its heart, with the aim to enhance prediction performance. CHT consists of 4 parts: (i) correlated sparse topological initialization (CSTI), to initialize the network with a hierarchical topology; (ii) sparse weighting initialization (SWI), to tailor weights initialization to a sparse topology; (iii) ESUL, to shape the ANN topology during training; (iv) early stop with weight refinement, to tune only weights once the topology reaches stability. We conduct experiments on 6 datasets and 3 network structures (MLPs, VGG16, Transformer) comparing CHT to sparse training SOTA method and fully connected network. By significantly reducing the node size while retaining performance, CHT represents the first example of parsimony sparse training.

1 Introduction

Over-parameterization[1] represents a significant advancement in deep learning. It has demonstrated that the loss function experiences a double descent phenomenon as the model complexity increases. This observation has facilitated the development of large models incorporating numerous modules and parameters to achieve superior performance. However, this approach places a significant burden on computational and storage resources, particularly for university researchers, small companies, and edge devices. As a result, researchers are currently focusing on methods to compress the model

Preprint. Under review.

in order to increase its trade-off efficiency between performance and model size. There are many possible ways to compress a neural network, including Knowledge Distillation[2] to compress the model size, Model Quantification[3, 4] to decrease the floating numbers during model training, sparse-based methods[5, 6, 7, 8, 9, 10, 11, 12, 13] to sparsify the connections in the model, and so on. Of all the compression-based methods, the sparse-based ones are the closest to the original design of Artificial Neural Networks (ANNs), which aim to imitate the structure of Brain Neural Networks (BNNs) where the connections in BNNs have already been proven to be extremely sparse. And in general, the sparse-based methods have shown good performance in various tasks[10] Current sparse-based methods can be divided into two categories: a) Pruning[5, 6, 7, 8], which involves starting from a fully connected network and pruning the connections step by step; b) Sparse Training (ST), which begins with a sparsely connected network. Dynamic Sparse Training (DST)[9, 10, 11, 12, 13], a type of ST, is considered the most effective strategy as it can dynamically evolve the network topology. However, most DST algorithms only utilize the gradient information[10, 13] or randomly assign new links[9, 12] without considering and exploiting ANNs complex network topology. We believe that integrating topological network science (or, more specifically, brain-inspired learning paradigms from complex network intelligence theory) theories into DST could yield interesting results.

Epitopological Learning (EL)[14, 15, 16] is a field of network science inspired by the brain that explores how to implement learning on networks by changing the shape of their connectivity structure (epitopological plasticity). One approach to implement EL is via link prediction: predicting the existence likelihood of each non-observed link in a network. EL was developed alongside the Cannistraci-Hebb (CH) learning theory according to which: the sparse local-community organization of many complex networks (such as the brain ones) is coupled to a dynamic local Hebbian learning process and contains already in its mere structure enough information to predict how the connectivity will evolve during learning partially. Building upon CH3-L3[17], one of the network automata rules under CH theory, we propose Epitopological Sparse Ultra-deep Learning (ESUL) to implement EL in evolving the topology of ANNs based on the DST procedure. Finally, we design CH training (CHT), a training methodology that put ESUL at its heart, with the aim of enhancing prediction performance. The experimental results show that CHT performs often better and trains faster than the SOTA dynamic sparse training algorithm. CHT reduces also the network node size significantly. Furthermore, in many experiments, CHT can equal or lightly outperform fully connected networks.

2 Related work

In this section, we will introduce some basic notions of DST. Furthermore, we introduce epitopological learning and Cannistraci-Hebb theory for network automata link prediction, providing the knowledge to understand our proposed CHT methodology for ESUL in ANNs.

2.1 Dynamic Sparse Training

Dynamic Sparse Training (DST) is a novel training paradigm that can evolve the network structure efficiently for achieving better performance. It begins with a sparse initialized network and undergoes the removal and regrown procedure with multiple iterations. The main difference in modern DST methods is how to regrow the new links of the model. As summarized in Suppl. Table 1, there are two main streams of regrowth methods, random-based and gradient-based. random-based methods, such as SET[9], MEST[12], randomly assign new links to the network after removal, while gradient-based methods, such as RigL[10], extract the non-existing links with top-k absolute values of gradients. Generally, gradient-based methods perform better since they utilize more information from the learning process. However, the trade-off is that the backpropagation process of the gradient transformation is still fully connected. Recently, researchers are exploring ways to reduce the computation of the backpropagation process using gradient-based regrown[13] or designing some tricks to improve DST[12, 11, 18]. Therefore, in this paper, we introduce epitopological leaning, a new brain-inspired paradigm from network science, to evolve the sparse network topology.

2.2 Epitopological Learning and Cannistraci-Hebb network automata theory for link prediction

Drawn from neurobiology, Hebbian learning was introduced in 1949[19] and can be summarized in the axiom: “neurons that fire together wire together”. This could be interpreted in two ways: changing

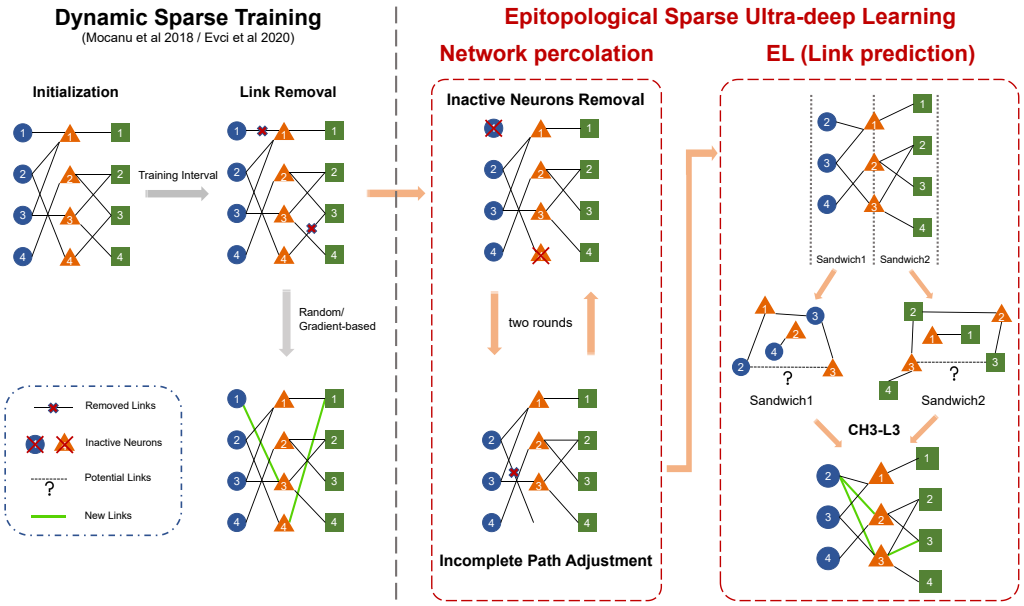


Figure 1: Illustration of the process of standard DST and ESUL. ESUL incorporates two additional steps for network evolution, namely network percolation and EL(link prediction). Network percolation involves inactive neurons removal (INR) and the adjustment of incomplete paths (IPA). In the epitopological learning (EL) part, which is based on link prediction, after dividing the entire network into several sandwich layers, the chosen link predictor separately calculates the link prediction score for each non-existing link and then regrows the top-k links in the same quantity as those removed.

