Towards distillation guarantees under algorithmic alignment

Thien Le*
SEAS
Harvard University
Cambridge, MA
thien_le@seas.harvard.edu

Melanie Weber
SEAS
Harvard University
Cambridge, MA
mweber@seas.harvard.edu

Abstract

Distillation is the process of condensing learnt knowledge from a large neural network trained on large datasets to a more efficient one suitable for deployment. Building on recent developments in the learning theory of distillation (Boix-Adsera, 2024), we rigorously analyze a phenomenon in which if the target class of the distillation process is algorithmically aligned with the task at hand, in terms of a linear representation hypothesis (Elhage et al., 2022), then the distillation process can be efficient. This gives rise to a novel and rigorous characterization of algorithmic alignment that could be of independent interest.

1 Introduction

An understated paradigm of modern machine learning is the incorporation of inductive biases into the learning algorithm, either through the architecture or through the optimization process. For example, the use of local shift-invariant kernels in convolutional neural networks has led to one of the most important breakthroughs of computer vision in the past century, the learning of ImageNet (Krizhevsky et al., 2012; LeCun et al., 2015; Goodfellow et al., 2016).

In recent years, as computing at unprecedented scale is made possible with modern hardware, the absolute necessity of incorporating inductive bias into the model has been questioned (e.g. "transformer vs. convolution" (Bachmann et al., 2023; Brehmer et al., 2025)). It is therefore a question of great significance whether there are inductive biases that would give tremendous benefit to be incorporated as opposed to being learned with data.

In this paper, we rigorously analyze the advantages of employing models with high inductive bias for the right tasks. We focus on the learning paradigm known as "learn first, distill later" in big data settings, where an enormous multipurpose network or a foundation model is trained on an enormous dataset of the task. Later on in production, the knowledge learnt is distilled into a more efficient models for deployment (e.g. on edge devices). More formally, we argue that if the source class has latent representation learnt to perfectly perform some combinatorial optimization tasks on graphs of size n; and if the target model is *algorithmically aligned* with an algorithm that solves this optimization (e.g. dynamic programming), then the distillation from source to target can be efficient, in a rigorous model of learning theory known as PAC-distillation (Boix-Adsera, 2024).

*Please contact thien_le@seas.harvard.edu for the latest version of our paper.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: DiffCoAlg 2025.

1.1 Graph machine learning

Graph machine learning is a testbed for graph-based inductive biases that may allow for exponential gains in learning efficiency. Informally, symmetry constraints of graph functions, in terms of vertex permutations, induce certain sparsity structures in the function space, making learning more data-efficient(Bietti et al., 2021; Elesedy, 2021; Tahmasebi and Jegelka, 2023). However, learning graph neural networks and other equivariant networks is still computationally hard in the worst case, requiring, for example, exponentially or superpolynomially many queries in the correlation statistical queries model of learning (Kiani et al., 2024). Understanding which settings exactly give rise to quantitative benefits for learning is an important and active area of research.

More specifically for graphs, a graph neural network (GNN) (Gilmer et al., 2017; Kipf and Welling, 2017) is a deep-learning parameterization of the space of functions on graphs, potentially of different sizes.

1.2 Combinatorial optimization with graph ML and algorithmic alignment

One proposed area where GNNs could have strong inductive bias with the learning task is that of using neural networks to learn combinatorial optimization. It is observed (Xu et al., 2020) that the loop structure of an MPNN closely follows that of local graph algorithms, such as Bellman-Ford for shortest path. As such, Xu et al. (2020) argues that the neural network used in the aggregation operation of an MPNN only had to learn a simple function of its inputs, and not the actual for-loop structure, thereby decreasing the sample complexity of learning from supervised examples produced by such algorithms. Although the original paper provided a theoretical justification for this phenomenon through PAC learning (Valiant, 1984), a tighter analysis of what constitutes such *algorithmic alignment* has drawn many follow-up investigations (Dudzik and Veličković, 2022; Dudzik et al., 2024). Nevertheless, the idea that the learning architecture should be built to resemble a potential algorithmic paradigm, such as dynamic programming, is intuitive and has been the inspiration for many neural heuristics that are widely successful in practice (Kahng et al., 2024; Nerem et al., 2025; He and Vitercik, 2025; Gasse et al., 2019).

1.3 The linear representation hypothesis (LRH) (Elhage et al., 2022) and model distillation

The main structural assumption used in this paper is a form of linear representation hypothesis (LRH) (). One way to state it is as followed:

Definition 1 (τ-LRH)**.** *Fix a source neural network $f : \mathcal{X} \to \mathcal{Y}$ and let $\varphi : \mathcal{X} \to \mathbb{R}^m$ be the latent representation of f. Let $\mathcal{Z}$ be a set of functions $z : \mathcal{X} \to \mathcal{Y}$. For any $\tau > 0$, we say that f satisfies τ-LRH for features $\mathcal{Z}$ if for all $z \in \mathcal{Z}$, there exists $w \in \mathbb{R}^m$ such that $\|w\| \leq \tau$ and $\langle w, \varphi(x)\rangle = z(x)$ for all $x \in \mathcal{X}$.*

In the above definition, it is common to think of the source neural network f as a large foundation model, or a black-box large language model where one does not have access to the latent representation of f. The task at hand, model distillation, is to train your own smaller network given black-box access to these large networks, and a guarantee that their unknown latent representation is somehow rich enough to capture the target ground truths with just another linear layer. It turns out that this hypothesis captures a sufficient guarantee for efficient distillation. The term latent representation is an intentional abstraction for the purpose of developing theory. In practice, the penultimate layer of a deep neural network, or the collection of all neural activations of the architecture, can act as the networks' latent representation. The size of this representation, m, does show up in the complexity bounds, so conceptually, given a source network, one would choose the smallest latent representation that still satisfies τ-LRH in order to optimize the bounds.

Lots of interpretability results of deep neural embeddings can be reduced to a form of linear representation hypothesis. The most famous examples are reports of word embeddings seemingly having a 'direction' in representation space for some concepts such as gender (Bolukbasi et al., 2016) or race (Manzini et al., 2019) (e.g. the vector differences between the

pairs ('man', 'woman') and ('king', 'queen') ()). Consequently, many techniques have been developed to identify these stereotypical subspaces to de-bias the language model (Liang et al., 2020; Chuang et al., 2023).

