SONGs: Self-Organizing Neural Graphs – Supplementary Materials

Łukasz Struski[1] Tomasz Danel[1] Marek Śmieja[1] Jacek Tabor[1]
Bartosz Zieliński[1,2]

[1] Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
[2] IDEAS NCBR, Warsaw, Poland

{lukasz.struski;marek.smieja;jacek.tabor;bartosz.zielinski}@uj.edu.pl
{tomasz.danel}@doctoral.uj.edu.pl

A. SONGs visualization

In this section, we provide the visualization of a SONG trained on MNIST where the nodes are represented by the learned filters and the "average" image passing through those nodes (see Figure 1). Moreover, we provide additional examples of the graph structures obtained by training SONG on the MNIST and CIFAR10 datasets (see Figures 2 and 3) together with the consecutive steps of the Markov process (see Figures 4, 5, 6, and 7).

Finally, we analyze the relationship between BCE loss and the probability of back and cross edges in the successive epochs of the training. We present the mean over multiple models and all test samples (as each test sample x has its graph represented by matrix P_x). The number of back and cross edges is obtained in the following way. We first calculate all paths from the root with a probability higher than a particular threshold 0.0001. Then we create a standard binary directed graph that contains all nodes and edges from those paths. Finally, we run the DFS algorithm for this graph (starting from the root) to obtain backed and cross edges.

B. Theoretical analysis

Let us consider SONG as the probabilistic model over trajectories. A trajectory of length N, starting at the root of SBDG G, is defined as $T = (u_{i_t})_{t=1..N}$ with binary decision $d_t \in \{0,1\}, i_t \in I$, where I denotes the set of node indexes. Thus, our trajectory starts at the root ($i_0 = 0$) and successively passes through nodes $u_{i_{t_1}}, \ldots, u_{i_{t_N}}$. The position of trajectory after time t is defined as $T(t) = u_{i_t}$ and the probability of trajectory T is defined as

$$\mathbf{prob}(T;G) = \prod_{t=1}^{N} (\sigma_{i_{t-1}}^{d_t} \cdot m_{i_t i_{t-1}}^{d_t}).$$

Then the probability of reaching leaf l after N steps with a random trajectory T equals $\mathbf{prob}(T(N) = l \,|\, T \sim G)$, where $T \sim G$ denotes that we sample trajectories with respect to distribution given by $\mathbf{prob}(\cdot; G)$.

Next, we introduce a binarized graph G, where we binarize the connections from any pair of nodes. For a fixed $d \in \{0,1\}$, we denote $G[i,j;d]$ as the graph that makes a decision of moving from u_i to u_j with probability 1.

In the following theorem, we show that if G has no cycles, then we can decompose the probability of its trajectory into the mixture of such binarized graphs.

Theorem B.1. *Let G be a SBDG where the probability of visiting twice an arbitrary node by a trajectory of length N is zero. Moreover, u_i be an internal node, fixed $d \in \{0,1\}$, and an arbitrary trajectory T of length N. Then*

$$\mathbf{prob}(T;G) = \sum_{j=1}^{n} m_{ji}^{d} \mathbf{prob}(T; G[i,j,d]). \qquad (1)$$

Proof. Let $T = (u_{i_t})_{t=1..N}$ be a given trajectory and let us consider three cases of passing trough node u_i. First case assumes that T does not pass through u_i, i.e. $i \neq i_t$ for $t = 1, \ldots, N$. Then, directly from the definition of the trajectory's probability

$$\mathbf{prob}(T;G) = \mathbf{prob}(T; G[i,j,d]), \text{ for an arbitrary } j.$$

This completes the proof of (1) in this case. Hence, let us now consider the cases where the trajectory T passes through node u_i.

Suppose the second case, when T passes through u_i more than once. In this case, we will show that both the left and right sides of (1) are zero. Obviously, $\mathbf{prob}(T;G) = 0$ follows directly from the assumption that the probability of visiting twice an arbitrary node by a trajectory of length N in G is zero. Assume, for an indirect proof that there exist j such that $m_{ji}^{d} \mathbf{prob}(T; G[i,j,d]) > 0$. Then $m_{ji}^{d} > 0$. Moreover, if T passes though u_i and makes a decision d, then it has to move to u_j. In consequence,