the synaptic weights (weight plasticity) and changing the shape of synaptic connectivity[20, 14, 15, 16, 21]. The latter is also called epitopological plasticity[20] because plasticity means "to change shape" and epitopological means "via a new topology". Epitopological Learning (EL)[14, 15, 16] is derived from this second interpretation of Hebbian learning, and studies how to implement learning on networks by changing the shape of their connectivity structure. In this study, we adopt CH3-L3[17] as link predictor to regrow the new links during the DST process. CH3-L3 is one of the best and most robust performing network automata which is inside Cannistraci-Hebb (CH) theory[17] that can automatically evolve the network topology with the given structure. The rationale of the CH theory is that, in any complex network with local-community organization, the cohort of nodes (neurons in the case of brain networks) tend to be co-activated (fire together) and to learn by forming new connections between them (wire together) because they are topologically isolated in the same local community[17]. CH3-L3 perfectly accomplishes this task by reducing the interaction with the nodes from the external community, which can be formalized as below:

$$CH3-L3=\sum_{z_1,z_2\in L3}\frac{1}{\sqrt{(1+de_{z_1})*(1+de_{z_2})}} \tag{1}$$

where: u and v are the two seed nodes of the candidate interaction; z_1, z_2 are the intermediate nodes on the considered path of length three; de_{z_1}, de_{z_2} are the respective external node degrees; and the summation is executed over all the paths of length three between the seed nodes u and v. The formula of $CH3-Ln$ is defined on path of any length, which adapts to any type of network organization[17]. Here, we consider $CH3-L3$ because the topological regrowth process is implemented on each ANN sandwich layer separately, which has bipartite network topology organization. And, bipartite topology is based on the path of length 3 which displays a quadratic closure[14].

3 Epitopological sparse ultra-deep learning and Cannistraci-Hebb training

3.1 Epitopological sparse ultra-deep learning

Following the second interpretation of the Hebbian Learning rule, we propose epitopological sparse ultra-deep Learning (ESUL), which implements EL in the sparse deep learning ANN. ESUL changes

the perspective of training ANNs from weights to topology. ESUL focuses on the sparse topology of each sandwich layer, which is a bipartite sub-network composed of two consecutive layers of the ANN. The topological update formula for EL is:

$$T^{n+1} = LP(T^n) \tag{2}$$

Where T^n represents the topology of the system at state n and T^{n+1} represents the topology of the system at state $n+1$. LP is the link predictor used by EL to predict the links between nodes in the network. In the standard DST process, as shown in Fig. 1, there are three main steps: initialization, weight update, and network evolution, which include link removal and regrowth. ESUL follows initially the same steps of standard DST(Fig. 1, left side), but then progresses with substantial improvements (Fig. 1, right side): new removal part (network percolation), and new regrowth part (EL via link prediction). Firstly, in the removal phase, ESUL adopts the commonly used removal method that prunes links with lower absolute weights, which are considered less useful for the model's performance. However, after the process of removing the lower magnitude weights, ESUL checks and corrects the topology for the presence of inactive neurons that either do not have a connection in one layer side or do not have any connections in both layer sides. When information passes through these inactive neurons, the forward and backward processes can get stuck. Moreover, if a node does not have any links in a bipartite network (i.e., a sandwich layer of the ANN), even if it has connections on the other side, no new links will be assigned to it in that network according to the formula of CH theory. Thus, ESUL implements the inactive neurons removal (INR) to ablate from the topology those inactive neurons. In particular, for the case that one node possesses connections on one layer side, we need to conduct incomplete path adjustment (IPA) that is to remove those links part of an incomplete path connected to the inactive neuron. Since IPA may produce some extra inactive neurons, we execute INR and IPA for two rounds. We termed the coupled process of INR and IPA as network percolation because it is reminiscent of the percolation theory in network science, which studies the connectivity behavior of networks when a fraction of the nodes or edges are removed[22]. If there are still some inactive neurons, we leave them in the network and clean up them during the next evolution. Once the network percolation is finished, EL via link prediction is implemented to regrow the links with higher link prediction scores. We repeat the process of removal and regrown after each learning weight update and until the network convergence to a stable topology. The steps of ESUL are explained in Suppl. Algorithm 1.

3.2 Cannistraci-Hebb training

ESUL is a methodology that can work better when the network has already formed some relevant structural features. For instance, the fact that in each layer the nodes display a hierarchical position is associated with their degree. Nodes with higher degrees are network hubs in the layer and therefore they have a higher hierarchy. This hierarchy is facilitated by a hyperbolic network geometry with power-law-like node degree distribution[23]. In addition, the efficiency of the link predictor can be impaired by sparse random connectivity initialization. Therefore, we introduce below a strategy to initialize the sparse network topology and its weights. On the other hand, once the network topology has already stabilized (nodes removed and regrown are consistently similar), there may no longer be necessity to perform ESUL which could save the time of evolution. Therefore, to address these concerns, we propose a novel 4-step training procedure named Cannistraci-Hebb training (CHT).

- *Correlated sparse topological initialization (CSTI)*: as shown in Fig. 2, CSTI consists of 4 steps. During the vectorization phase, we follow a specific procedure to construct a matrix of size n x M, where n represents the number of samples selected from the training set. Here, M denotes the number of valid features, obtained by excluding features with zero variance among the selected samples. Once we have this n x M matrix, we proceed with feature selection by calculating the Pearson Correlation for each feature. This step allows us to construct a correlation matrix. Subsequently, we construct a sparse adjacency matrix, where the positions marked with "1" (represented as white in the heatmap plot of Fig. 2A) correspond to the top-k% values from the correlation matrix. The specific value of k depends on the desired sparsity level. This adjacency matrix plays a crucial role in defining the topology of each sandwich layer. The dimension of the hidden layer is determined by a scaling factor denoted as 'x'. A scaling factor of 1x implies that the hidden layer's dimension is equal to the input dimension, while a scaling factor of 2x indicates that the hidden layer's dimension is twice the input dimension which allows the dimension of the hidden layer to be variable. In

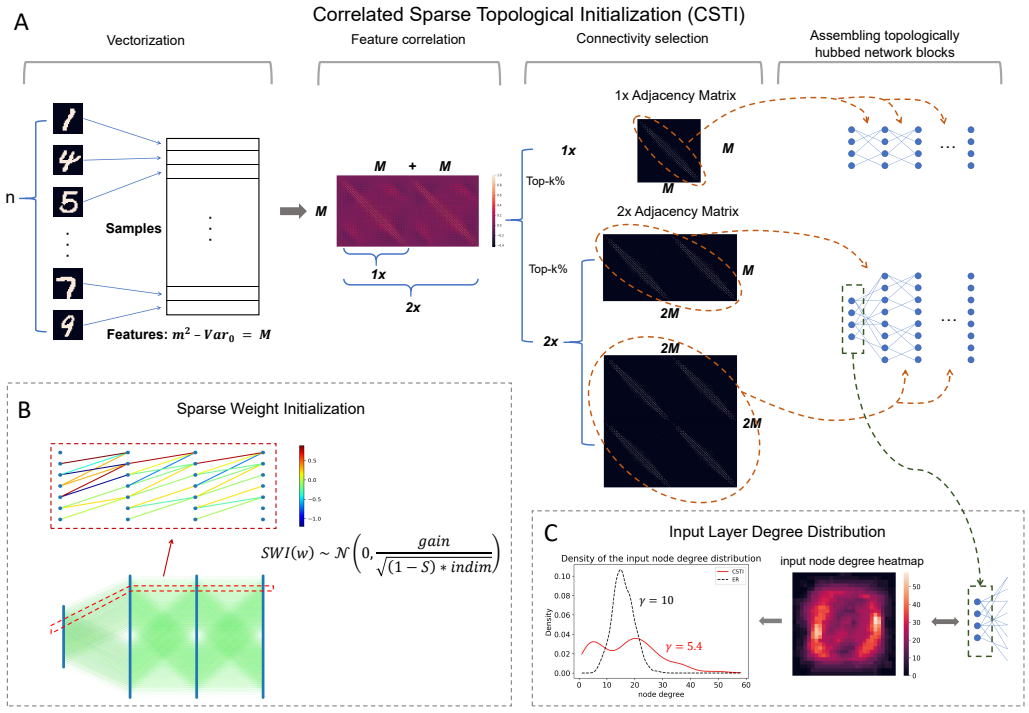


Figure 2: Main innovations introduced with CHT. **A** shows an example of how to construct the CSTI on the MNIST dataset, which involves four steps: vectorization, feature correlation, connectivity selection, and assembling topological hubbed network blocks. **B** presents a real network example of how SWI assigns weights to the sparse network. **C** displays the input layer degree distribution along with the associated heatmap on the MNIST figure template and the node degree distribution.

