LINEAR-TIME GENERATIVE INFERENCE IN NEURAL PROBABILISTIC CIRCUITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Probabilistic Circuits (NPCs) are neuro-symbolic models that combine both the recognition capacity of modern neural networks and the tractable reasoning of probabilistic circuits. In NPCs, a neural network predicts human-interpretable attributes from the input, and a follow-up probabilistic circuit models the joint distribution of those attributes and the class label. To connect the two modules, the existing inference procedure in NPCs averages the circuit's class conditional probabilities over all attribute configurations, both during training and at test time. In the worst case, the average operation costs time exponential in the number of attributes, which limits the scalability of NPCs to applications with at most tens of attributes. To overcome the above limitation, we propose a new inference procedure, which we term as the *generative inference*, that averages each class by the joint probability, rather than the class conditional probability, over the network's attribute predictions. We show that under factorized attribute predictions from the neural network module, each smooth and decomposable circuit can instead compute the overall prediction exactly in one forward pass of the circuit, which reduces the inference time complexity from exponential in the number of attributes to linear in the circuit size. Our key insight that enables the exponential reduction lies in the observation that factorized functions over the leaf nodes in a smooth and decomposable circuit can be absorbed into the leaf factors, hence effectively exploiting the tractability of circuits. Empirically, the generative inference helps lift NPCs to datasets with hundreds of attributes, which the original discriminative inference cannot scale to. In terms of statistical performance, by reusing the generative inference during learning, we demonstrate that the learned NPCs are at least comparable to, and sometimes better than the original NPCs.

1 INTRODUCTION

Deep Neural Network based classifiers reach high accuracy but their decisions aren't explainable. Neuro-symbolic models insert an explicit symbolic layer between perception and prediction (Manhaeve et al., 2018), so the decision is based on physical attributes of the underlying system that a practitioner can inspect and interpret. Concept Bottleneck Models (CBMs) (Koh et al., 2020) are the design that later concept-based models for image classification extend (Espinosa Zarlenga et al., 2022; Chen et al., 2025): a network predicts a predefined vocabulary of interpretable attributes, and a second simple model, typically linear models, maps those to the class label. Because the attributes are named, a practitioner can read off which ones drove a prediction and correct a mispredicted one at test time (Koh et al., 2020).

A CBM maps the predicted attributes to the label with a discriminative head. It is common to use a linear head on the predicted attributes for predicting the class label, but since it is additive in the attributes, it represents no interaction among them. Similarly, a nonlinear head represents only the conditional distribution $P(Y \mid A)$. Neither holds a distribution over the attributes themselves, so neither scores a class when some attributes have no evidence, and neither infers an attribute it did not observe. Neural Probabilistic Circuits (NPCs) (Chen et al., 2025) solve these issues by keeping a discrete bottleneck and replace the head with a probabilistic circuit (Choi et al., 2020) that encodes a joint distribution over the attributes *and* the class label. This circuit gives an NPC exactly what the head in existing CBMs lacks: it scores a class from the attributes' joint behavior, so it allows scoring a class under partial evidence and inferring the unobserved attributes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, this expressiveness is expensive at prediction time. To turn the joint distribution into a class prediction, an NPC averages the circuit's class conditional probability $P(Y \mid A)$ over the attribute distribution the neural network outputs. Computing this average exactly visits every joint attribute configuration, so its cost is exponential in the number of attributes. Chen & Zhao (2025) acknowledge this exponential cost and visit only the high-probability configurations of each class to improve average case performance, but the worst case cost is still exponential. As a consequence, so far NPCs have only been studied on image benchmarks with eight attributes or fewer (Chen et al., 2025; Chen & Zhao, 2025), while the datasets that motivate concept bottlenecks are far past it: CUB-200 has 112 annotated attributes and SUN has 102.