$$\mathbf{prob}(T;G) = m_{ji}^{d} \mathbf{prob}(T; G[i,j,d]) > 0$$

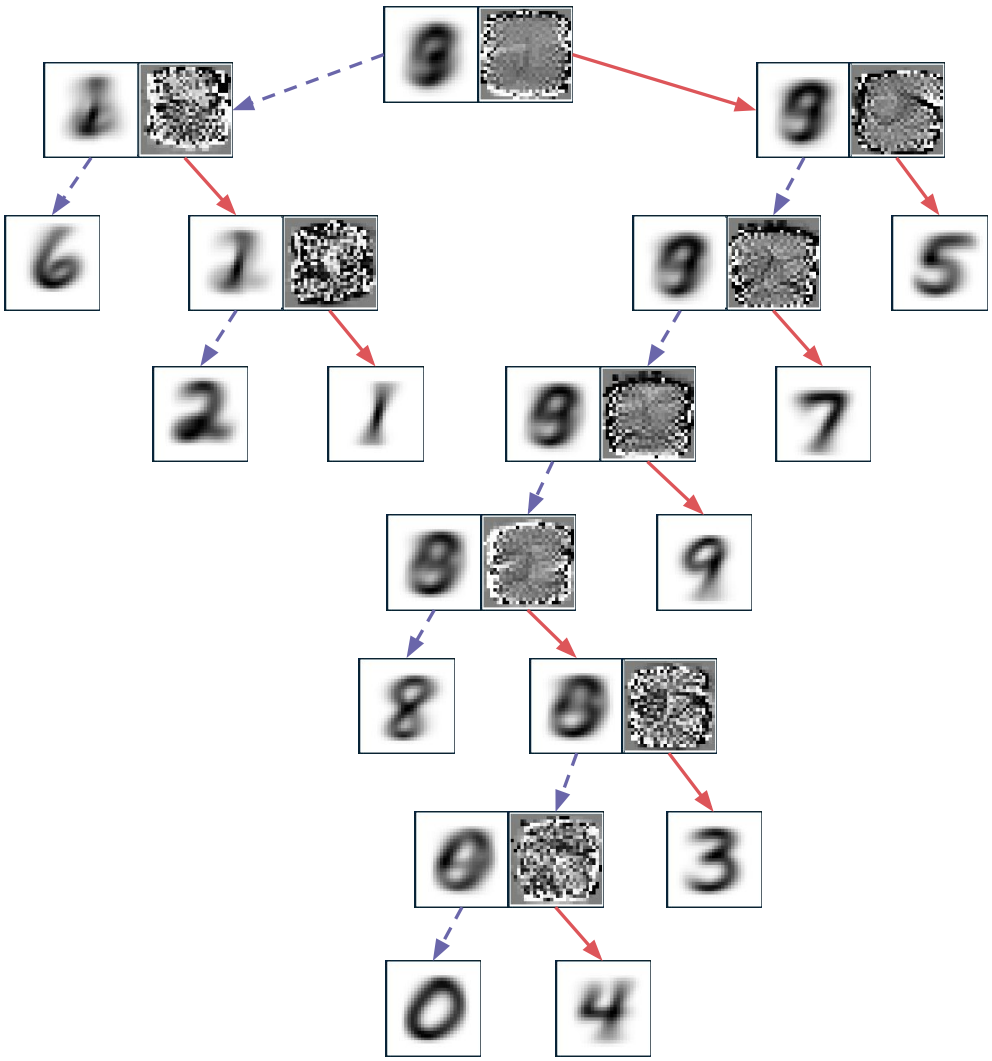


Figure 1: Visualization of a shallow SONG (SONG-S) trained on MNIST where the nodes are represented by the learned filters and the "average" image passing through those nodes (corresponding to the right and left side of each node, respectively). Notice that SONGs contain filters only in the inner nodes, as it is impossible to move out from the leaves.

is a contradiction.

Let us consider the remaining, third case, when T passes through u_i only once, and makes a decision d. In other words, there exists a unique t such that $i_t = i$ and $d_t = d$. Observe that if $j = i_{t+1}$ then we move from u_i to u_j and all the probabilities $\mathbf{prob}(T; G[i, l, d]) = 0$, for $l \neq j$. Moreover, since T visits u_i only once, we get $\mathbf{prob}(T; G) = m^d_{ji}\mathbf{prob}(T; G[i, j, d])$, which completes the proof. □

We now show the consequences of the above theorem for the SONG model. For this purpose, we assume that $X = (x_i)_{i=1..K}$ where each x_i is associated with a label y_i. We also consider SONG $\mathcal{G}$ trained on X for trajectories of length N. Thus for each pair (x, y), we define the probability that a random trajectory of length N reaches leaf corresponding to y as $\mathbf{prob}(T(N) = y \,|T \sim \mathcal{G}_x)$.

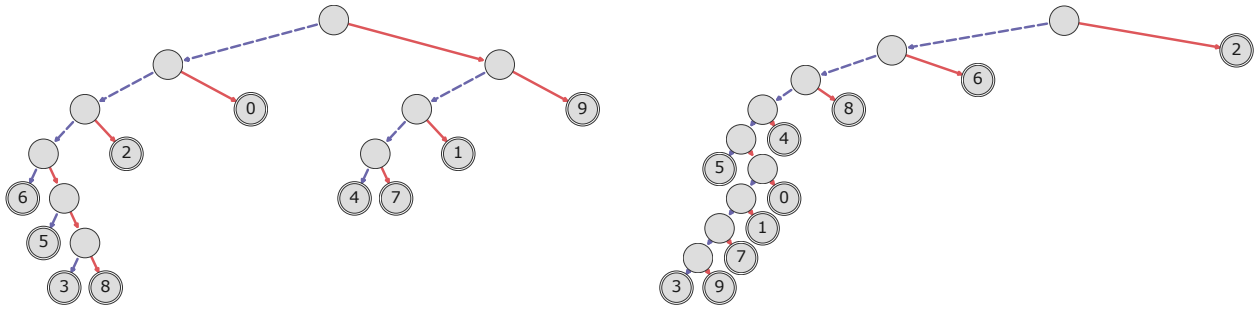
In the following theorem, we show that if SONG is trained with zero CE or BCE loss, then no trajectory of length N in $\mathcal{G}_x$ visits the same internal node twice with nonzero probability.

Theorem B.2. *Let us consider SONG classifier with N moves and x being a data point with class y, such that* $\mathrm{loss}\big(\mathbf{prob}(T(N) = y \,|T \sim \mathcal{G}_x), y\big) = 0$.

Then no trajectory of length N in $\mathcal{G}_x$ visits the same internal node twice with nonzero probability.

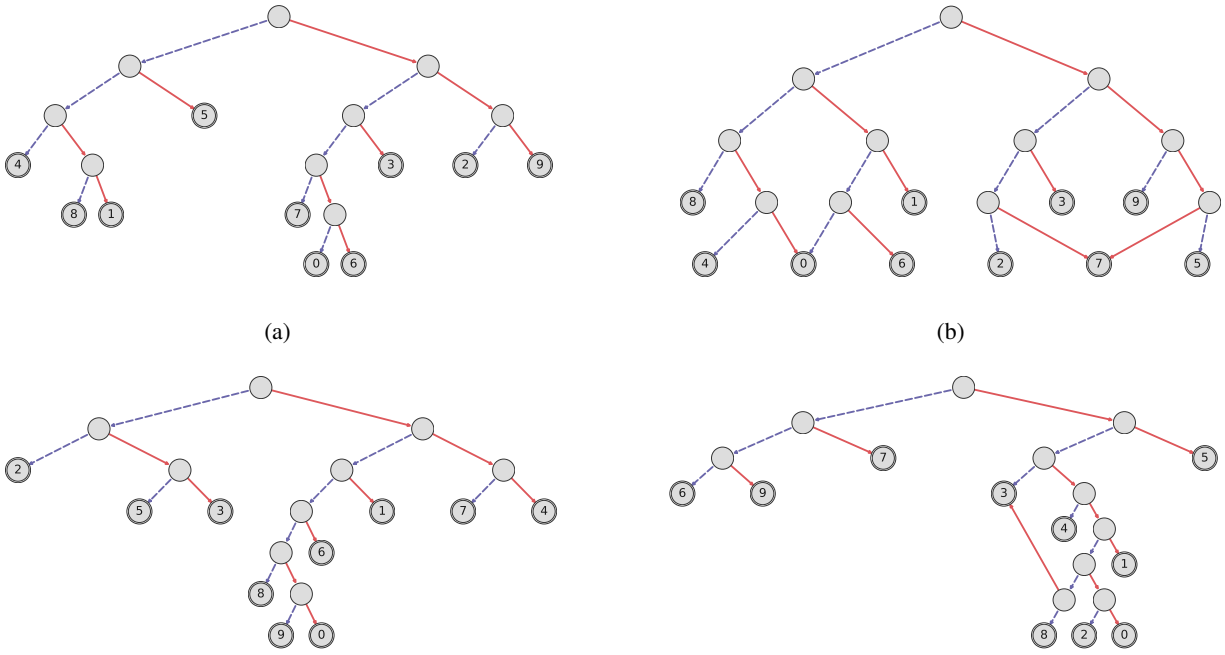
Proof. First observe, that directly from the fact that both CE and BCE are non-negative, $\mathrm{loss}\big(\mathbf{prob}(T(N) = y \,|T \sim \mathcal{G}_x), y\big) = 0$ iff

$$\mathbf{prob}(T(N) = y \,|T \sim \mathcal{G}_x) = 1.$$



(a) (b)

Figure 2: Examples of the graph structures obtained by training SONG on the MNIST dataset. The root is the top-most node in each graph, and the leaves are denoted by double node borders. The numbers on the leaves are the MNIST classes. For each node v_i, we present two edges corresponding to the highest probability from two transition vectors $m^0_{\cdot i}$ and $m^1_{\cdot i}$ (represented as dashed blue and solid red arrows, respectively).