fact, since ESUL can efficiently reduce the dimension of each layer, the hidden dimension can automatically reduce to the inferred size. In this study we are only focusing on the sandwich layers in each model, in cases where sandwich layers cannot receive direct information from the input features, such as CNN and Transformer, we directly run the untrained network and use CSTI to initialize the adjacency matrix of each layer. The detailed implementation of VGG16 and Transformer can be found in the supplementary material.

- *Sparse Weight Initialization (SWI)*: in addition to the topological initialization, we also recognize the importance of weight initialization in the sparse network. The standard initialization methods such as Kaiming[24] or Xavier[25] are designed to keep the variance of the values consistent across layers. However, these methods are not suitable for sparse network initialization since the variance is not consistent with the previous layer. To address this issue, we propose a method that can assign initial weights in any sparsity cases. SWI can also be extended to the fully connected network, in which case it becomes equivalent to Kaiming initialization. Here we provide the mathematical formula for SWI and in supplementary we provide the rationale that brought us to its definition.

$$SWI(w) \sim \mathcal{N}(0, \sigma^2), \sigma = \frac{gain}{\sqrt{(1-S)*indim}} \tag{3}$$

$\mathcal{N}(0, \sigma^2)$ denotes the normal distribution with a mean of 0 and a variance of σ^2. The value of $gain$ varies for different activation functions. In this article, except for Transformer, all other models use ReLU, where the gain is always $\sqrt{2}$. S here denotes the desired sparsity and $indim$ indicates the input dimension of each sandwich layer.

- *Epitopological prediction*: this step corresponds to ESUL (introduced in section 3.1), but evolves the sparse topological structure starting from CSTI + SWI initialization rather than random.
- *Early stop and weight refinement*: during the process of epitological prediction, it is common to observe an overlap between the links that are removed and added, as shown in the last plot in Suppl. Fig. 2D. After several rounds of topological evolution, the overlap rate can reach a high level,

indicating that the network has achieved a relatively stable topological structure. In this case, the ESUL algorithm may continuously remove and add mostly the same links, which can slow down the training process. To solve this problem, we introduce an early stop mechanism for each sandwich layer. When the overlap rate between the removed and added links reaches a certain threshold (we use a significant level of 90%), we stop the epitopological prediction for that layer. Once all the sandwich layers have reached the early stopping condition, the model starts to focus on learning and refining the weights using the obtained network structure.

4 Results

In this section, we present the experimental results obtained by applying ESUL and CHT on 6 datasets (MNIST[26], Fashion_MNIST[27], EMNIST[28], CIFAR10[29], CIFAR100[29], and Multi-30K[30]) using 3 different network architectures (MLP, VGG16[31], and Transformer[32]) to evaluate their performance compared to DST methods and fully connected networks. All the experiments are using 3 random seeds and the hyperparameter settings are reported in the Suppl. Table 2. All experiments are with 99% sparsity since we want to investigate extremely sparse learning scenarios and compare them with fully connected ones.

4.1 Results of ESUL

In this section, we aim to investigate the effectiveness of ESUL from a network science perspective. We utilize the framework of ESUL and compare CH3-L3 (the proposed topological link prediction-based regrown method) with randomly assigning new links (which is typical of SET[9]). We run all the experiments with the default hyperparameter setting as shown in Suppl. Table 2, and conduct experiments on 5 datasets: MNIST, Fashion_MNIST, EMNIST using MLP and CIFAR10, CIFAR100 using VGG16. For the first 3 datasets, we adopt MLP with an architecture of 784-1000-1000-1000-10 (47 output neurons for EMNIST), while for VGG16, we replace the fully connected layers after the convolution layers with the sparse network and assign them a structure of 512-1000-1000-1000-10 (100 output neurons for CIFAR100). Fig. 3 compares the performance of CH3-L3 (ESUL) and random (SET) on the five datasets. The first row shows the accuracy curves of each algorithm and on the upper right corner of each panel, we mark the accuracy improvement at the 250^{th} epoch. The second row of Fig. 3 shows the Area Across the Epochs (AAE), which reflects the learning speed of each algorithm. The computation and explanation of AAE can be found in the supplementary material. Based on the results of the accuracy curve and AAE values, CH3-L3 (ESUL) outperforms random (SET) on all the datasets, demonstrating that the proposed epitopological learning method is effective in identifying lottery tickets, and CH theory is boosting deep learning.

Fig. 4A provides evidence that the sparse network trained with ESUL can automatically percolate the network to a significantly smaller size and form a hyperbolic network with community organizations. The example network structure of MNIST dataset on MLP is presented, where each block shows a plain representation and a hyperbolic representation of each network at the initial (that is common) and the final epoch status (that is random (SET) or CH3-L3 (ESUL)). The ESUL block in Fig. 4A shows that each layer of the network has been percolated to a small size, especially the middle two hidden layers where the active neuron post-percolation rate (ANP) has been reduced to less than 20%. This indicates that the network trained by ESUL can achieve better performance with significantly fewer active neurons and that the size of the hidden layers can be automatically learned by the topological evolution of ESUL. ANP is particularly significant in deep learning because neural networks often have a large number of redundant neurons that do not contribute significantly to the model's performance. ESUL can reduce the computational cost of training and inference, making the network more efficient and scalable. In addition, reducing the number of active neurons can also help to prevent overfitting and improve the generalization ability of the model which remarkably reduces the dimensionality of the embedding in the hidden layers.

Furthermore, we perform coalescent embedding[33] of each network in Fig. 4A into the hyperbolic space, where the radial coordinates are associated with node degree hierarchical power-law-like distribution and the angular coordinates with geometrical proximity of the nodes in the latent space. This means that the typical tree-like structure of the hyperbolic network emerges if the node degree distribution is power law ($\gamma \leq 3$[23]). Indeed, the more power-law in the network, the more the

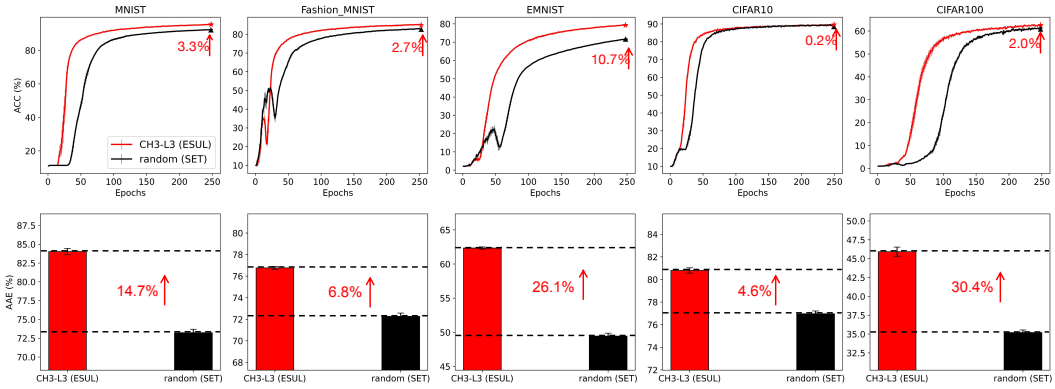


Figure 3: The comparison of CH3-L3 (ESUL) and random (SET). In the first row, we report the accuracy curves of CH3-L3 and random on 5 datasets and mark the increment at the 250^{th} epoch on the upper right corner of each panel. The second row reports the value of area across the epochs (AAE) which indicates the learning speed of different algorithms corresponding to the upper ACC plot.

nodes migrated towards the center of the hyperbolic 2D representation, and the more the network displays a hierarchical hyperbolic structure.