1.4 Contribution of this paper

We are interested in the setting where the ground truth is a function computed by a local-iteration graph algorithm (Definition 3), which is a dynamic programming algorithm (DP) with states over the pair $(t, v) \in [T] \times V$ pairs of message-passing-round indices and vertices in the input graph; and the DP transition function is defined by a small decision tree of depth r. An example of a local-iteration algorithm is graph reachability. We are then asked to learn this ground truth, given black-box access to a large network or foundation model that has been trained to perfectly linearly represent a set of some simple local-iteration graph algorithms (that may not include our ground truth). In this setting,

1. Firstly, we detail a sufficient condition in terms of a linear representation hypothesis of the source network that allows for efficient distillation of certain local algorithms. Our condition, term 'local-iteration alignment' Definition 4 requires that the source network's representation can linearly represent extremely simple local-iteration algorithm of conjunctions, with a vector of coefficient of norm at most τ.
2. Secondly, we prove rigorously that, under this condition, the distillation from this large source network to the GNN is efficient (Theorem 2). Efficiency is not guaranteed when there is an algorithmic mismatch, for example, learning with decision trees (Lemma 1).
3. Finally, we give an explicit (ϵ, δ)-distillation algorithm (Boix-Adsera, 2024) for the framework (**??** 2). The algorithm draws $\mathrm{poly}(1/\epsilon, \log(1/\delta), rl)$-many samples and runs in time $\mathrm{poly}(2^{O(l^2 r)}, n, 1/\epsilon, \log(1/\delta))$ where n is the size of the graph, r is the depth of the true inner decision tree and l is the number of rounds of message passing.

This is the first, as far as the authors are aware, rigorous study of distillation into graph neural network that makes use of a form of algorithmic alignment.

2 Preliminaries

In this section, we will discuss some of the tools and materials that will be used to arrive at our results.

2.1 Notation

In general, we denote by $\mathcal{X}$ the input set and by $\mathcal{Y}$ the label set. We focus on binary classification in this paper, so, unless otherwise stated, $\mathcal{Y} = \{0, 1\}$. For a vector in S with n entries, we denote by S_i its i-th entry. We will also write $[n] := \{1, 2, \ldots, n\}$.

In congruence with languages of computational learning theory, we follow the notations of Boix-Adsera (2024). We will usually denote by $\mathcal{C}$ the concept class (class of possible ground truths), $\mathcal{H}$ the hypothesis class (output range of the learning algorithm), the input distribution $\mathcal{D}$ that induces a distribution over sample $\mathcal{D}_c \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ for some ground truth $c \in \mathcal{C}$. When there are many inputs $S \in \mathcal{X}^n$ for some $n \in \mathbb{N}$, we overload c to apply pointwise to each element of the vector $c(S) := [c(S_i)]_{i \in [n]}$. In the setting of distillation, we have a source class $\mathcal{F}$ and a target class $\mathcal{H}$.

For graph-theoretic notation, we define $\mathcal{G}_n$ to be the space of all graphs on n vertices. To make the exposition cleaner, we assume that all graphs are labeled graphs and drop the subscript n if it is clear from context. Considering a Boolean input/output graph learning model, the input is both the initial feature vector (which encodes the initialization of some DP algorithm) and also the graph structure of $\binom{n}{2}$ bits. Assuming that the dimension of each node feature is fixed and independent of n, we let $d = O(n^2)$ denote the dimension of the input of the models.

For machine learning theoretic notation, we say that a neural network defines a latent representation $\varphi : \mathcal{X} \to \mathbb{R}^m$. The exact definition of what constitutes the 'latent representation' is abstracted away. Some authors defined this as the concatenation of all neuron activations in the neural network, while others have also defined this representation as that of the penultimate layer of a deep network. The exact choice does not matter in this paper as long as they satisfy a structural assumption called a linear representation hypothesis (Definition 1).

2.2 Background

PAC learning In the traditional PAC framework (Valiant, 1984), a *concept class* (i.e., the class of possible ground truth) is (ϵ, δ)-learnable with n samples if there is an algorithm $\mathcal{A}$ such that for any distribution $\mathcal{D}$ over the input and any concept in $c \in \mathcal{C}$,

$$\Pr_{S\sim\mathcal{D}^n}[\text{error}_{c,\mathcal{D}}(\mathcal{A}(S, c(S)) \leq \epsilon] \geq 1 - \delta. \tag{1}$$

Here, the error function is the 0-1 population risk: $\text{error}_{c,\mathcal{D}}(f) := \Pr_{\mathcal{D}}[f(x) \neq c(x)]$.

PAC-distillation PAC-distillation (Boix-Adsera, 2024) is a relaxation of PAC learning in which one assumes the accessibility of a successful model class $\mathcal{F}$ to train a target class $\mathcal{H}$ by finding an algorithm $\mathcal{A}$ such that for any distribution $\mathcal{D}$ on $\mathcal{X}$, any source $f \in \mathcal{F}$,

$$\Pr_{S\sim\mathcal{D}^n}[\text{error}_{f,\mathcal{D}}(\mathcal{A}(S, f) \leq \epsilon] \geq 1 - \delta. \tag{2}$$

Such an algorithm is said to (ϵ, δ)-distill $\mathcal{F} \to \mathcal{H}$. Note that since the algorithm has access to the successful model f, giving a PAC distillation algorithm is easier than giving a PAC algorithm since one can just use f to query labels and simulate PAC. The advantage of this framework is 1) to sidestep some of the hardness results of PAC learning with relaxation and 2) to make use of extra natural structures in the class $\mathcal{F}$, such as some forms of linear representation hypothesis that is widely observed in practice (Elhage et al., 2022; Bolukbasi et al., 2016).

In practice, $\mathcal{F}$ can be thought of as pre-trained complex neural networks that have achieved low errors on some tasks, and the target class $\mathcal{H}$ can be understood as function classes with inductive bias that can more efficiently represent the ground truth, for example, invariant neural networks such as convolutional neural nets (CNNs) or graph neural nets (GNNs). Distillation then asks if there are efficient algorithms to find a good representation of the ground truth in the target class.