(a) (b)

(c) (d)

Figure 3: Examples of the graph structures obtained by training SONG on the CIFAR10 dataset. The root is the top-most node in each graph, and the leaves are denoted by double node borders. The numbers on the leaves are the CIFAR10 classes. For each node v_i, we present two edges corresponding to the highest probability from two transition vectors $m^0_{\cdot i}$ and $m^1_{\cdot i}$ (represented as dashed blue and solid red arrows, respectively).

Now suppose that there exists a trajectory T with nonzero probability, which goes through a given internal node u twice, i.e. $T(t_1) = T(t_2) = v$ for $t_1 < t_2$. Observe that $T(t)$ is not a leaf for $t \in [t_1, t_2]$, since after reaching the leaf, we stay in it. Consider the trajectory $\tilde{T}$ given by

$$\tilde{T}(t) = \begin{cases} T(t) & \text{if } t \leq t_1, \\ T(t_1 + s) & \text{if } t = t_1 + l(t_2 - t_1) + s, \\ & \quad l \in \mathbb{N}, s \in \{0, .., t_2 - t_1\} \end{cases} .$$

In other words, this is a trajectory that forms a cycle after reaching u. Thus we does not end in a leaf with nonzero probability, which leads to a contradiction. □

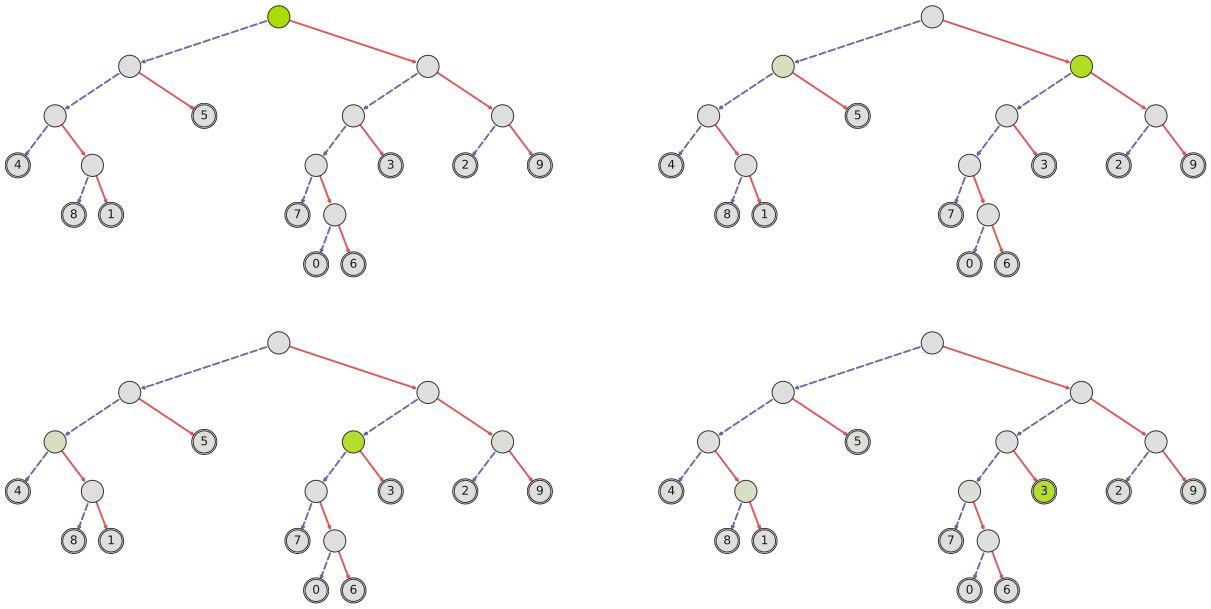


Figure 4: An input image passing through a SONG trained on CIFAR10. High saturation of the green color denotes high probability in the node. Each graph represent a consecutive step of the inference (from left to right, then top to bottom). For each node v_i, we present two edges corresponding to the highest probability from two transition vectors $m^0_{\cdot i}$ and $m^1_{\cdot i}$ (represented as dashed blue and solid red arrows, respectively).

The accuracy of $\mathcal{G}$ over set X is defined as the probability of predicting the correct class

$$\mathrm{acc}(\mathcal{G}; X) = \frac{1}{K}\sum_{i=1}^{K} \mathbf{prob}(T(N) = y_i \,|T \sim \mathcal{G}_{x_i}).$$

As a direct consequence of Theorem B.1, we formulate the following fact.

Theorem B.3. *Let $\mathcal{G}$ be a SONG. We assume that for every $x \in X$ no trajectory in $\mathcal{G}_x$ of length N that visits twice the same internal node with nonzero probability. Let a node index $i \in \{1, \ldots, n\}$ and $d \in \{0, 1\}$ be fixed. Then*

$$\mathrm{acc}(\mathcal{G}; X) = \sum_{j=1}^{n} m^d_{ji}\mathrm{acc}(\mathcal{G}[i, j, d]; X).$$

Proof. By Theorem B.1, for an arbitrary point $x \in X$ (with class y) and trajectory of length N, we have

$$\mathbf{prob}(T; \mathcal{G}_x) = \sum_{j=1}^{n} m^d_{ji}\mathbf{prob}(T; \mathcal{G}_x[i, j, d]).$$

In consequence,

$$\mathbf{prob}(T(N) = y \,|T \sim \mathcal{G}_x) = \\ = \sum_{j=1}^{n} m^d_{ji}\mathbf{prob}(T(N) = y \,|T \sim \mathcal{G}_x[i, j, d]).$$

Averaging the above probability over all points from X and applying the definition of accuracy, we obtain the assertion of the theorem. □