Surprisingly, we found that the network formed by CH3-L3 finally (at the 250^{th} epoch) becomes a hyperbolic power law ($\gamma = 2.32$) network with a hyperbolic community organization (angular separability index, ASI = 1, indicating a perfect angular separability of the community formed by each layer[34]), while the others (initial and random at the 250^{th} epoch) do not display either power law ($\gamma = 10$) topology with latent hyperbolic geometry or crystal-clear community organization (ASI around 0.6 denotes a substantial presence of aberration in the community organization[34]). Community organization is a fundamental mesoscale structure of real complex networks[35] such as biological and socioeconomical[36]. For instance, brain networks[37] and maritime networks[38] display distinctive community structure that is fundamental to facilitating diversified functional processing in each community separately and global sharing to integrate these functionalities between communities. ESUL approach can not only efficiently identify the important neurons but also learn a more complex and ultra-deep network structure. This mesoscale structure leverages topologically separated layer-community to implement in each of them diversified and specialized functional processing. The result of this 'regional' layer-community processing is then globally integrated together via the hubs (nodes with higher degrees) that each layer-community owns. Thanks to the power-law distributions, regional hubs that are part of each regional layer-community emerge as ultra-deep nodes in the global network structure and promote a hierarchical organization that makes the networks ultra-small-world (as we will explain a few lines below). Indeed, in Fig. 4A left we show that the community layer organization of the ESUL learned network is ultra-deep, meaning that also each layer has an internal hierarchical depth due to power-law node degree hierarchy. Nodes with higher degrees in each layer community are also more central and cross-connected in the entire network topology playing a central role (their radial coordinate is smaller) in the latent hyperbolic space which underlies the network topology. It is as the ANN has an ultra-depth that is orthogonal to the regular plan layer depth. The regular plan depth is given by the organization in layers, the ultra-depth is given by the topological centrality of different nodes from different layers in the hyperbolic geometry that underlies the ANN trained by ESUL. Moreover, we present the analysis of the ANP reduction throughout the epochs within the ESUL blocks. From our results, it is evident that the ANP diminishes to a significantly low level after approximately 50 evolution iterations. This finding highlights the efficient network percolation capability of ESUL, while in the meantime it suggests that a prolonged network evolution may not be necessary for ESUL, as the network stabilizes within a few epochs, indicating its rapid convergence.

To evaluate the structural properties of the sparse network, we utilize measures commonly used in network science (Fig. 4B). The details of each measure are explained in the supplementary material, and here we will focus on the observed phenomena. We consider the entire ANN network topology and compare the average values in 5 datasets of CH3-L3 (ESUL) with random (SET) using 4 network topological measures. The plot of power-law gamma indicates that CH3-L3 (ESUL) produces networks with power-low degree distribution ($\gamma \leq 3$), whereas random (SET) does not. We

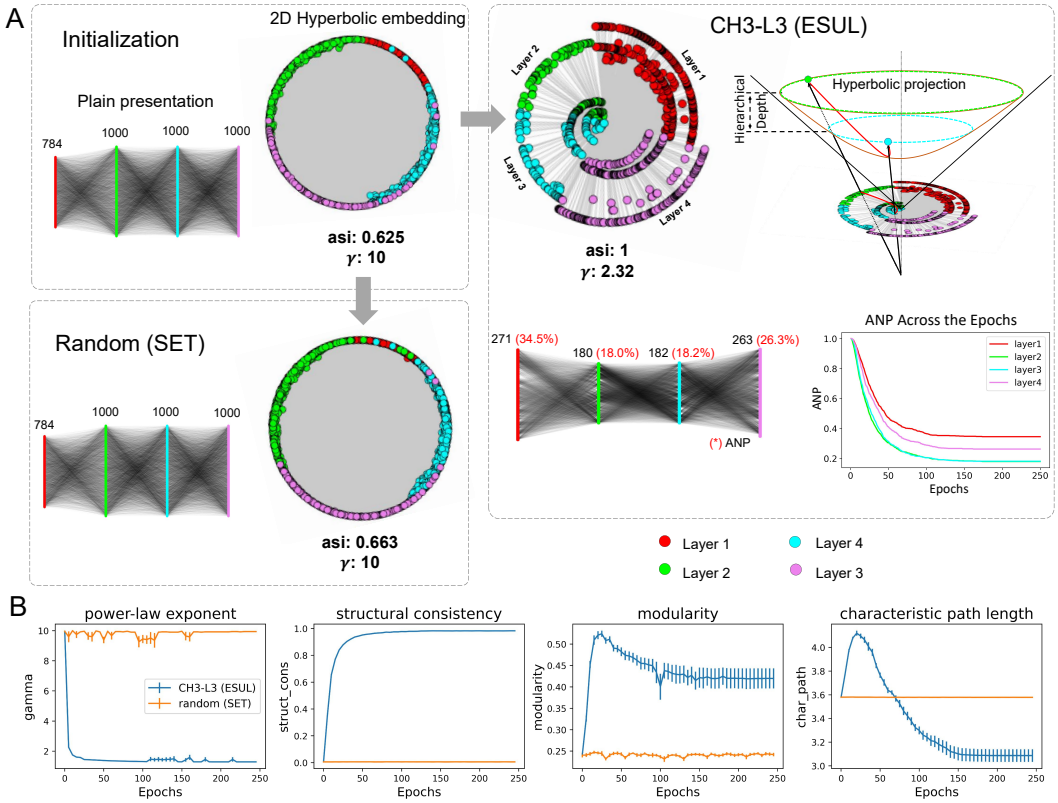


Figure 4: Network science analysis of the sparse ANNs topology. **A** shows the presentation of different statuses of the network, including the network initialized by ER, the network trained by random (final epoch), and the network trained by ESUL (final epoch). The angular separability index (ASI) and power law exponent γ are reported in each panel. Additionally, the active neuron post-percolation rate (ANP) curve across the epochs is reported inside the ESUL panel. **B** shows the 4 network topological measures averaged on 5 different ANNs, trained in 5 datasets by CH3-L3 (ESUL) and random (SET), the shadow area means the standard error of the 5 datasets.

want to clarify that Mocanu et al.[9] also reported that SET could achieve a power-law distribution with MLPs but in their case, they used the learning rate of 0.01 and extended the learning epochs to 1000. We acknowledge that maybe with a higher learning rate and a much longer waiting time, SET can achieve a power-law distribution. However, we emphasize that in Fig. 4B (first panel) ESUL achieves power-law degree distribution regardless of the conditions in 5 different datasets and 2 types (MLPs, VGG16) of network architecture. This is because ESUL regrows connectivity based on the existing network topology according to a brain-network automaton rule, instead SET regrows at random. Furthermore, structure consistency[39] implies that CH3-L3 (ESUL) makes the network more predictable, while modularity shows that it can form distinct obvious communities. The characteristic path length is computed as the average node-pairs length in the network, it is a measure associated with network small-worldness and express also message passing and navigability efficiency of a network[23]. ESUL-trained network topology is ultra-small-world, which happens when a network is small-world with power-law degree exponent lower than 3 (more precisely closer to 2 than 3)[23], and this means that the transfer of information or energy in the network is even more efficient than a regular small-world network[23]. All of these measures give strong evidence that ESUL is capable of transforming the original random network into a complex structured topology that is more suitable for training and potentially a valid lottery-ticket network[8]. This transformation can lead to significant improvements in network efficiency, scalability, and performance.