The design of $\mathcal{F}$ and $\mathcal{H}$ should be taken with great care so that the distillation of PAC is not trivial. For example, if $\mathcal{F} \subseteq \mathcal{H}$ then the learning algorithm can just return the second argument. Or, if $\mathcal{H}$ admits efficient approximations of functions in $\mathcal{F}$, then returning the approximation also solves the problem.

To give a taste of the results that can be obtained from this framework, we restate a distillation theorem. Recall that in the Boolean setting, a decision tree has vertices labeled by some literals of the input bits and each vertex is only reachable by inputs that satisfy the conjunction of literals on the path from the root to said vertex.

Theorem 1 (Theorem 3.6 of (Boix-Adsera, 2024))**.** *Let $\mathcal{F}$ be the set of neural networks f that implicitly compute a decision tree $T : \{0,1\}^d \to \{0,1\}$ of depth r and size s such that f satisfies τ-LRH for features $\mathcal{Z}_T := \{\bigwedge_{p\in S} p : S$ is a path from the root of T any of its vertices$\}$. Let $\mathcal{H}$ be the set of decision trees with depth r and size s. Then for any $\epsilon, \delta \in (0,1)$, there is an algorithm that (ϵ, δ)-distills from $\mathcal{F}$ to $\mathcal{H}$ that runs in polynomial time in $d, m, 1/\epsilon, s, 2^r, \log(1/\delta), \tau$ and B and takes polynomially many samples in $1/\epsilon, s, \log(d/\delta), \log(\tau B)$ where $B \geq \max_x \|\varphi(x)\|$.*

This is a remarkable result, since learning a decision tree in the PAC framework is conjectured to take $d^{\Omega(r)}$ time (Bary-Weisberg et al., 2020), but PAC distillation takes only $\text{poly}(d, 2^r)$ time.

Algorithmic alignment In this paper, we propose that a trained neural network that satisfies τ -LRH for some features Z is an example of algorithmic alignment. More general alignments, which will be defined later, can also lead to efficient distillation bounds.

Graph neural networks The main architecture we consider as the target class is that of graph neural networks with strong inductive bias for graph datasets. In particular, we are interested in message passing neural network (MPNN), in which each node aggregates neighboring information and processes them with a neural network to form a new latent representation in each round. After a fixed number of rounds, the network outputs a learnt representation for each vertex of the graph, or combines them together to form a single representation for the whole graph, depending on the specific tasks.

3 Main results

All graphs discussed in this section are labeled and assumed to have n vertices for some $n \in \mathbb{N}$.

We first give some definitions specific to our settings:

Definition 2 (Neural networks that computes graph algorithms)**.** *A neural network $\nu : \mathcal{G} \times \mathcal{X} \to \mathcal{Y}$* computes graph algorithm *$\mathcal{A}$ if ν agrees with A on all inputs of size n. It is* efficient *if it can be evaluated in polynomial time in n.*

Definition 3 (Local-iteration algorithm)**.** *For any function $f : \{0,1\}^n \times \mathcal{G} \times [n] \to \{0,1\}$ let $\mathcal{A}^l[f] : \{0,1\}^n \times \mathcal{G} \to \{0,1\}$ be a graph input algorithm that computes:*

Algorithm 1: Local-iteration algorithm $\mathcal{A}^l[f]$

Input: Initialization vector INIT, Graph $G = (V, E)$
Output: $\{0,1\}$ classification

$h_{v,0} \leftarrow$ Initialize for each $v \in V$ with INIT
for $t \in [l]$ **do**
  for $v \in V$ **do**
    $h_{v,t} \leftarrow f(\{\{h_{u,t-1}\}\}_{u \in N(v)}, G, v)$
return $h_{n,l}$.

In the remainder of the paper, we will assume that we are given the stopping time l *a priori*. For k-local algorithms, $l = k$, while for more general algorithms, l can depend on n. Unless otherwise stated, we focus on the former case.

We consider the following intuitive form of algorithmic alignment that was proposed in the seminal paper of Xu et al. (2020):

Definition 4 (Local-iteration alignment)**.** *Fix a source neural network $\nu \in \mathcal{F}$ and let $\varphi : \mathcal{X} \to \mathbb{R}^m$ be the latent representation of ν. Let Z be a set of functions $z : \{0,1\}^n \times \mathcal{G} \to \{0,1\}$. For any $\tau > 0$, we say that ν satisfies τ-local-iteration alignment for features Z if for all $z \in Z$, there exists a $w \in \mathbb{R}^m$ with $\|w\| \leq \tau$ and $\langle w, \varphi \rangle = \mathcal{A}^l[z]$.*

Finally, we define a decision tree:

Definition 5. *A decision tree $T : \{0,1\}^d \to \{0,1\}$ is a labeled rooted binary tree with leaf labeled 0 or 1 and internal node labeled by its input variables $x_1 \dots x_d$. Each input takes a path by evaluating the input variable to arrive at the leaf that is the output of the tree.*

3.1 Concept class, source class and target class

Our concept class (the collection of possible ground truths) will be:

$$\mathcal{C}_{s,r} = \{\mathcal{A}[T] \mid \ T \text{ is a decision tree with depth } s \text{ size } r\} \tag{3}$$

To define the source class, we first need to give the set of features that is supposedly linearly represented by our source functions. This is done analogously to (Boix-Adsera, 2024): given a decision tree T, the root-prefix path conjunctions form T's features:

$$Z'_T := \left\{ \bigwedge_{i=1}^{l} p_i \mid p_i \text{ is a literal } (x_j \text{ or } \neg x_j \text{ for some } j) \text{ s.t. } (p_1, \dots, p_l) \text{ is a path from root} \right\} \tag{4}$$

Given these features for decision trees, the features for local-iteration algorithms are:

$$Z_T := \left\{\mathcal{A}^l[b] \mid b \in Z'_T\right\} \tag{5}$$

We postulate that such loops over prefix path conjunctions are simply representable by the neural network's latent representation.