In light of the above gap that limits the scalability of NPCs, we propose a novel inference procedure over the same NPC structure: instead of averaging the circuit's class *conditional* $P(Y \mid A)$, we score each class by the *joint* probability mass $P(Y, A)$ the circuit places on it under the network's attribute predictions, and normalize over the classes at the end. The two inference procedures differ in one step. Averaging the conditional divides by the attribute marginal separately at each assignment, and that division inside the sum couples the attribute variables, which forces the enumeration. On the other hand, the joint score carries no division inside the sum, so it is a factorized soft-evidence query, which a smooth and decomposable circuit answers exactly in one forward pass at a cost linear in the circuit's size (Section 4.2). We call the predictor it defines the *generative* inference, and prior work's the *discriminative* inference.

The exponential reduction follows from our key observation about smooth and decomposable circuits: a function that factorizes over the leaves can be absorbed into the leaf factors themselves. Each leaf averages its own variable against the evidence for that variable, and every node above it runs unchanged. Decomposability makes the step valid at a product node, since the evidence splits across the node's disjoint scopes, and smoothness makes it valid at a sum node, since the children of a sum node range over the same variables. The sum over attribute configurations therefore never has to be formed, and a class score costs one evaluation of the circuit. The same pass is also differentiable and since learning treats inference as a sub-procedure, so both training and model selection can reuse the efficient generative inference procedure to accelerate their own as well.

Our contributions are as follows. (1) We show that generative inference is a factorized soft-evidence query, so a smooth and decomposable circuit computes every class score and their normalizer in time linear in the circuit's size, which makes NPC prediction linear in the circuit size and attribute count (Theorem 1). (2) We give a compositional total-variation bound on the distance between our inference and the true class posterior. It splits that distance into a term on the class simplex and a term on the attribute assignment, and it holds pointwise at each input (Section 5). (3) Because the inference is differentiable, it also serves as the training objective, so training, model selection and inference can all benefit from the improved inference complexity. (4) We evaluate on six benchmarks with 2 to 112 attributes. Where full enumeration is feasible, our inference matches prior work's to within 2.29 points, and on the two benchmarks where it is infeasible to enumerate, our method outperforms the sampled estimate by $+10.01$ and $+18.47$ points at a fraction of its circuit time (Section 6).

2 Related Work

Concept-based interpretable models. Concept Bottleneck Models (Koh et al., 2020) predict the class through an intermediate concept layer, and the family's variants differ in how they build the concept set (Oikarinen et al., 2023), how they propagate concept uncertainty (Kim et al., 2023) and how they model dependence among the concepts (Vandenhirtz et al., 2024), each still predicting the class using a linear map; CEM (Espinosa Zarlenga et al., 2022) passes the task predictor a per-concept embedding instead, and ECBM (Xu et al., 2024) scores input, concept and class tuples with trained energy networks; Poeta et al. (2025) survey the wider concept-based family. NPCs (Chen et al., 2025) replace this map with a probabilistic circuit, which introduces a sum over every attribute assignment, and variants of NPCs like RNPC (Chen & Zhao, 2025) and CNPC (Chen & Zhao, 2026) don't change the worst-case exponential cost of that sum which our inference method addresses.

Probabilistic circuits and tractable inference. PCs (Choi et al., 2020) unify Sum-Product Networks (Poon & Domingos, 2011) and related tractable models, and a smooth and decomposable circuit answers marginal queries in time linear in its size. Prior work studies the two regimes on the *learning*

side for sum-product networks, generatively (Poon & Domingos, 2011) and discriminatively (Gens & Domingos, 2012); we apply the same distinction to inference for NPCs. On the error side, Chen et al. (2025) and Chen & Zhao (2025) each bound a two-module predictor's error by a sum of per-module terms, and Kang et al. (2024) compose per-module robustness certificates end to end. Our Theorem 2 provides a error decomposition for generative inference and it holds pointwise at each input, and it averages its circuit-side term under the model's own posterior. This posterior is known, so the weighting needs no estimation. Our target meets the tractability conditions of the compositional atlas of Vergari et al. (2021) and the discriminative target does not, and no linear-cost method is known for the discriminative quotient (Remark 2).