4.2 Results of CHT

To enhance the performance of ESUL, we propose a 4-step procedure named CHT. We first report the initialization structure in Fig. 2C which is a CSTI example with 2x version of MNIST dataset,

Table 1: Result of CHT comparing to RigL and Fully connected network on 1x and 2x cases

	MNIST(MLP)		**Fashion_MNIST(MLP)**		**EMNIST(MLP)**	
	ACC	AAE	ACC	AAE	ACC	AAE
FC$_{1X}$	98.69±0.02	96.33±0.16	90.43±0.09	87.40±0.02	85.58±0.06	81.75±0.10
RigL$_{1X}$	97.40±0.07	93.69±0.10	88.02±0.11	84.49±0.12	82.96±0.04	78.01±0.06
CHT$_{1X}$	**98.05±0.04**	**95.45±0.05**	**88.07±0.11**	**85.20±0.06**	**83.82±0.04**	**80.25±0.20**
FC$_{2X}$	98.73±0.03	96.27±0.13	90.74±0.13	87.58±0.04	85.85±0.05	82.19±0.12
RigL$_{2X}$	97.91±0.09	94.25±0.03	**88.66±0.07**	85.23±0.08	83.44±0.09	79.10±0.25
CHT$_{2X}$	**98.34±0.08**	**95.60±0.05**	88.34±0.07	**85.53±0.25**	**85.43±0.10**	**81.18±0.15**
	CIFAR10(VGG16)		**CIFAR100(VGG16)**		**Multi30K(Transformer)**	
	ACC	AAE	ACC	AAE	BLEU	AAE
FC$_{1X}$	91.52±0.04	87.74±0.03	66.73±0.06	57.21±0.11	24.0±0.20	20.9±0.13
RigL$_{1X}$	91.60±0.10*	86.54±0.14	**67.87±0.17***	53.80±0.49	21.1±0.08	18.1±0.08
CHT$_{1X}$	**91.68±0.15***	**86.57±0.08**	67.58±0.30*	**57.30±0.20***	**21.3±0.29**	**18.3±0.13**
FC$_{2X}$	91.75±0.07	87.86±0.02	66.34±0.06	57.02±0.04	-	-
RigL$_{2X}$	91.75±0.03	87.07±0.09	**67.88±0.35***	54.08±0.43	-	-
CHT$_{2X}$	**91.98±0.03***	**88.29±0.10***	67.70±0.16*	**57.79±0.08***	-	-

we construct the heatmap of the input layer degrees and node distribution density curve. It can be clearly observed that with CSTI, the links are assigned to nodes mostly associated with input pixels in the center of figures in the MNIST dataset, and indeed the area at the center of the figures is more informative. We also report the CSTI-formed network has an initialized power law exponent $\gamma = 5.4$ which indicates a topology with more hub nodes than in a random network with $\gamma = 10$. To assess the effectiveness of each process (excluding ESUL which was investigated in the previous section) proposed by CHT, we conducted an ablation test and reported the results in the first three plots of Suppl. Fig. 2. These experiments demonstrate the utility of each individual component within the CHT framework.

In Table 1, we report the final performance of CHT compared to the SOTA DST method RigL and fully connected network on 6 datasets and 3 different models. Based on the obtained results, it is evident that CHT outperforms RigL in the majority of cases and consistently achieves faster training. The analysis of running time and FLOPs please refer to the Suppl. Table 3. Furthermore, we achieve comparable results to the Fully Connected (FC) model, with only 1% of the links remaining, on both the MNIST and EMNIST datasets. Moreover, note that while our performance is comparable to RigL on the CIFAR100 dataset, we observe higher AAE in our approach, indicating that CHT exhibits faster learning capabilities. Additionally, as illustrated in Table 2, CHT successfully percolated

Table 2: Active neuron post-percolation rate corresponding to the results in Table 1

	MNIST	**Fashion_MNIST**	**EMNIST**	**CIFAR10**	**CIFAR100**	**Multi-30k**
CHT$_{1X}$	31.15%	30.45%	42.22%	32.52%	32.32%	33.86%
RigL$_{1X}$	97.45%	98.69%	89.19%	94.14%	93.50%	88.7%
CHT$_{2X}$	33.27%	29.25%	39.78%	34.24%	35.71%	-
RigL$_{2X}$	100%	100%	99.82%	99.83%	99.80%	-

the network to a range of 30-40% of the original neurons in every dataset. This implies that, from a logical standpoint, we utilize fewer neurons to achieve similar results as RigL, therefore CHT can generalize and abstract better in the hidden layers of the data information associated with the classification task, providing a new solution to find the Lottery Ticket of ANNs[8]. CHT's result to adaptively percolate the network size providing a minimalistic network modeling that preserves performance with a smaller architecture is a typical solution in line with Occam's razor also named in science as the principle of parsimony, which is the problem-solving principle advocating to search for explanations constructed with the smallest possible set of elements[40]. For instance, in physics, parsimony was an important heuristic in Albert Einstein's formulation of special relativity[41]; Pierre Louis Maupertuis' and Leonhard Euler's work on the principle of least action[42]; and Max Planck,

Werner Heisenberg and Louis de Broglie development of quantum mechanics[43, 44]. On this basis, we can claim that CHT represents the first example of an algorithm for "parsimony sparse training".

References

[1] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[2] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[3] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[4] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2285–2294. JMLR.org, 2015.

[5] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann, 1989.

[6] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143, 2015.

[7] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[9] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[10] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2943–2952. PMLR, 2020.

[11] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6989–7000. PMLR, 2021.

[12] Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34:20838–20850, 2021.

[13] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. pages 9908–9922, 2021.

[14] Simone Daminelli, Josephine Maria Thomas, Claudio Durán, and Carlo Vittorio Cannistraci. Common neighbours and the local-community-paradigm for topological link prediction in bipartite networks. *New Journal of Physics*, 17(11):113037, nov 2015.

[15] Claudio Durán, Simone Daminelli, Josephine M Thomas, V Joachim Haupt, Michael Schroeder, and Carlo Vittorio Cannistraci. Pioneering topological methods for network-based drug–target prediction by exploiting a brain-network self-organization theory. *Briefings in Bioinformatics*, 19(6):1183–1202, 04 2017.

[16] Carlo Vittorio Cannistraci. Modelling self-organization in complex networks via a brain-inspired network automata theory improves link reliability in protein interactomes. *Sci Rep*, 8(1):2045–2322, 10 2018.

[17] A. Muscoloni, U. Michieli, and C.V. Cannistraci. Adaptive network automata modelling of complex networks. *preprints*, 2020.

[18] Geng Yuan, Yanyu Li, Sheng Li, Zhenglun Kong, Sergey Tulyakov, Xulong Tang, Yanzhi Wang, and Jian Ren. Layer freezing & data sieving: Missing pieces of a generic framework for sparse training. *arXiv preprint arXiv:2209.11204*, 2022.

[19] Donald Hebb. The organization of behavior. emphnew york, 1949.

[20] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific reports*, 3(1):1613, 2013.

[21] Vaibhav et al Narula. Can local-community-paradigm and epitopological learning enhance our understanding of how local brain connectivity is able to process, learn and memorize chronic pain? *Applied network science*, 2(1), 2017.