The source class (the collection of neural networks that have successfully learnt some graph algorithms) will be:

$$\mathcal{F}^{\tau}_{s,r} = \{\text{neural networks implicitly computing } \mathcal{A}^l[T] \text{ for some decision tree } T \\ \text{that also satisfied } \tau\text{-local-iteration alignment for features } Z'_T\}$$

And finally, the target class is the same as the concept class. In practice, this is a subset of graph neural networks, and the distillation process can be understood as distilling from learnt neural networks to graph neural networks.

3.2 GNNs are more efficient than decision trees

In the following, we state a simple fact that makes for a good exercise:

Lemma 1. *There exists a simple decision tree $T : \{0,1\}^n \times \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ that can be evaluated in polynomial time in n such that while $\mathcal{A}[T]$ can be evaluated in polyonomial time, it cannot be represented by decision trees of polynomial size.*

Proof sketch. Consider the 2-reachability DP. In other words, given the adjacency matrix of a graph on $n \geq 2$ vertices, is there a path of length at most 2 that connects the vertex labeled 1 and n? The full proof is in the Appendix. □

Intuitively, this separation comes from the power of memorization utilized by dynamic programming algorithms where subproblems are solved and their solutions stored and reused many times in a 'DP table'. This fact also means that without the for loop structure, one cannot just convert the concept class $\mathcal{C}_{s,r}$ into the class of efficient decision trees. This forms a type of algorithmic mismatch which would be resolved in the next part, using graph neural networks.

3.3 GNNs can be distilled from learnt and aligned neural networks efficiently

We are now ready to state the main theorem.

Theorem 2. *a For any $\epsilon, \delta \in (0,1)$, there is an algorithm that (ϵ,δ)-distills from $\mathcal{F}^{\tau}_{s,r}$ to $\mathcal{C}_{s,r}$ and runs in polynomial time in $n, m, 1/\epsilon, s^n, 2^{nrl}, \log(1/\delta), \tau, B$ and takes a polynomial sample in $1/\epsilon, s, \log(d/\delta), \log(\tau B)$ from $\mathcal{D}$ where $B = \max_x \|\varphi(x)\|$.*

The runtime is polynomial, assuming that the (maximun) number of vertices n and the number of GNN iteration l are fix constants, and that the depth of the true decision trees are of logarithmic order. The proof uses Algorithm 2, which first builds a set of conjunctions that is a superset of all root-prefix conjunctions in the true tree and then stitches these conjunctions together efficiently using a well-known tree building DP that is classical in the literature of PAC-learning decision trees (Mehta and Raghavan, 2002).

In our algorithm, two key subroutines are used:

1. The LinearProbe$(g, \varphi, B, \tau, \epsilon, \delta, \mathcal{D})$ subroutine comes from Lemma 3.7 of (Boix-Adsera, 2024) that runs in polynomial time and draws polynomially many samples to return true w.p. $\geq 1-\delta$ if there is a $w \in \mathbb{R}^m$ with $\|w\| \leq \tau$ and $\mathbb{E}_x[(\langle w, \varphi\rangle - g)^2] \leq 2$ and return false w.p. $\geq 1-\delta$ if for all such w, the expectation is at least 2ϵ.
2. The maximization over decision trees in the final step is done with a DP (Guijarro et al., 1999; Mehta and Raghavan, 2002) and the function val is chosen such that maximizing it corresponds to minimizing the 0-1 risk.

Algorithm 2: GNN distillation algorithm

Input: Neural network ν, random samples from $\mathcal{D}$, depth bound $R \in \mathbb{N}$, error parameters $\epsilon, \delta > 0$.

Output: GNN that computes $\mathcal{A}^l[\widehat{T}]$

$\mathcal{S}_0 \leftarrow \{\emptyset\}$ **for** $i = 1 \in [R]$ **do**

  r

$\mathcal{P}_{i-1} \leftarrow \left\{ S \in \mathcal{S}_{i-1} \text{ s.t } \textsc{LinearProbe}(\mathcal{A}^l[\bigwedge_{p_i \in S} p_i], \varphi, B, \tau, 2^{-i^l-3}, \frac{\delta}{2|\mathcal{S}_{i-1}|R}) = \text{true} \right\}$

$\mathcal{S}_i \leftarrow \bigcup_{S \in \mathcal{P}_{i1}} \bigcup_{j=1}^{d} \{S \cup x_l, S \cup \neg x_l\}$ $\mathcal{S} \leftarrow \bigcup_{j=1}^{R} \mathcal{S}_j$

$\hat{v}_{S'} \leftarrow \mathbb{E}_x[\bigwedge_{p_i \in S'_i, i \in [n]} p_i(x)(2\nu(x) - 1)] \pm \epsilon/|\mathcal{S}|^n$, for each $(S'_1, S'_2, \ldots, S'_n) \in \mathcal{S}^n$

return $\text{argmax}_T \text{val}(T_1, \ldots, T_n, \hat{v})$ where T_i is over decision trees with $Z'_T \subseteq \{\bigwedge_{p_i \in S} p_i \mid S \in \mathcal{S}\}$

The idea of this algorithm is for $\mathcal{S}$ to gather a collection of clauses that is a superset of Z'_T–the set of all root-prefix path in the true tree, with high probability. This is done by starting with the empty clause and recursively grow them by one valid literal at a time. In this growing process, if we meet a clause S whose conjunction $\bigwedge_{p \in S} p$ cannot be linearly represented by the source network with a low norm coefficient vector then we prune this clause. LinearProbe subroutine gives us the ability to decide whether to prune a clause or keep it in $\mathcal{S}$. The key technical challenge is to ensure that even with all these pruning, the size of the $\mathcal{S}$ is not too large and that $\mathcal{S}$ does contain every clauses in the true tree with high probability.

In the second step, we set up a dynamic programming algorithm as per Mehta and Raghavan (2002) that builds up our estimated inner decision tree. This step is more involved than when we only have one global decision tree as in the setting of (Boix-Adsera, 2024), because the same input to $\mathcal{A}^l[\hat{T}]$ can traverse the tree $\hat{T}$ in more than one paths. This is because although the description of $\hat{T}$ is the same throughout the loops, its inputs are different based on the state of the local-iteration DP. To resolve this issue, we work under the formulation that each vertex has its own decision tree, which can then be appended under a $\log n$-depth index selector. Then we build all n vertex-dependent trees simultaneously.