3 PRELIMINARIES

3.1 NOTATION AND ASSUMPTIONS

Let X denote the input and $Y \in \mathcal{Y}$ the class label, with $C := |\mathcal{Y}|$ classes. The concept representation is a vector of K attributes $A = (A_1, \ldots, A_K)$, where attribute A_k, the *group* k, takes one of $m_k := |\mathcal{A}_k|$ mutually exclusive values; a binary attribute is the case $m_k = 2$. An assignment $a = (a_1, \ldots, a_K)$ gives every attribute a value, so total possible assignments are $\prod_k m_k$. The neural attribute model f_θ, the recognizer, outputs one distribution per group, $\boldsymbol{\pi}^{(k)}(x) \in \Delta^{m_k - 1}$, and we write $\pi_k(a_k \mid x)$ for the probability it gives value a_k in group k. Both inference procedures consume the product $\pi(a \mid x) := \prod_k \pi_k(a_k \mid x)$, abbreviated $\pi(a)$ once x is fixed. The circuit g_ω, with parameters ω, encodes a nonnegative joint score $\tilde{P}_\omega(A, Y) \geq 0$, which need not be normalized (Remark 1), with attribute score $\tilde{P}_\omega(a) := \sum_y \tilde{P}_\omega(a, y)$ and induced class conditional $P_\omega(Y \mid a) := \tilde{P}_\omega(a, Y)/\tilde{P}_\omega(a)$. This conditional is defined wherever $\tilde{P}_\omega(a) > 0$. The leaf distributions and sum weights here are softmax-parameterized, hence strictly positive, so it is defined everywhere. P_D is the true data distribution, and $\|P - Q\|_{\mathrm{TV}} := \frac{1}{2}\|P - Q\|_1$.

Assumption 1 (Attribute sufficiency)**.** *$Y \perp\!\!\!\perp X \mid A$, so $P_D(y \mid x) = \sum_a P_D(y \mid a)\, P_D(a \mid x)$.*

Assumption 2 (Conditionally independent attributes)**.** *$A_1, \ldots, A_K$ are mutually independent given X, so $P_D(a \mid x) = \prod_k P_D(a_k \mid x)$.*

These are Assumptions 1 and 2 of Chen et al. (2025) and are standard in the NPC and CBM literature. Assumption 1 says the attribute vector loses no class-relevant information, so the bottleneck reparameterizes the prediction problem rather than compressing it; where a vocabulary does not satisfy it, the information the concepts miss is what a soft bottleneck recovers through leakage (Havasi et al., 2022). Assumption 2 allows us to model the attribute distribution as a factorized product over individual attribute groups i.e. it makes the product $\pi(a \mid x)$ a model of the attribute posterior, and both inference procedures use this product. It allows us to directly feed the neural network generated K per-group distributions as soft evidence into the circuit leaves (Theorem 1).

3.2 PROBABILISTIC CIRCUITS

The circuit g_ω of Section 3.1 is a probabilistic circuit (PC), also denoted by $\mathcal{S}$. A PC (Choi et al., 2020) is a directed acyclic computational graph whose leaf nodes represent nonnegative functions of individual variables, whose product nodes multiply their children, and whose sum nodes take weighted sums of their children with nonnegative weights η. For a node v, let $\mathrm{sc}(v)$ denote its scope, $\mathrm{ch}(v)$ denote its children and $P_v(\mathbf{z})$ denote its output computed by the PC for variable assignment $\mathbf{z}$. The function computed at v is

$$P_v(\mathbf{z}) = \begin{cases} f_v(z_{\mathrm{sc}(v)}) & v \text{ is a leaf,} \\ \prod_{c \in \mathrm{ch}(v)} P_c(\mathbf{z}) & v \text{ is a product node,} \\ \sum_{c \in \mathrm{ch}(v)} \eta_{v,c} P_c(\mathbf{z}) & v \text{ is a sum node.} \end{cases} \tag{1}$$