Observe that the above theorem implies that if we discretize connections in the graph by applying formula (2) (below), then we do not decrease the accuracy of the model (statistically, we increase it):

Theorem B.4. *Let $\mathcal{G}_x$ be SONG generated for $x \in X$ with CE or BCE loss equals zero. Moreover, let node index $i \in I$ and $d \in \{0, 1\}$ be fixed, and*

$$j = \arg\max_{\tilde{j}} \mathrm{acc}(\mathcal{G}[i, \tilde{j}, d]; X). \tag{2}$$

Then

$$\mathrm{acc}(\mathcal{G}; X) \leq \mathrm{acc}(\mathcal{G}[i, j, d]; X).$$

Proof. From Theorem B.2 we obtain that $\mathcal{G}_x$ is SONG generated for $x \in X$ with no trajectory of length N that visits twice the same point with nonzero probability. Theorem B.3 implies that if we discretize connections in the graph by applying formula (2), then we do not decrease the accuracy of the model. □

C. Ablation study on leaves regularization

In Figure 8, we present a comparison between SONG trained on MNIST dataset with (a) and without L_{leaves} regularization (b). The accuracy and BCE loss reported at the final stage of training are similar for both models. However, there are significant differences between their convergence

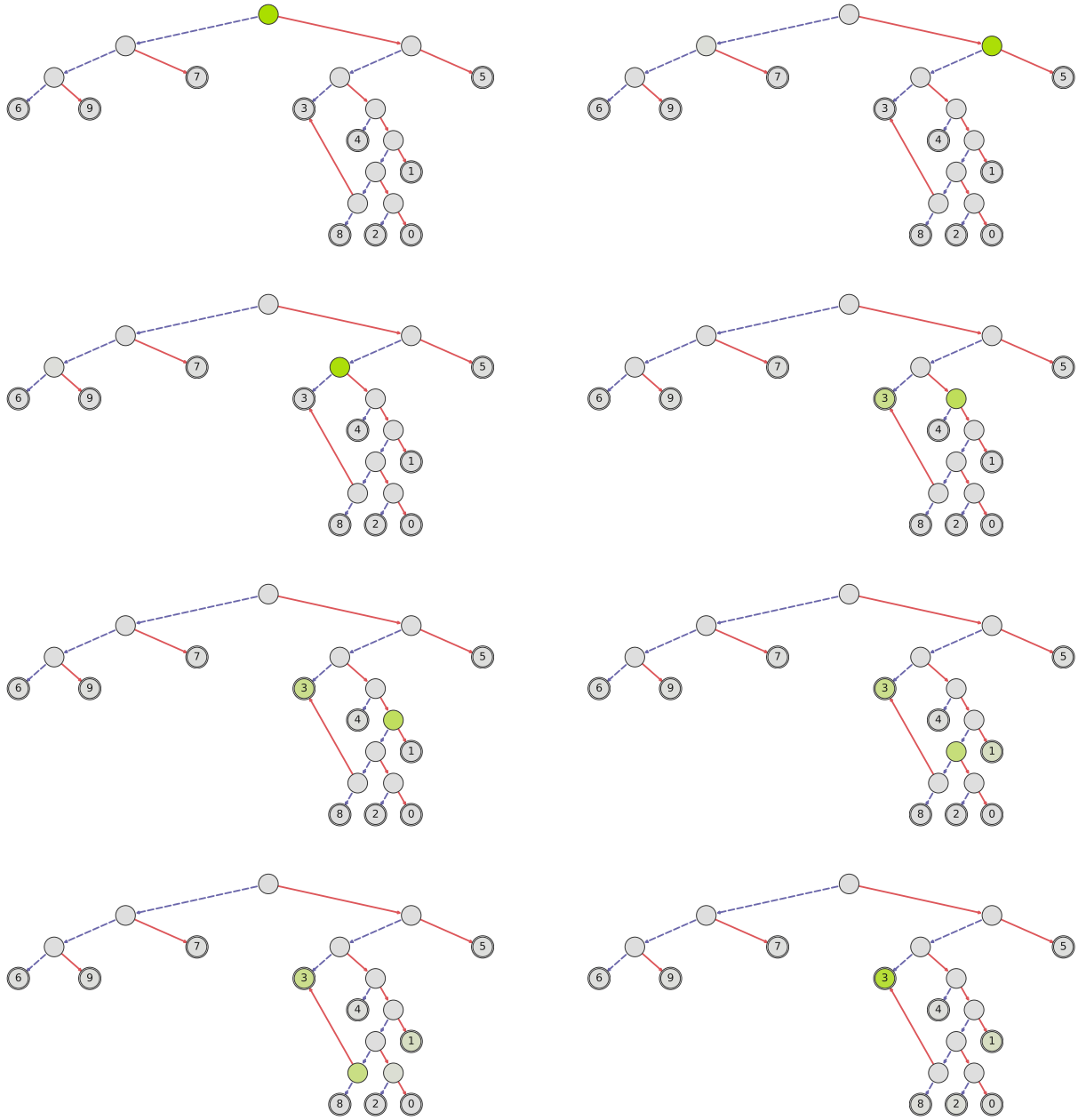


Figure 5: An input image passing through a SONG trained on CIFAR10. High saturation of the green color denotes high probability in the node. Each graph represent a consecutive step of the inference (from left to right, then top to bottom). For each node v_i, we present two edges corresponding to the highest probability from two transition vectors $m^0_{\cdot i}$ and $m^1_{\cdot i}$ (represented as dashed blue and solid red arrows, respectively).

times. Most interestingly, models with regularization hold L_{leaves} close to 0 during the whole training, so the sum of probability in the leaves is close to 1 all the time. On the other hand, the models without regularization have an increased value of L_{leaves} between 50 and 150 epoch, meaning that the leaves are not reached for some of the input samples. Such behavior can be especially detrimental for larger datasets that require more training epochs to converge.