[22] Ming Li, Run-Ran Liu, Linyuan Lü, Mao-Bin Hu, Shuqi Xu, and Yi-Cheng Zhang. Percolation on complex networks: Theory and application. *Physics Reports*, 907:1–68, 2021.

[23] Carlo Vittorio Cannistraci and Alessandro Muscoloni. Geometrical congruence, greedy navigability and myopic transfer in complex networks and brain connectomes. *Nature Communications*, 13(1):7308, 2022.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[25] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[28] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009.

[30] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74. Association for Computational Linguistics, 2016.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[33] Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature communications*, 8(1):1615, 2017.

[34] Alessandro Muscoloni and Carlo Vittorio Cannistraci. Angular separability of data clusters or network communities in geometrical space and its relevance to hyperbolic embedding. *arXiv preprint arXiv:1907.00025*, 2019.

[35] Alessandro Muscoloni and Carlo Vittorio Cannistraci. A nonuniform popularity-similarity optimization (npso) model to efficiently generate realistic complex networks with communities. *New Journal of Physics*, 20(5):052002, 2018.

[36] Claudio Durán, Alessandro Muscoloni, and Carlo Vittorio Cannistraci. Geometrical inspired pre-weighting enhances markov clustering community detection in complex networks. *Applied Network Science*, 6(1):1–16, 2021.

[37] Alberto Cacciola, Alessandro Muscoloni, Vaibhav Narula, Alessandro Calamuneri, Salvatore Nigro, Emeran A Mayer, Jennifer S Labus, Giuseppe Anastasi, Aldo Quattrone, Angelo Quartarone, et al. Coalescent embedding in the hyperbolic space unsupervisedly discloses the hidden geometry of the brain. *arXiv preprint arXiv:1705.04192*, 2017.

[38] Mengqiao Xu, Qian Pan, Alessandro Muscoloni, Haoxiang Xia, and Carlo Vittorio Cannistraci. Modular gateway-ness connectivity and structural core organization in maritime network science. *Nature Communications*, 11(1):2849, 2020.

[39] Linyuan Lü, Liming Pan, Tao Zhou, Yi-Cheng Zhang, and H. Eugene Stanley. Toward link predictability of complex networks. *Proceedings of the National Academy of Sciences*, 112(8):2325–2330, 2015.

[40] Hugh G Gauch Jr, Hugh G Gauch, and Hugh G Gauch Jr. *Scientific method in practice*. Cambridge University Press, 2003.

[41] Albert Einstein. Does the inertia of a body depend upon its energy-content. *Annalen der physik*, 18(13):639–641, 1905.

[42] P. L. M. de Maupertuis. Mémoires de l'académie royale. *p. 423*, 1744.

[43] Roald Hoffmann, Vladimir I Minkin, and Barry K Carpenter. Ockham's razor and chemistry. *Bulletin de la Société chimique de France*, 2(133):117–130, 1996.

[44] L. de Broglie. Annales de physique. *pp. 22–128*, 1925.

[45] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4646–4655. PMLR, 2019.

[46] Christopher L Rees, Keivan Moradi, and Giorgio A Ascoli. Weighing the evidence in peters' rule: does neuronal morphology predict connectivity? *Trends in neurosciences*, 40(2):63–71, 2017.

Supplementary Information

A Algorithm of ESUL

Algorithm 1 Algorithm of Epitological sparse ultra-deep learning

Input: Weight of all layers at epoch n: w_n,
The number of links that should be removed: N
Sparse connectivity predictor: For instance link predictor (LP)
Output: w_{n+1}, T_{n+1}

for each evolutionary epoch n **do**
  for each sandwich layer l **do**
    $T_n^l = ArgTopK(-|w_n^l|, N^l)$,
    $w_n^l = T_n^l * w_n^l$.
  end for
  for each sandwich layer l **backward do**
    Inactive Neurons Removal,
    Incomplete Path Adjustment,
    update $N^l = N^l + IPA\_num^l$.
  end for
  for each sandwich layer l **forward do**
    Inactive Neurons Removal,
    Incomplete Path Adjustment,
    update $N^l = N^l + IPA\_num^l$.
  end for
  for each sandwich layer l **do**
    $T_{n+1}^l = LP(T_n^l, N^l)$,
    Initialize new values $\hat{w}^l$ to the new links predicted by LP,
    $w_{n+1}^l = (T_{n+1}^l - T_n^l) * \hat{w}^l + w_n^l$.
  end for
end for

B Sparse Weight Initialization

Our derivation mainly follows kaiming[24]. Similar to SNIP[7] we introduce auxiliary indicator variables $\mathbf{C} \in \{0,1\}^{m \times n}$. Then For a densely-connected layer, we have:

$$\mathbf{y}_l = \mathbf{C}_l \odot \mathbf{W}_l \mathbf{x}_l + \mathbf{b}_l \tag{4}$$

where $\odot$ denotes the Hadamard product, $\mathbf{W}^{m \times n}$ is the weight matrix, $\mathbf{x}^{n \times 1}$ is the input vector, $\mathbf{b}^{m \times 1}$ is a vector of biases and $\mathbf{y}^{m \times 1}$ is the output vector. We use l to index a layer.

Then for i-th element of the pre-activation layer $\mathbf{y}_l$, by ignoring the influence of bias, we can get its variance:

$$Var[y_l^i] = Var[\sum_{j=1}^{n} c_l^{ij} w_l^{ij} x_l^j] \tag{5}$$

As we initialize elements in $\mathbf{C}$ to be independent and identically distributed random variables, also same for elements in $\mathbf{W}$. We also assume elements in $\mathbf{x}$ are mutually independent and share the same distribution. $\mathbf{C}$, $\mathbf{W}$ and $\mathbf{x}$ are independent of each other. Variance of y will be:

$$Var[y_l^i] = nVar[c_l^{ij} w_l^{ij} x_l^j] = n((Var[c] + \mu_c^2)(Var[w] + \mu_w^2)(Var[x] + \mu_x^2) - \mu_c^2 \mu_w^2 \mu_x^2) \tag{6}$$

where μ is mean. We assume c follows Bernoulli distribution, then the probability mass function is:

$$f(c) = \begin{cases} S & c = 0 \\ 1 - S & c = 1 \end{cases} \tag{7}$$

where S denotes the sparsity. We could infer that $Var[c] = S(1-S)$ and $\mu_c^2 = (1-S)^2$. Same as previous work[24], we define w follow zero-mean Gaussian distribution, therefore $\mu_c^2\mu_w^2\mu_x^2 = 0$. And when activation layer is ReLU, $Var[x_l^i] + \mu_{x_l^i}^2 = \frac{1}{2}Var[y_{l-1}^i]$. Then equation 6 could be changed to:

$$Var[y_l^i] = n(Var[c] + \mu_c^2)(Var[w] + \mu_w^2)(Var[x] + \mu_x^2) = \frac{1}{2}n(1-S)Var[w]Var[y_{l-1}^i] \quad (8)$$

We expect the variance of signal to remain the same across layers $Var[y_l^i] = Var[y_{l-1}^i]$, then:

$$Var[w] = \frac{2}{n(1-S)} \quad (9)$$

We denote $indim$ as n, which represents the dimension of the input layer of each sandwich layer. The constant $gain$ varies depending on the activation function used. For example, when using ReLU, we set $gain$ to $\sqrt{2}$. The resulting values of w are sampled from a normal distribution, resulting in the following expression:

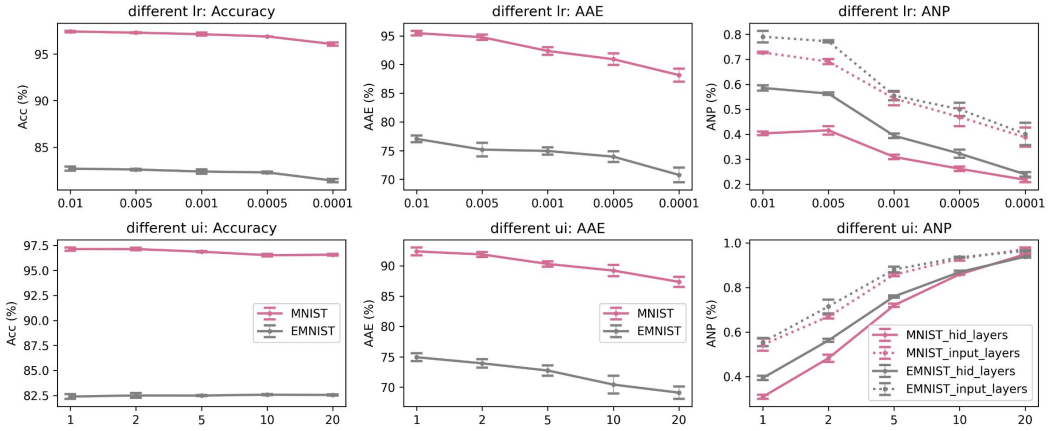
$$SWI(w) \sim \mathcal{N}(0, \sigma^2), \sigma = \sqrt{Var[w]} = \frac{gain}{\sqrt{(1-S) * indim}} \quad (10)$$

C Different properties of current dynamic sparse training methods

Suppl. Table 1: Different properties of current different DST methods and ours

Method	Remove	Regrow	Back Propagation	Fixed Sparsity	Weight Initialization	Weight Update	Topology Initialization
SET[9]	Weight magnitude	Random	Sparse	Yes	kaiming	zero	ER
DSR[45]	Weight magnitude	Random	Sparse	Yes	Not-mentioned	zero	ER
RigL[10]	Weight magnitude	Gradient	Dense	Yes	kaiming	zero	ER
MEST[12]	Weight and gradient magnitude	Random	Sparse	Yes	kaiming	zero	ER
GraNet[13]	Weight magnitude	Random	Sparse	No	kaiming	zero	ER
CHT(Ours)	Weight magnitude + Network percolation	Topological Link Prediction	Sparse	Yes	SWI	SWI	CSTI

We present a comparative analysis between CHT and five prominent DST methods, outlining their distinctions in Suppl. Table 1. It is apparent that the weight magnitude technique is widely utilized for link removal in the early stages of the evolutionary process. Additionally, we employ network percolation, which enables the reduction of the number of active neurons. The regrowth phase includes two previously utilized methods: random-based and gradient-based, along with our proposed topological link prediction-based method. However, gradient-based methods encounter challenges due to their requirement of gradients for all non-existent links and subsequent selection of the top-k based on these gradients, resulting in dense backpropagation. GraNet[13] explores a way to sparsify the backpropagation process while utilizing gradient-based methods, but it introduces a new situation where sparsity cannot be achieved from the outset. GraNet initializes with a dense network and gradually evolves towards the desired sparsity through multiple iterations of weight updates and topology evolution, which incurs higher computational costs compared to other fixed-sparsity DST methods. Additionally, our analysis reveals that all the articles and released code on GitHub repositories predominantly adopt Kaiming weight initialization, zero weight updates, and ER (Erdos-Rényi) for topology initialization. In contrast, we utilize 1) SWI: as demonstrated in Suppl. B, which is more suitable for the initialization and update of sparse networks; 2) CSTI: to provide enhanced topological guidance to the link predictor, results in a structured network topology for each sandwich layer.



Suppl. Figure 1: Comparison of the influence of learning rate and update interval to ESUL. All the experiments are with 3 random seeds.

D Extra experiments and the hyperparameter settings

D.1 Hyperparameter settings of each figure

For the detailed setting please refer to Suppl. Table 2.

Suppl. Table 2: The hyperparameter setting of each figure. The other hyperparameters are fixed in each figure: sparsity=0.99, ζ (fraction of removed links)=0.3, weight_decay=1e-05, batch_size=32. '-' indicates the compared setting in the figures which has been mentioned in either the figure legend or the manuscript.

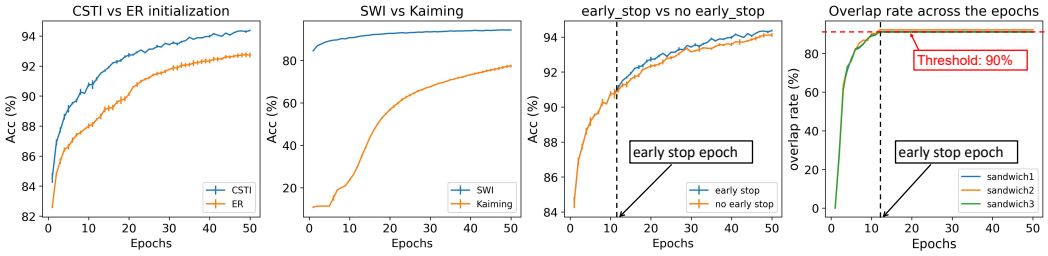
	lr	ui	Weight_initialization	Topology_initialization	Using early_stop
Fig. 3	0.0001	1	Kaiming	ER	No
Fig. 4A & B	0.0001	1	Kaiming	ER	No
Table1	0.01	1	SWI	SCSTI	Yes
Suppl. Fig. 1	-	-	Kaiming	ER	No
Suppl. Fig. 2	0.0001	1	-	-	-
Suppl. Table. 3	0.01	1	SWI	SCSTI	Yes

D.2 The influence of learning rate and update interval to ESUL

We investigate the influence of hyperparameters which are also significant for the training of ESUL. Learning rate, in particular, is a crucial factor that controls the size of the steps taken in each mini-batch during training. In DST, the learning rate is even more dominant. As an example, consider a scenario where weights are assigned to new links with a value of zero. If the learning rate is set to a very small value, these weights may not be able to increase or decrease beyond the positive or negative threshold. Consequently, the weights may get directly pruned in the next round. The update interval (ui) also faces a similar situation. Suppl. Fig. 1 illustrates the effect of learning rate (lr) and update interval (ui) on the performance of ESUL. In the first row, we fix ui=1 and vary lr from [0.01, 0.005, 0.001, 0.0005, 0.0001], while in the second row, we fix lr=0.001 and vary ui from [1, 2, 5, 10, 20]. The conclusion drawn from the figure is that a low learning rate could lead to a more percolated network with lower ANP, indicating the effectiveness of ESUL in reducing the network size. However, if the learning rate is too small, the network may not be able to learn well due to the slow weight update. On the other hand, a larger ui could make the network learn slower and result in a less percolated network. Therefore, we need to find a balance between lr and ui to achieve good performance, fast learning speed, and appropriate network size.

D.3 Ablation test of each exponent in CHT

We evaluate each component introduced (CSTI, SWI, and early stop) in CHT in Suppl. Fig. 2 uses an example of MNIST on the MLP structure. The first three plots compare CSTI vs ER initialization,



Suppl. Figure 2: Ablation test of each component in CHT. The first three figures demonstrate the efficiency of different strategies while keeping the other components fixed and only varying the one being compared. The last figure illustrates the overlap rate of the new links and removed links, providing insight into the necessity of utilizing early stop of the evolutionary process.