The value $\tilde{P}_\omega$ at the root is obtained by evaluating the circuit bottom-up, with each node combining the values computed by its children; this takes $O(|\mathcal{S}|)$ time. The circuit is *decomposable* if the children of every product node have disjoint scopes and *smooth* if the children of every sum node have the same scope. Under these conditions, marginal queries can also be answered in one bottom-up pass: for a variable set $\mathbf{Z}_M$, replace each leaf over a marginalized variable by $\sum_z f_v(z)$, while leaving all

other nodes unchanged. The root then computes the corresponding marginal of $\tilde{P}_\omega$ exactly (Choi et al., 2020). The circuit size $|\mathcal{S}|$ is the number of sum and product edges plus the number of leaf parameters.

3.3 Neural Probabilistic Circuits

An NPC (Chen et al., 2025) pairs such a circuit with a neural network based feature recognizer, with the class label among the circuit's variables. Chen et al. (2025) predict by contracting the circuit's conditional probability table $T(y)_a := P_\omega(Y{=}y \mid A{=}a)$ against the outer product of the recognizer's distributions, giving the conditional predictive probability which we denote as $h^{\text{disc}}(y \mid x)$:

$$h^{\text{disc}}(y \mid x) \;=\; \sum_a T(y)_a \prod_{k=1}^{K} \pi_k(a_k \mid x) \;=\; \langle T(y),\; \boldsymbol{\pi}^{(1)}(x) \otimes \cdots \otimes \boldsymbol{\pi}^{(K)}(x) \rangle. \tag{2}$$

We call (2) *discriminative* inference, since it averages the circuit's class conditional. Prior work computes it by materializing $T(y)$ over all $\prod_k m_k$ attribute assignments, one pass over the circuit for each assignment and class, the same per-pass accounting Theorem 1 uses, so building the table costs $O(|\mathcal{S}|C\prod_k m_k)$ and using it costs $O(C\prod_k m_k)$ per input, both exponential in K. Section 4 introduces our generative inference, which computes a different quantity at a cost linear in $|\mathcal{S}|$.

4 Generative Inference of Neural Probabilistic Circuits

We now describe our generative inference method. We first define the unnormalized per-class score our inference method generates and the normalization which yields class probabilities. We then show that this score is a factorized soft-evidence query, so one pass over the circuit computes it, and that the same step does not apply to prior work's inference. We provide the pseudocode for our procedure in Appendix C.

4.1 The Generative Class Score

For a trained NPC, define the class score

$$s_y(x) \;:=\; \mathbb{E}_{\boldsymbol{\pi}(x)}\Big[\tilde{P}_\omega(A, Y{=}y)\Big] \;=\; \sum_a \tilde{P}_\omega(a, y) \prod_{k=1}^{K} \pi_k(a_k \mid x), \tag{3}$$

the joint mass the circuit places on class y under the evidence the network supplies for x. Note that the above scores are not necessarily probabilities since they do not sum to one, so the *generative NPC inference* normalizes them, i.e.,

$$h^{\text{gen}}(y \mid x) \;:=\; \frac{s_y(x)}{\sum_{y'} s_{y'}(x)}, \qquad \hat{y} \;=\; \arg\max_y s_y(x), \tag{4}$$

and that normalizer is a sum over the C classes, not a partition function over attribute space. Marginalizing Y gives the same sum in one further pass (Theorem 1). The pass returns $\log s_y(x)$, so the implementation applies a softmax to the C log scores, and the training objective of Section 4.4 is the cross entropy on them.