We can now describe the DP that estimates our tree. The states of the DP are indexed by tuples $(S'_1, \ldots, S'_n, s'_1, \ldots, s'_n) \in \mathcal{S}^n \times [s]^n$ where s is the size (upper bound) of the true decision tree given when we define the ground truth function class. The DP transition is computed by, at clause S'_i and size s'_i, we compute the optimal subtree of size s'_i rooted at the end of path S'_i for all $i \in [n]$. Optimality is defined with respect to an evaluation function val that estimates the 0-1 loss of the candidate tree (to be rigorously defined in the appendix). The tree that optimizes 0-1 loss can then be queried at the final DP state.

The full correctness proof Theorem 2 is defered to the Appendix.

4 Conclusion and discussion

In this paper, we extend the work of Boix-Adsera (2024) in PAC-distillation to characterize learning models that have built-in algorithmic alignment properties, such as GNNs for dynamic programming. We showed that while there are some DP algorithms whose inner function is a small decision tree, the whole DP itself cannot be represented by an efficient decision tree – a case of misalignment. On the other hand, the local iteration structure of a GNN with decision tree aggregation allows for efficient distillation from a large, learnt neural network that exhibits a certain kind of linear representability.

Although this marks the first work in this direction, there is much room for improvement with this version of the paper. For brevity of exposition, we adapt the analysis of (Boix-Adsera, 2024) as is and naively, under the assumption that the outer loop of the DP has a constant number of iterations and that the size of the innermost decision tree is small. A more complicated and nuanced analysis greatly reduces these requirements.

Finally, the purpose of this paper is to introduce the distillation pipeline, which has shown great promise in rigorously studying algorithmic alignment. Besides a more detailed analysis, future directions include the statistical learning theory questions of sample complexity under distillation or the study of pretrained large language models, whose finetuning might be thought of as a form of distillation.

Acknowledgments and Disclosure of Funding

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) under agreement no. HR0011-25-3-0205. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Scaling MLPs: A tale of inductive bias. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=R45A8eKcax.

Galit Bary-Weisberg, Amit Daniely, and Shai Shalev-Shwartz. Distribution free learning with local queries. In Aryeh Kontorovich and Gergely Neu, editors, *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pages 133–147. PMLR, 08 Feb–11 Feb 2020. URL https://proceedings.mlr.press/v117/bary-weisberg20a.html.

Alberto Bietti, Luca Venturi, and Joan Bruna. On the sample complexity of learning under invariance and geometric stability. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Enric Boix-Adsera. Towards a theory of model distillation, 2024. URL https://arxiv.org/abs/2403.09053.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4356–4364, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Johann Brehmer, Sönke Behrends, Pim De Haan, and Taco Cohen. Does equivariance matter at scale? *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=wilNute8Tn.

Ching-Yao Chuang, Varun Jampani, Yuanzhen Li, Antonio Torralba, and Stefanie Jegelka. Debiasing vision-language models via biased prompts, 2023. URL https://arxiv.org/abs/2302.00070.

Andrew J Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20635–20647. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8248b1ded388fcdbbd121bcdfea3068c-Paper-Conference.pdf.

Andrew Joseph Dudzik, Tamara von Glehn, Razvan Pascanu, and Petar Veličković. Asynchronous algorithmic alignment with cocycles. In Soledad Villar and Benjamin Chamberlain, editors, *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pages 3:1–3:17. PMLR, 27–30 Nov 2024. URL https://proceedings.mlr.press/v231/dudzik24a.html.

Bryn Elesedy. Provably strict generalisation benefit for invariance in kernel methods. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. *Exact combinatorial optimization with graph convolutional neural networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning (book). In *MIT Press*, 2016.

David Guijarro, Víctor Lavín, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Inf. Process. Lett.*, 70(5):233–239, June 1999. ISSN 0020-0190. doi: 10.1016/S0020-0190(99)00063-0. URL https://doi.org/10.1016/S0020-0190(99)00063-0.

Yu He and Ellen Vitercik. Primal-dual neural algorithmic reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=iBpkzB5LEr.

Andrew B. Kahng, Robert R. Nerem, Yusu Wang, and Chien-Yi Yang. Nn-steiner: a mixed neural-algorithmic approach for the rectilinear steiner minimum tree problem. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i12.29200. URL https://doi.org/10.1609/aaai.v38i12.29200.

Bobak Kiani, Thien Le, Hannah Lawrence, Stefanie Jegelka, and Melanie Weber. On the hardness of learning under symmetries. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ARPrtuzAnQ.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25. Curran Associates, Inc., 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Paul Pu Liang, Irene Mengze Li, Emily Zheng, Yao Chong Lim, Ruslan Salakhutdinov, and Louis-Philippe Morency. Towards debiasing sentence representations. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5502–5515, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.488. URL https://aclanthology.org/2020.acl-main.488/.

Thomas Manzini, Lim Yao Chong, Alan W Black, and Yulia Tsvetkov. Black is to criminal as Caucasian is to police: Detecting and removing multiclass bias in word embeddings. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 615–621, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1062. URL https://aclanthology.org/N19-1062/.

Dinesh P. Mehta and Vijay Raghavan. Decision tree approximations of boolean functions. *Theor. Comput. Sci.*, 270(1-2):609–623, 2002. URL https://doi.org/10.1016/S0304-3975(01)00011-1.

Robert R. Nerem, Samantha Chen, Sanjoy Dasgupta, and Yusu Wang. Graph neural networks extrapolate out-of-distribution for shortest paths, 2025. URL https://arxiv.org/abs/2503.19173.

Behrooz Tahmasebi and Stefanie Jegelka. The exact sample complexity gain from invariances for kernel regression. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=6iouUxI45W.

L. G. Valiant. A theory of the learnable. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, page 436–445, New York, NY, USA, 1984. Association for Computing Machinery. ISBN 0897911334. doi: 10.1145/800057.808710. URL https://doi.org/10.1145/800057.808710.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJxbJeHFPS.

