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 $T = (u_i)_{t=1..N}$ 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 $u_{i_1}, \ldots, u_{i_N}$. The position of trajectory after time $t$ is defined as $T(t) = i_t$ and the probability of trajectory $T$ is defined as

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

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

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

$$
\text{prob}(T; G) = \sum_{j=1}^{n} m_{j}^{d_1} \text{prob}(T; G[i, j, d]) .
$$

Proof. Let $T = (u_i)_{t=1..N}$ 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

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

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, $\text{prob}(T; G) = 0$ 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 $m_{ji}^{d_1} \text{prob}(T; G[i, j, d]) > 0$. Then $m_{ji}^{d_1} > 0$. Moreover, if $T$ passes through $u_i$ and makes a decision $d$, then it has to move to $u_j$. In consequence,

$$
\text{prob}(T; G) = m_{ji}^{d_1} \text{prob}(T; G[i, j, d]) > 0
$$

This completes the proof of (1) in this case.
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 $\text{prob}(T; G[i, l, d]) = 0$, for $l \neq j$. Moreover, since $T$ visits $u_i$ only once, we get $\text{prob}(T; G) = m_{ji} \text{prob}(T; G[i, j, d])$, 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 $G$ 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 $\text{prob}(T(N) = y | T \sim G_x)$.

In the following theorem, we show that if SONG is trained with zero CE or BCE loss, then no trajectory of length $N$ in $G_x$ 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 the loss $\text{loss}(\text{prob}(T(N) = y | T \sim G_x), y) = 0$. Then no trajectory of length $N$ in $G_x$ 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, $\text{loss}(\text{prob}(T(N) = y | T \sim G_x), y) = 0$ iff

$$\text{prob}(T(N) = y | T \sim G_x) = 1.$$
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 \( m^0_i \) and \( m^1_i \) (represented as dashed blue and solid red arrows, respectively).

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 \( m^0_i \) and \( m^1_i \) (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

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

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.
The accuracy of $G$ over set $X$ is defined as the probability of predicting the correct class

$$\text{acc}(G; X) = \frac{1}{K} \sum_{i=1}^{K} \text{prob}(T(N) = y_i | T \sim G_{x_i}).$$

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

**Theorem B.3.** Let $G$ be a SONG. We assume that for every $x \in X$ no trajectory in $G_x$ 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

$$\text{acc}(G; X) = \sum_{j=1}^{n} m_{ji}^d \text{acc}(G[i, j, d]; X).$$

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

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

In consequence,

$$\text{prob}(T(N) = y | T \sim G_x) = \sum_{j=1}^{n} m_{ji}^d \text{prob}(T(N) = y | T \sim G_x[i, j, d]).$$

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 $G_x$ 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

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

Then

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

**Proof.** From Theorem B.2 we obtain that $G_x$ 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 $L_{\text{leaves}}$ 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.
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 $m_{0i}$ and $m_{1i}$ (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 $m_{0i}$ and $m_{1i}$ (represented as dashed blue and solid red arrows, respectively).
Figure 8: Accuracy, BCE loss, and $L_{\text{leaves}}$ 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),

Moreover, we consider two types of setups, deep (SONG) and shallow (SONG-S). In SONG, we build neural networks
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 $5 \times 5$, each followed
by ReLU and $2 \times 2$ 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

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.
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.

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

Figure 12: Sample matrices $M^0$ and $M^1$ of the SONG before and after training on the MNIST dataset with 64 internal nodes.
Figure 13: Sample matrices $M^0$ and $M^1$ of the SONG before and after training on the MNIST dataset with 128 internal nodes.

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