D. Nodes and edges statistics

Here, we show the nodes and edges statistics calculated for SONGs trained on the MNIST dataset (see Figure 9,

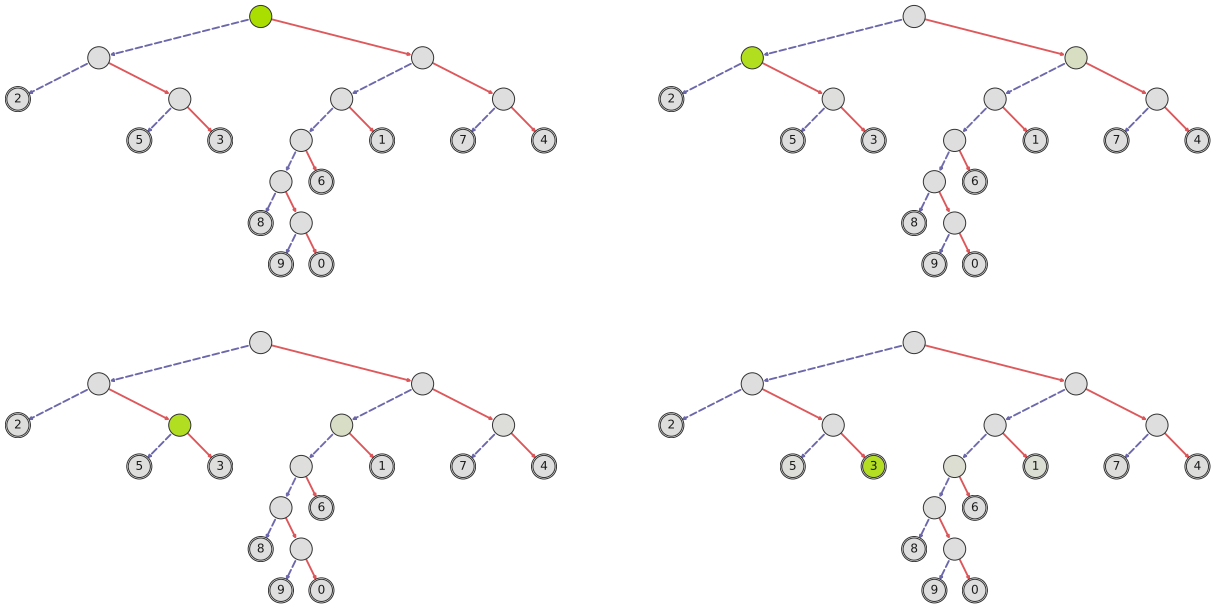


Figure 6: An input image passing through a SONG trained on CIFAR10. High saturation of the green color denotes high probability in the node. Each graph represent a consecutive step of the inference (from left to right, then top to bottom). For each node v_i, we present two edges corresponding to the highest probability from two transition vectors $m^0_{\cdot i}$ and $m^1_{\cdot i}$ (represented as dashed blue and solid red arrows, respectively).

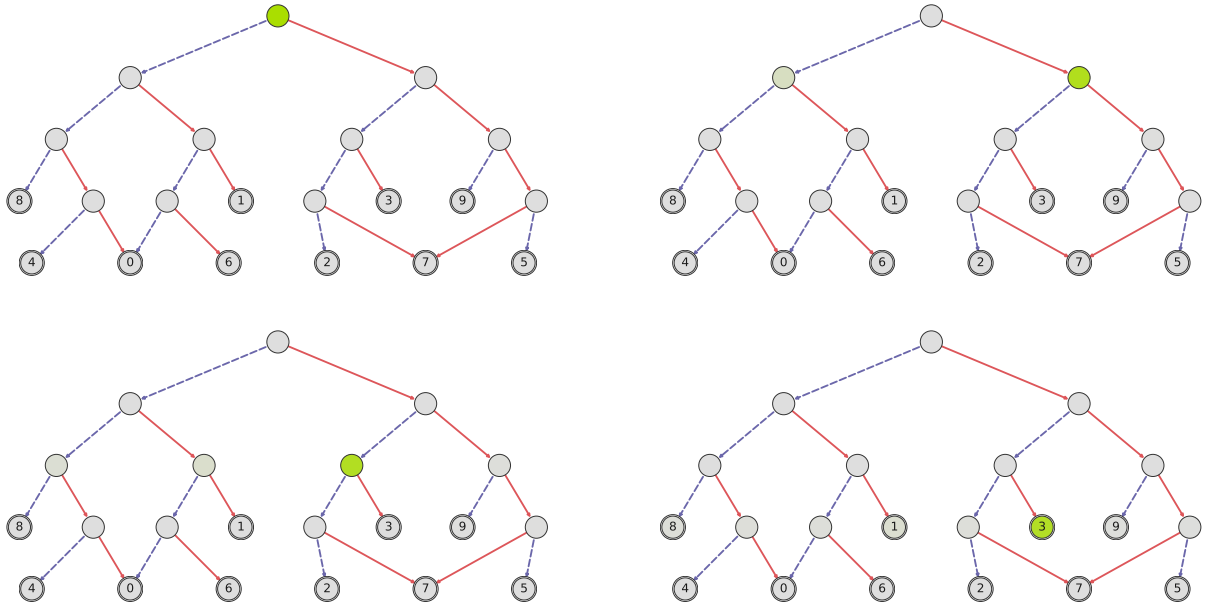
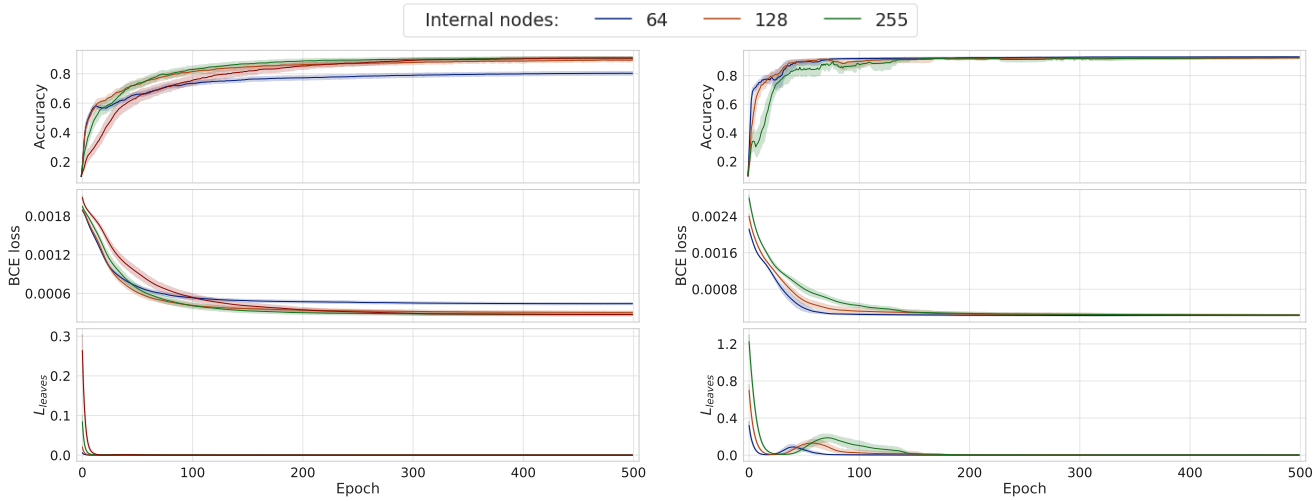


Figure 7: An input image passing through a SONG trained on CIFAR10. High saturation of the green color denotes high probability in the node. Each graph represent a consecutive step of the inference (from left to right, then top to bottom). For each node v_i, we present two edges corresponding to the highest probability from two transition vectors $m^0_{\cdot i}$ and $m^1_{\cdot i}$ (represented as dashed blue and solid red arrows, respectively).