A Proof of Theorem 2

A.1 Notations and definitions

We will set up some notation for this particular proof and also remind the readers of previously defined notation:

- **The input space:** Since we are using a graph neural network to emulate an algorithm of the form $\mathcal{A}^l[T]$ for some decision tree T, there is a difference between the input of $\mathcal{A}^l[T]$ and that of T. The former takes as input an initialization feature in $\{0,1\}^n$ a graph adjacency matrix $\{0,1\}^{n\times n}$ and we write $\mathcal{X}_{\mathcal{A}}$ for this input space. T itself has an input consisting of the previous representation of each layer $\{0,1\}^n$, a graph adjacency matrix $\{0,1\}^{n\times n}$ and additionally the index of a vertex v and we reserve $\mathcal{X} := \{0,1\}^d$ where $d = O(n^2)$.
- **Logical notation:** for some input bit x_j, $j \in [d]$, a *literal* is x_j or its negation $\neg x_j$. A *clause* $S = (p_1, \ldots, p_s)$ is an ordered tuple of s literals and we define $\text{AND}_S(x) := \bigwedge_{p\in S} p$ the conjunction of literals in S. A *non-degenerate* k-clause is a clause S such that $|S| = k$ and each variable appears at most once in S.
- **Decision trees:** Given a decision tree T, recall that Z'_T is the set of clauses each corresponds to a path in T with one end-point being the root (i.e. a *root-prefix path*). We include the trivial path $\emptyset$ with $\text{AND}_\emptyset = \text{TRUE}$ in this collection. We also denote by Z_T the collection of all $\mathcal{A}^l[\text{AND}_S]$ functions for each $S \in Z'_T$.

In our algorithm, which is an extension of that in (Boix-Adsera, 2024), we use some subroutines from the original paper.

Lemma 2 (Lemma 3.7 (Boix-Adsera, 2024))**.** *Given a function $g : \mathcal{X} \to [-1,1]$, a representation map $\varphi : \mathcal{X} \to \mathbb{R}^m$ with norm bounded by $B \geq \max_x \|\varphi(x)\|$, $\tau, \epsilon, \delta > 0$ and an input distribution $\mathcal{D}$, there is a subroutine $\textsc{LinearProbe}(g, \varphi, B, \tau, \epsilon, \delta, \mathcal{D})$ that runs in time $\text{poly}(1/\epsilon, \log(1/\delta), \tau, B, m)$ and draws $\text{poly}(1/\epsilon, \log(1/\delta), \tau, B)$ samples from $\mathcal{D}$ such that:*

- *If there is a $w \in \mathbb{R}^m$ with $\|w\| \leq \tau$ and $\mathbb{E}[(w\cdot\varphi(x) - g(x))^2] \leq \epsilon$, then $\textsc{LinearProbe}$ returns true with probability $1-\delta$.*
- *If there is no $w \in \mathbb{R}^m$ with $\|w\| \leq \tau$ and $\mathbb{E}[(w\cdot\varphi(x) - g(x))^2] \leq \epsilon$, then $\textsc{LinearProbe}$ returns false with probability $1-\delta$.*

A.2 Proof of Theorem 2

We are now ready to start the proof. Assume that there is a true decision tree T. We first show that the paths collection from the distillation algorithm contains all root prefix paths in the true tree: $\mathcal{S} \supseteq Z_T$ with high probability.

$\mathcal{S}$ contains all root-prefix paths of the true tree From the guarantees of Lemma 2, it suffices to show that any clause $S \in Z'_T$ is checked by $\textsc{LinearProbe}$ and thus added to $\mathcal{S}$ with high probability. Assume to the contrary that this is not true; in other words, there is a root-prefix path $S \in Z'_T$ of length i that was not added to $\mathcal{S}_i$. Recall that whenever $\textsc{LinearProbe}$ accepted a clause S', we added all possible extension of S' to our collection $\mathcal{S}$. The fact that S was not included means that either it was not checked by $\textsc{LinearProbe}$ or it was checked and then rejected. In the former case, this means that $S \neq \emptyset$ (since the $\emptyset$ is always checked) and the root-prefix path corresponds to S' parent was not included in the previous set $\mathcal{S}_{i-1}$. We can then use induction on this parent node instead. In the latter case, S was checked but $\textsc{LinearProbe}$ returns false, which occurs with probability $1 - \frac{\delta}{2|\mathcal{S}_i|R}$ because of our linear representation hypothesis and Lemma 2.

A union bound at each layer suffices to argue that all length i clauses in Z'_T are in $\mathcal{S}$ with probability $1 - \frac{\delta}{2R}$ for each i. Another union bound over all i then concludes that $Z'_T \in \mathcal{S}$ with probability $1 - \delta/2$.

Now we argue that $|S_i| \leq \text{poly}(2^{\Theta(il)}, \tau, B, d)$

$\mathcal{S}_i$ size is upper-bounded The key way we control $\mathcal{S}_i$ size is by arguing that only the $\mathcal{A}^l[\mathrm{AND}_S]$'s that can be linearly represented by the source network are kept while the rest are pruned. Furthermore, there cannot be too many of these $\mathcal{A}^l[\mathrm{AND}_S]$ kept, at the same time. A naïve approach uses the following lemma from (Boix-Adsera, 2024).

Lemma 3 (Lemma 3.8 (Boix-Adsera, 2024))**.** *Let $\mathcal{S}$ be a collection of non-degenerate k-clauses. Let $\mathcal{G} = \{AND_S \mid S \in \mathcal{S}\}$. If φ approximately satisfies τ-LRH w.r.t. $\mathcal{G}$:*

$$\forall g \in \mathcal{G}, \exists w \in \mathbb{R}^m, \|w\| \leq \tau \text{ and } \mathbb{E}_{x\sim U[\{0,1\}^d]}[(w \cdot \varphi(x) - g(x))^2] \leq 2^{-k-2}, \tag{6}$$

then $|\mathcal{S}| \leq 2^{3k+4}\tau^2\mathbb{E}_x\|\varphi(x)\|^2$

We want to strengthen this to the following:

Lemma 4 (Packing with for-loops)**.** *Let $\mathcal{S}$ be a collection of non-degenerate k-clauses. If φ approximately satisfies local iteration alignment w.r.t. $\mathcal{S}$:*

$$\forall S \in \mathcal{S}, \exists w \in \mathbb{R}^m, \|w\| \leq \tau \text{ and } \mathbb{E}_{x\sim U[\{0,1\}^d]}[(w \cdot \varphi(x) - \mathcal{A}^l[[AND_S]](x))^2] \leq 2^{-\Theta(kl)}, \tag{7}$$

then $|\mathcal{S}| \leq 2^{\Theta(kl)}\tau^2\mathbb{E}_x\|\varphi(x)\|^2$.