SWI vs Kaiming weight initialization, and early stop strategy vs no early stop strategy while keeping the other components fixed. We demonstrate that with structured topology initialization, ESUL can achieve faster learning compared to ER initialization. This can be explained by the theory that link predictors are most effective when the network has already formed certain structural features. Next, we evaluate SWI compared to the commonly used weight initialization method, Kaiming, and the results show that SWI enables faster training from the initial stages. The third panel presents evidence that early stop can prevent the over-evolution of the network topology. Once the network structure becomes stable, running ESUL is unnecessary, and it is more optimal to conduct early stop and focus on refining the weights to achieve convergence. As an example, at the 12th epoch, the overlap rate (the overlap between removed links and new links) exceeds 90%, which we set as the stability threshold, indicating that the network should no longer evolve. Based on this observation, we implement an early stop for the evolutionary process.

In this

D.4 Running time & FLOPs of CHT and RigL

In this section, we compare the running time and FLOPs (Floating Point Operations) of CHT and RigL. We provide the running time for each case, starting from the beginning of training until convergence. For VGG16 and Transformer, we specifically present the FLOPs of the sandwich layers, which utilize DST in this article. The computation of FLOPs follows the same approach as described in the supplementary materials of RigL[10], and we report the ratio of the FLOPs of CHT compared to RigL in Suppl. Table 3. CHT demonstrates faster convergence and lower FLOPs compared to RigL. The reason behind this is that CHT solely relies on the information of topology, whereas RigL relies on global gradient information.

Suppl. Table 3: Running Time & FLOPS of CHT and RigL on 1x and 2x cases

	MNIST	Fashion_MNIST	FLOPS	EMNIST	FLOPS	CIFAR10	FLOPS	CIFAR100	FLOPS	Multi-30k	FLOPS
CHT$_{1X}$	505s	496s	86%	1040s	69%	6003s	81%	11642s	51%	849s	94%
RigL$_{1X}$	972s	846s	-	3124s	-	9145s	-	14763s	-	962s	-
CHT$_{2X}$	592s	531s	90%	948s	77%	8247s	87%	16124s	59%	-	-
RigL$_{2X}$	956s	736s	-	2146s	-	10268s	-	17510s	-	-	-

E Area Across the Epochs

The Area Across the Epochs (AAE) is computed as the cumulative sum of the performance indicator of an algorithm till a certain epoch is divided by the number of epochs. Therefore, AAE is the average performance that an algorithm displays till a certain epoch during training, it is bounded [0,1] and it is an indicator of the learning speed of an algorithm. This means that algorithms whose training curves grow faster have larger AAE and can be considered: "faster learners". For instance, considering the same algorithm and fixing the same number of epochs for each trial, AAE can be used to measure and compare the impact that the tuning of a hyperparameter has on learning speed and to quantify the extent to which this hyperparameter governs learning speed.

$$AAE(A) = \frac{\int_1^E mdm}{E-1} \tag{11}$$

Where A denotes the algorithms that need to be evaluated, E is the number of epochs. M_i denotes the performance of the algorithms in different cases, such as in the classification task, m denotes the accuracy at each epoch, while in the machine translation task, m denotes BLEU.

Pay attention, when we come to comparing different algorithms, there are two important remarks about the way to apply AAE for a fair comparison of learning speed. First, AAE should be applied considering for all different algorithms the same number of epochs and the same hyperparameter settings that can influence learning speed. If an algorithm stops at a number of epochs that is smaller than others then all the algorithms' AAEs will be computed till this smallest epochs value, to make sure that the algorithms are fairly evaluated considering the same interval of epochs. If two algorithms are run using different update intervals of epochs during training, then the comparison is not fair and both algorithms should be run using the same update intervals, and this is valid for any setting that can influence the learning speed. All learning speed settings should be fairly fixed to the same value across methods to ensure that the learning speed difference between algorithms is intrinsic to the sparse training mechanisms and it is not biased by confounding factors. Second, AAE can be only used to compare the learning speed of algorithms inside the same computational experiment and not across different datasets and network architectures, because different datasets or network architectures can generate different learning speeds. However, if the mission of the comparison is to investigate the different learning speeds of the same algorithm with the same hyperparameters settings and network structure, but using different datasets, then using AAE is still valid and can help to determine how the same algorithm might react differently in learning speed because of some variations to the data. An interesting test to try is for instance to investigate the impact of different increasing noise levels on the same type of data considering the same algorithm and setting. This would help to reply to the question of how the noise impacts the learning speed of an algorithm.

F Implementation of VGG16 and Transformer

For MLPs, the correlation between input neurons is easy to compute as the input layer directly consists of pixels from each dataset. This correlation demonstrates the relevance of input information. However, for VGG16 and Transformer models that contain sandwich layers after several modules, it is more reasonable to delay CSTI for several epochs of weight training in order to establish some initial correlation between the inputs.

From a mathematical perspective, given a vector of input variables X_{n*M} and a group of random variables W_{M*M} that follows an independent and identically distributed (i.i.d.) distribution, the output $Y = W * X^T$ is also a set of random variables that follow an i.i.d. distribution. When we directly conduct CSTI on the sandwich layers of VGG16 and Transformer, the information received by these layers is in the form of Y. This implies that CSTI introduces a form of randomization. However, we performed an internal check comparing direct CSTI (CSTI) and delayed CSTI (dCSTI), and the results showed that CSTI produced similar results to dCSTI for VGG16 and Transformer. This suggests that for ESUL, the correlation of the input layers may not be as important as the formation of the hierarchical structure.

Consider the scenario where we have a matrix of inputs, denoted as Y, as mentioned above. We need to compute the correlation between these random variables $Y_1, Y_2, ...$ and select the top-k correlated variables. Intuitively, these variables may appear to be completely random. However, there exists a triangular relationship such that if Y_1 has a high correlation with Y_2 and Y_2 has a high correlation with Y_3, then the correlation between Y_1 and Y_3 may also be high. This creates a random yet hierarchical structure, which can enhance the learning process of DST by providing a better topology.

Therefore, for the sandwich layers in VGG16 and Transformer, we directly compute the correlation based on the information transferred from the previous modules, and then form the structures following the steps introduced in Fig. 2A in the main text.

G Subranking strategy of CH3-L3

Here we describe the sub-ranking strategy adopted by CH3-L3[17] to internally rank all the node pairs that have the same CH score. Although we didn't apply this strategy in the current article due to the ultra-sparse scenario, it can be useful for lower-sparsity DST applications in the future. The sub-ranking strategy aims to provide a more precise ranking for links with identical link prediction scores, particularly those that are tied-ranked and located just above the regrown threshold links.

To implement the sub-ranking strategy, the following algorithmic steps are followed:

- Assign a weight to each link (i, j) in the network, denoted as $w_{i,j} = \frac{1}{1+CH_{i,j}}$.
- Compute the shortest paths (SP) between all pairs of nodes in the weighted network.
- For each node pair (i, j), compute the Spearman's rank correlation ($SPcorr$) between the vectors of all shortest paths from node i and node j.
- Generate a final ranking where node pairs are initially ranked based on the CH score ($CH_{i,j}$), and ties are sub-ranked based on the $SPcorr_{i,j}$ value. If there are still tied scores, random selection is used to rank the node pairs.

While the $SPcorr$ can be replaced with any other link predictor, we adopted this strategy because it aligns with the neurobiological principles underlying the CH model[17]. According to the interpretation of Peters' rule, the probability of two neurons being connected is influenced by the spatial proximity of their respective axonal and dendritic arbors[46]. In other words, connectivity depends on the geometric proximity of neurons. This neurobiological concept is consistent with the $SPcorr$ score and fits within the framework of CH modeling, as a high correlation suggests that two nodes have similar shortest paths to other nodes in the network, indicating spatial proximity due to their close geometric positioning.