(a) SONG trained with L_{leaves} regularization.

(b) SONG trained without L_{leaves} regularization.

Figure 8: Accuracy, BCE loss, and L_{leave} in the successive training epochs of SONG trained on the MNIST dataset. Each color represents a different number of internal nodes (64, 128, 255), and each line corresponds to mean and standard deviation over multiple training repetitions.

respectively). It is discussed in the article.

E. Additional results

Tables 1, and 2 show the relationship between the number of nodes and steps and prediction accuracy.

F. Transition matrices

In Figures 10-14, we present sample matrices M^0 and M^1 before and after training. One can observe that at the beginning, there are weak connections between all nodes. However, trained matrices are almost binary and usually contain one value close to 1 in each column, and all other values are close to 0.

G. Experimental setup

We used the following datasets in our experiments:

- Letter (`https://archive.ics.uci.edu/ml/datasets/Letter+Recognition`),
- Connect4 (`http://archive.ics.uci.edu/ml/datasets/connect-4`),
- MNIST (published under CC BY-SA 3.0 license),
- CIFAR 10 & CIFAR 100 (published under MIT license),
- TinyImageNet (`https://www.kaggle.com/c/tiny-imagenet/data`).

nodes	steps	base	finetune	reset
9	10	97.95	98.43	98.67
16	8	98.23	**98.81**	98.66
32	8	98.35	98.61	98.81
32	10	98.65	98.52	98.71
64	20	98.68	98.63	98.72

(a) MNIST.

nodes	steps	base	finetune	reset
9	10	94.94	94.98	95.26
16	6	94.95	95.09	95.47
32	6	94.95	95.12	**95.62**
64	10	94.94	95.03	95.41

(b) CIFAR10.

Table 1: Results obtained for selected models from Table 3 in the main paper ("base") and their finetuned versions. We analyze two types of finetuning, either by using basis weights and finetune all the parameters of the network ("finetune") or by taking the graph structure from the base model, reset other network parameters, and train the network from scratch ("reset"). One can observe that there is no obvious winning strategy, and it should be considered a hyperparameter. Notice also that we bold the performance reported in the main paper.

Moreover, we consider two types of setups, deep (SONG) and shallow (SONG-S). In SONG, we build neural networks

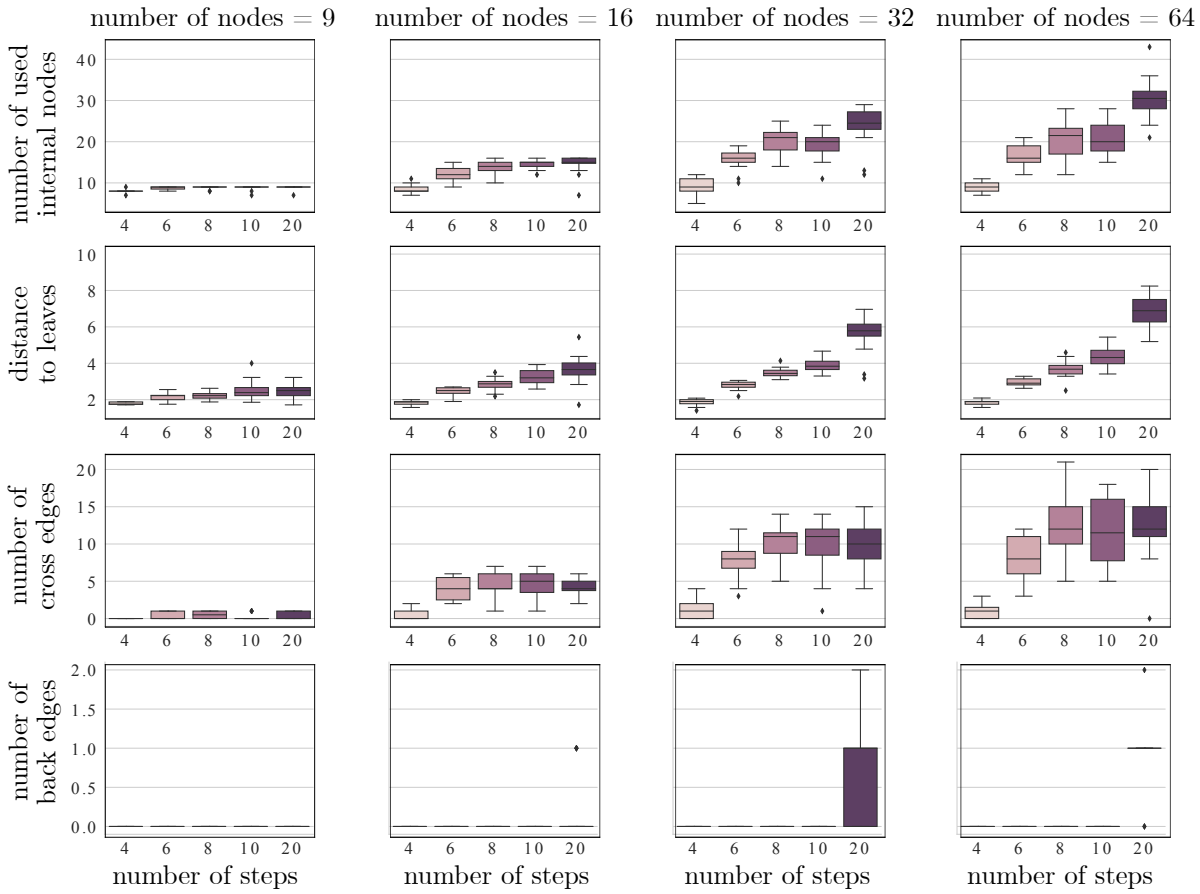


Figure 9: Nodes and edges statistics calculated for SONGs trained on the MNIST dataset. For each combination of the number of internal nodes and steps, 20 graphs are trained and used to plot the distributions of four statistics. One can observe a significant difference in SONG structure depending on those hyperparameters.

that contain two successive parts, CNN and a graph. For the MNIST dataset, the CNN is built from two convolution layers with 8 and 16 filters of size 5×5, each followed by ReLU and 2×2 max pooling. Finally, a linear layer returns representation vectors of dimension 50. For other datasets (CIFAR10, CIFAR100, and TinyImageNet), we use model ResNet18 without the last linear layer. At the same time, for SONG-S, we only flatten the input sample to a one-dimensional vector.