Proof. To demonstrate the structure of $\mathcal{A}^l[\mathrm{AND}_S]$, we will perform a loop unrolling. Recall that the input to the inner tree has three parts: some bits to specify the vertex, which we will denote $x_{v,1}\dots x_{v,\log n}$; some bits to query the graph adjacency matrix $x_{e,1}\dots x_{e,\binom{n}{2}}$ and some bits to query the DP table $x_{dp,1}\dots x_{dp,n}$. Fix a k-clause S. Let its index set be $\{v_1,\dots,v_a;e_1,\dots,e_b;dp_1,\dots,dp_c\}$ where $a+b+c=k$ and denote by z_I the literal for x_I for some I in the index set. We have, for some initialization vector and graph adjacency A:

$$\begin{aligned}\mathcal{A}^l[\mathrm{AND}_S](\textsc{Init}, A) &:= h_{n,l}(\textsc{Init}, A) &(8)\\ &= \bigwedge_{i\in[a]} z_{v_i}(n) \wedge \bigwedge_{j\in[b]} z_{e_j}(A) \wedge \bigwedge_{u\in[c]} z_{dp_u,l-1}(\textsc{Init}, A) &(9)\end{aligned}$$

Now, the conjunctions $\bigwedge_{i\in[a]} z_{v_i}$ defines a bipartition of the vertex set into two parts,. Denote by $c_{S,v} \in \{0,1\}$ the indicator function for the set carved out by $\bigwedge_{i\in[a]} z_{v_i}$ and remark that we know $c_{S,v}$ even before seeing any input (and thus they are constants w.r.t. $\mathcal{A}^l[\mathrm{AND}_S]$). Furthermore, we write $\bigwedge_{j\in[b]} z_{e_j}(A)$ as $S(A)$ since this quantity depends only on the input graph. Thus:

$$\begin{aligned}\mathcal{A}^l[\mathrm{AND}_S](\textsc{Init}, A) &= c_{S,n} \wedge S(A) \bigwedge_{u\in[c]} z_{dp_u,l-1}(\textsc{Init}, A) &(10)\\ &= c_{S,n} \wedge S(A) \wedge \bigwedge_{u\in[c]^+} c_{S,dp_u} \wedge S(A) \wedge \bigwedge_{u_2\in[c]} h_{dp_{u_2},l-2}(\textsc{Init}, A)\\ &\wedge \bigwedge_{u\in[c]^-} \neg\left(c_{S,dp_u} \wedge S(A) \wedge \bigwedge_{u_2\in[c]} h_{dp_{u_2},l-2}(\textsc{Init}, A)\right) &(11)\end{aligned}$$

At layer i of the for-loop, evaluating an entry asks for c evaluation of the previous layer, naïvely giving us c^l evaluations when completely unrolling all l layers of $\mathcal{A}^l$. However, because the evaluations are only at the indices $dp_1,\dots,dp_c$ which are determined by S, we do not need to fill out the whole DP table but only at these c points. Thus, $\mathcal{A}^l[\mathrm{AND}_S](\textsc{Init}, A)$ depends on $\textsc{Init}$ only through the c bits and on A only through $S(A)$ which looks at the b bits of A. Thus, $\mathcal{A}^l[\mathrm{AND}_S]$ is a function of $(b+c)$ bits ($\leq k$ bits) of its input, i.e., a $(b+c)$-junta.

Having established that $\mathcal{A}^l[\mathrm{AND}_S]$ are functions of at most k bits, we can use the same packing bound based on Fourier-analytic arguments of (Boix-Adsera, 2024).

Recall that τ-local-iteration alignment implies that: for every $S \in Z'_T$, there is a $w_S \in \mathbb{R}^m$ with $\|w_S\| \leq \tau$ such that, $\langle w, \varphi(x)\rangle = \mathcal{A}^l[\mathrm{AND}_S(x)]$ for all $x \in \{-1,1\}^d$ (note that here

we use the domain $\{\pm 1\}^d$ that works better with Fourier analytic arguments of boolean functions). Collect all such w into the rows of a matrix $W \in \mathbb{R}^{|\mathcal{S}| \times m}$ and $\varphi(x)$ into the columns of some $\Phi \in \mathbb{R}^{m \times 2^d}$. We can derive the packing bound from the fact that orthogonal projection to the row span of $V \in \mathbb{R}^{\binom{d}{k} \times 2^d}, V_{A,x} = \chi_A(x)$ for all $A \in \binom{[d]}{k}$ and χ_A being the parity function of indices in A, has large norm. Let $P \in \mathbb{R}^{2^d \times 2^d}$ be the orthogonal projection to this subspace of low degree polynomials in $L^2(\{\pm 1\}^d)$ and $P^\top$ the projection to the orthogonal subspace

First, we want to compute the coefficient of the degree k term of $\mathcal{A}^l[\mathrm{AND}_S](\textsc{Init}, A)$ when written as $\{\pm 1\}^d$-polynomial gives:

$$\mathcal{A}^l[\mathrm{AND}_S](\textsc{Init}, A) = h_{n,l} \tag{12}$$

$$= 2^{-(c+2)+1}(1 + c_{S,n}) \cdot (1 + S(A)) \cdot \prod_{u \in [c]} (1 + \sigma h_{dp_u, l-1}(\textsc{Init}, A))) - 1 \tag{13}$$

where σ is either 1 or -1 depending on the sign of the literal. Thus the leading term is of degree k has coefficient $\pm 2^{-\Theta(kl)}$. Therefore,

$$[W \Phi P_k]_{S,x} = \pm 2^{-\Theta(kl)} \chi_{I(S)}, \tag{14}$$

where $I(S)$ is the set of indices of the input that appears in literals of S.