For SONGs, we apply a similar experimental setup as in the state-of-the-art methods to have comparable results. More precisely, we take the previously trained ResNet18 network, remove its last layer, and use the remaining part as a CNN part. For the MNIST data, we train the first part directly using Binary Cross Entropy (BCE) loss. For the remaining datasets, we take a model from `github.com/alvinwan/neural-backed-decision-trees` (published under MIT license) trained with Cross Entropy (CE) and finetune it using BCE loss. During training the SONG, weights of CNNs are frozen. Moreover, the following hyper-parameters are considered in the grid-search:

- For MNIST and CIFAR10:
  - the number of nodes: 9, 16, 32, 64,
  - the number of steps: 4, 6, 8, 10, 20.
- For CIFAR100:
  - the number of nodes: 99, 256, 512,
  - the number of steps: 7, 12, 20, 40.
- For TinyImagenet200:
  - the number of nodes: 512,
  - the number of steps: 20, 40.

Additionally, we consider a batch size 64 or 128 and the learning rate 0.001 for all datasets. Finally, when it comes to initialization, M^0, M^1, and biases in nodes are initialized from a uniform distribution on the interval $[0, 1]$, and the

nodes	steps					
	5	10	20	30	40	50
25	52.65	63.45	62.90	63.85	67.65	68.55
32	53.65	62.65	72.90	73.30	73.20	73.55
64	57.95	74.00	78.70	79.70	82.95	82.95
128	57.00	73.85	79.60	83.05	84.45	85.75
511	48.75	72.35	81.60	82.50	84.05	**86.25**

(a) Letter.

nodes	steps		
	2	5	10
2	77.47	77.40	77.50
8	75.37	79.60	80.27
16	75.47	80.31	81.55
32	75.36	80.45	82.65
255	75.43	80.43	**82.82**

(b) Connect4.

nodes	steps							
	4	6	8	10	20	30	40	50
9	87.58	88.68	88.52	88.93	89.36	90.48	90.36	90.40
16	90.74	91.73	93.06	93.09	93.42	92.97	93.39	93.37
32	88.80	91.47	93.22	93.56	94.38	93.67	93.72	93.56
64	86.35	92.77	93.33	93.41	94.66	94.29	94.86	94.55
128	90.10	93.11	93.65	94.15	94.58	94.80	94.99	94.97
255	90.05	93.11	93.80	93.88	94.28	94.75	95.43	**95.74**

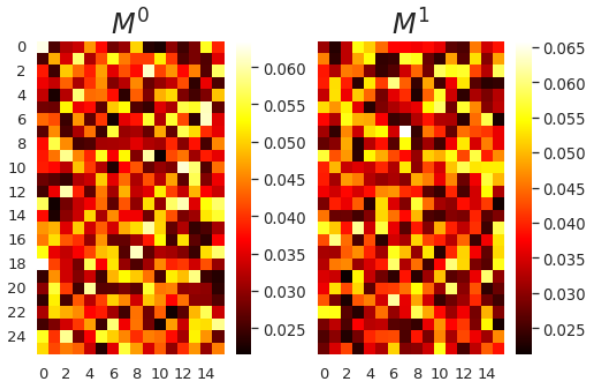
(c) MNIST.

Table 2: SONG as a shallow model (SONG-S). One can observe that the performance increases with the increasing number of nodes and steps for all datasets. We bold the performance reported in the main paper.

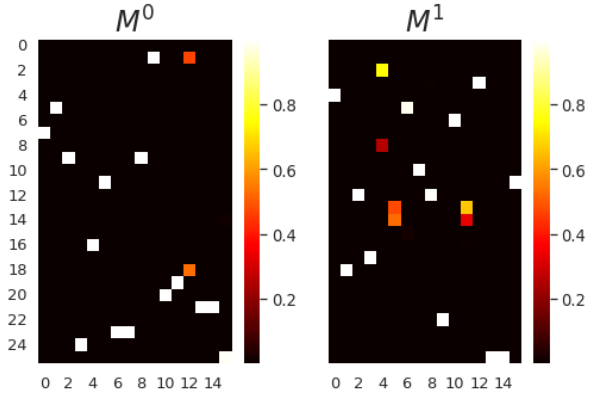
remaining parameters (filters in the nodes) use the Kaiming initialization.

For SONG-S, the following hyper-parameters are considered in the grid-search:

- For Letter dataset:
  - the number of nodes: 25, 32, 64, 128, 511,
  - the number of steps: 5, 10, 20, 30, 40, 50.
- For Connect4 dataset:
  - the number of nodes: 2, 8, 16, 32, 255,
  - the number of steps: 2, 5, 10.
- For MNIST dataset:



(a) Initial values of M^0, M^1.



(b) Trained values of M^0, M^1.

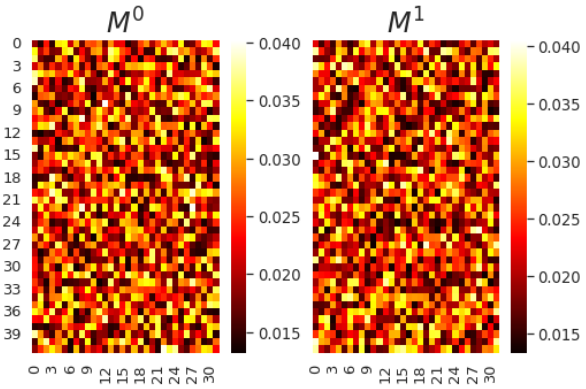
Figure 10: Sample matrices M^0 and M^1 of the SONG before and after training on the MNIST dataset with 16 internal nodes.

  - the number of nodes: 9, 16, 32, 64, 128, 256,
  - the number of steps: 4, 6, 8, 10, 20, 30, 40, 50.

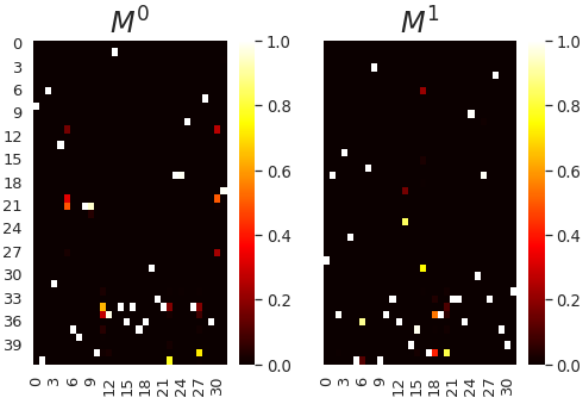
The remaining hyper-parameters are similar to the SONG setup.