Therefore,

$$W \Phi \Phi^\top W^\top \succeq W \Phi P P^\top \Phi^\top W^\top = 2^{d - \Theta(kl)} I, \tag{15}$$

and one conclude that $|\textsc{det}(W \Phi \Phi^\top W^\top)| \geq 2^{(d - \Theta(kl))|\mathcal{S}|}$.

Using τ-local-iteration alignment, we get the following for free:

Lemma 5 (Claim B.4 (Boix-Adsera, 2024)). *We have:* $\textsc{det}(W \Phi \Phi^\top W^\top) \leq (2^d (\mathbb{E}\|\varphi\|^2)^2 \tau^2 / |\mathcal{S}|)^{|\mathcal{S}|}$.

□

Combining the two gives:

$$2^{(d - \Theta(kl))|\mathcal{S}|} \leq (2^d (\mathbb{E}\|\varphi\|^2)^2 \tau^2 / |\mathcal{S}|)^{|\mathcal{S}|}, \tag{16}$$

which gives $|\mathcal{S}| \leq 2^{\Theta(kl)} (\mathbb{E}\|\varphi\|^2)^2 \tau^2$.

Finally we gives details on the DP procedures that stitch together all the root-prefix paths to find the true tree based on 0-1 loss surrogate.

Dynamic programming algorithm to infer final tree For this section, recall that the concept class dictates that the decision tree can be different when processing different vertices of the graph. To this end, we let t_i be the true decision tree for vertex i and treat them as different decision tree. To get back the full tree, we simply add a $\log n$-depth index selector at the beginning of the estimated tree.

The problem is reduced to inferring t_i's simultaneously. Recall that the set of possible clauses for tree i is $\mathcal{S}_i$. For any weight function $u : \bigotimes_i \mathcal{S}_i \to \mathbb{R}$, define the valuation function:

$$\mathrm{val}(\widetilde{T}_1, \ldots, \widetilde{T}_n, u) := \sum_{S_i \in \mathrm{Leaves}(T_i), i \in [n]} (2\widetilde{T}(S_1, \ldots, S_n) - 1) u_{S_1, \ldots, S_n}, \tag{17}$$

where $\widetilde{T}(S_1, \ldots, S_n)$ is defined as the computing the template function $\mathcal{A}[f]$ replacing $\widetilde{T}_i$ with S_i at only the first layer.

Recall that the weight function for our dynamic program is:

$$v : \bigotimes_i \mathcal{S}_i \to \mathbb{R}, v_{S_1, \ldots, S_n} := \mathbb{E}_{x \sim \mathcal{D}} \left[(2 f_\theta(x) - 1) \prod_i \mathrm{AND}(S_i) \right]. \tag{18}$$

Using this weight function, one gets back the 0-1 loss:

$$\mathrm{val}(\widetilde{T}_1,\ldots,\widetilde{T}_n,v) = \sum_{S_i\in\mathrm{Leaves}(T_i),i\in[n]} (2\widetilde{T}(S_1,\ldots,S_n)-1)\mathbb{E}_{x\sim\mathcal{D}}\left[(2f_\theta(x)-1)\prod_i \mathrm{AND}(S_i)\right] \tag{19}$$

$$= \mathbb{E}_{x\sim\mathcal{D}}\left[(2f_\theta(x)-1)\left(\sum_{S_i\in\mathrm{Leaves}(T_i),i\in[n]} (2\widetilde{T}_i(S_i)-1)\prod_i \mathrm{AND}(S_i)\right)\right]. \tag{20}$$

Since for each input x in the domain, by the definition of a decision tree, exactly one path S_i is traversed for each tree at node i. For these correct path, $\widetilde{T}(S_1,\ldots,S_n)=\widetilde{T}(x)$. Thus we have:

$$\mathrm{val}(\widetilde{T}_1,\ldots,\widetilde{T}_n,v) = \mathbb{E}_{x\sim\mathcal{D}}\left[(2f_\theta(x)-1)(2\widetilde{T}(x)-1)\right], \tag{21}$$

which is the $0-1$ loss for the estimated trees.

In our algorithm, we use Hoeffding inequality to approximate v with random sampling. Note that this step requires, in the worst case, with probability at least $1-\delta/2$, approximating $|\mathcal{S}|^n$ entries of v naively with error at most $\epsilon/\prod_i s_i$ where s_i is a bound on the number of leaves of t_i (naively $|\mathcal{S}|$), and runs in time $\mathrm{poly}(m',n)$ where $m'=\mathrm{poly}(1/\epsilon,\log(|S|^n/\delta))$ is the number of draws to obtain the Hoeffding bound. When choosing $R=r$ in the algorithm, $|\mathcal{S}|$ is of order $2^{O(lr)}(\mathbb{E}\|\varphi\|)^2\tau^2$ based on the previous bound on $|\mathcal{S}|$ and the approximation of v is done in $\mathrm{poly}(n,l,r,1/\epsilon,\log(1/\delta))$.

Finally, we can run the dynamic program that computes for each $S_1\in\mathcal{S}_1$, each tree size $s_1'=0..s_1$, for each $S_2\in\mathcal{S}_2$, each tree size $s_2'=0..s_2$, etc. the best subtrees $\widetilde{T}_i$ of size s_i' rooted at the end of the clause S_i, for all $i\in[n]$. The runtime of this DP is computed as $|\mathcal{S}|^n\cdot s^n\cdot\mathrm{poly}(n,l,r,1/\epsilon,\log(1/\delta)) = \mathrm{poly}(2^{nlr},(\mathbb{E}\|\varphi\|)^2\tau^2,s^n,n,l,r,1/\epsilon,\log(1/\delta))$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper is about proving a distillation results for graph learning architecture under some algorithmic alignment assumption.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussion part of the paper highlights limitations and future directions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Sketch proofs are provided with enough details so as to not repeat existing works extensively.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer:[NA]

Justification: There are no experiments in this paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: There are no data or code in this paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [NA]

   Justification: There are no experiments in this paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: There are no experiments in this paper.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: There are no experiments in this paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: The paper is a mathematical and theoretical study in the theory of computation and does not carry extra societal impacts that are worth highlighting.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No such risk

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All original papers are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No such risks

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLM are used only for editing

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.