H. Computation time and resources

We have run our experiments on Nvidia V100 32GB GPUs of our internal cluster. For deep setup, we trained 50, 50, 25, and 10 models for MNIST, CIFAR10, CIFAR100, and TinyImageNet, respectively. Each model required around 2, 2, 6, and 10 hours, respectively. For the shallow setup, we trained 60, 30, and 96 models for Letter, Connect4, and MNIST, respectively. In this case, each model required around 5, 2, and 2 hours, respectively.

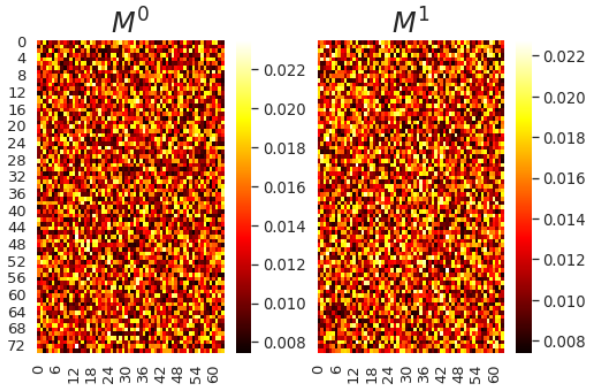


(a) Initial values of M^0, M^1.

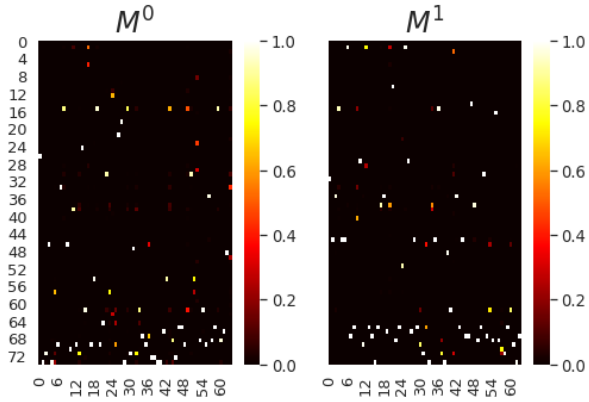


(b) Trained values of M^0, M^1.

Figure 11: Sample matrices M^0 and M^1 of the SONG before and after training on the MNIST dataset with 32 internal nodes.

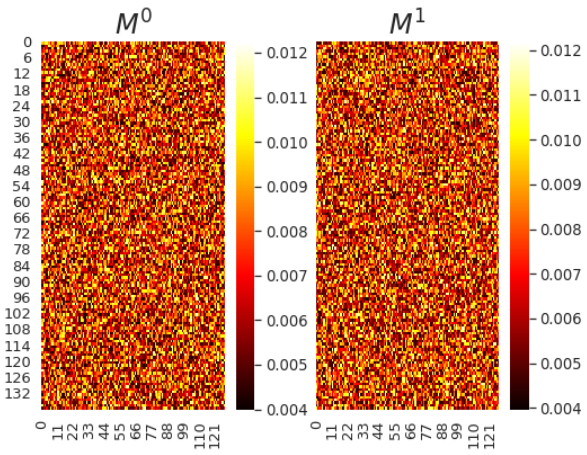


(a) Initial values of M^0, M^1.

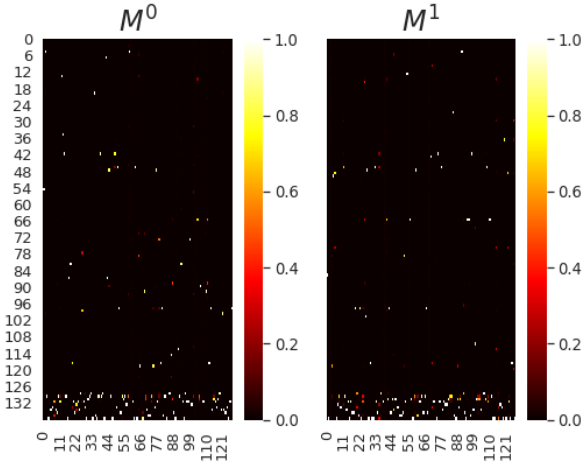


(b) Trained values of M^0, M^1.

Figure 12: Sample matrices M^0 and M^1 of the SONG before and after training on the MNIST dataset with 64 internal nodes.

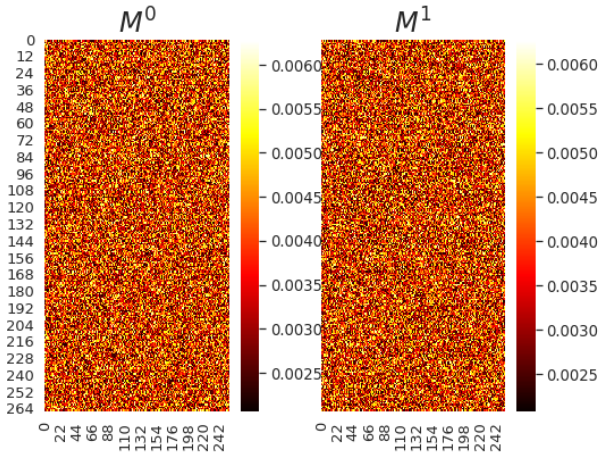


(a) Initial values of M^0, M^1.

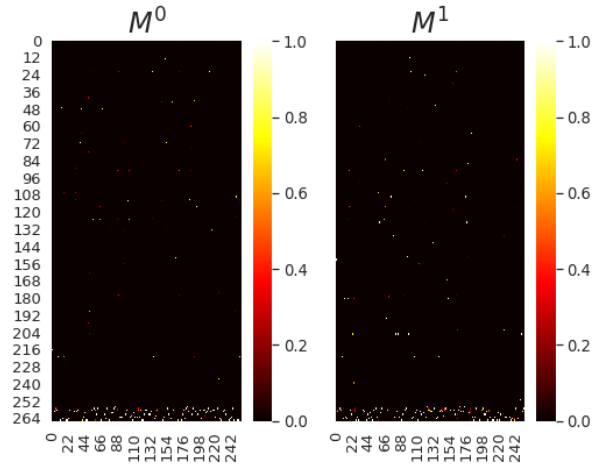


(b) Trained values of M^0, M^1.

Figure 13: Sample matrices M^0 and M^1 of the SONG before and after training on the MNIST dataset with 128 internal nodes.



(a) Initial values of M^0, M^1.



(b) Trained values of M^0, M^1.

Figure 14: Sample matrices M^0 and M^1 of the SONG before and after training on the MNIST dataset with 256 internal nodes.