Remark 1 (Unnormalized circuit)**.** *Every term here is invariant to an input-independent rescaling of $\tilde{P}_\omega$, so all results hold for the unnormalized circuit and we compute no partition function over attribute space. The pass and Theorem 1 need nothing beyond decomposability and smoothness; nonnegativity makes $\tilde{P}_\omega$ a score and lets the pass run in log space. Theorem 2 needs strict positivity as well, which makes the conditional $P_\omega(Y \mid a)$ and the reweighted posterior of Eq.* (7) *defined everywhere.*

4.2 The Class Score as a Factorized Soft-Evidence Query

A *factorized soft-evidence* query gives each variable a distribution over its values instead of a single observed value, and asks for the expectation of the circuit's function under those distributions. The

class score of Eq. (3) is a query of this kind. In general, writing w_k for per-group weights that are nonnegative and need not sum to one,

$$\mathbb{E}_w\Big[\tilde{P}_\omega(A, Y{=}y)\Big] \;=\; \sum_a \tilde{P}_\omega(A{=}a, Y{=}y) \prod_{k=1}^{K} w_k(a_k), \tag{5}$$

and the class score is the case $w_k = \pi_k(\cdot \mid x)$. A smooth and decomposable PC computes the left side without forming the sum explicitly over all the possible enumeration of K attributes on the right. Decomposability of the PC lets it factor through the product nodes, each leaf averaging its own function against the evidence for its own variable, and no step uses normalization of $\tilde{P}_\omega$. We have the following theorem that formalizes the above discussion:

Theorem 1 (Time complexity of generative inference)**.** *Let $\mathcal{S}$ be decomposable and smooth, encoding a nonnegative function $\tilde{P}_\omega(A, Y) \geq 0$, and let the evidence factorize across groups. For any x and any $y \in \mathcal{Y}$, one soft-evidence forward pass with leaf evidence $\boldsymbol{\pi}^{(k)}(x)$ computes $\mathbb{E}_{\boldsymbol{\pi}(x)}[\tilde{P}_\omega(A, Y{=}y)]$ in time $O(|\mathcal{S}|)$: replace $P_v(A_k{=}a_k)$ at each leaf of scope $\{k\}$ with $\sum_b \pi_k(b)\, P_v(A_k{=}b)$, clamp Y to y, and propagate unchanged. Marginalizing Y instead of clamping it returns the normalizer $Z_\omega(x) = \mathbb{E}_{\boldsymbol{\pi}(x)}[\tilde{P}_\omega(A)]$ the same way, so the C class scores and the normalizer together cost $O(C|\mathcal{S}|)$.*

Note that in Theorem 1 we do not assume the function $\tilde{P}_\omega$ that the circuit encodes to be normalized. Appendix B proves the pass for arbitrary nonnegative per-group weights, not only for evidence that sums to one, which Remark 2 and Algorithm 1 needs. We note that similar observations exist in other tractable probabilistic models beyond our NPCs. In particular, Theorem 1 of Liu et al. (2024) states it for a normalized circuit over categorical variables, which they use to steer a diffusion model, and it also follows from the composition rules of Vergari et al. (2021), since a factorized $\boldsymbol{\pi}$ multiplies into any smooth and decomposable circuit at linear cost and the product is then integrated in linear time. Our contribution is the identification that the NPC class score when written in our generative form is a query of this form, so one pass over the PC computes it.

4.3 Comparison to the Discriminative Inference

The discriminative inference, i.e., $\arg\max_y \mathbb{E}_{\boldsymbol{\pi}}[P_\omega(Y \mid A)]$ of Eq. (2), is the Bayes rule for $0/1$ loss under two conditions: the modeled conditional and $\boldsymbol{\pi}(\cdot \mid x)$ are correct, and Assumption 1 holds. We point out that the generative inference does not follow the Bayes rule in general. Instead, by defining a nonnegative score in Eq. (3), the generative inference is more akin to the way energy-based models or classic Markov networks parametrize. The two coincide when $\tilde{P}_\omega(A)$ is constant on the support of $\boldsymbol{\pi}(x)$.

Remark 2 (The per-assignment conditional makes discriminative inference intractable)**.** *Both terms of* (4) *are soft-evidence expectations of the joint score, so the generative inference is one pass of Theorem 1 and its single division is over the C class outputs (it depends on the PC implementation, for example if PC is implemented such that it encodes each class as an indicator variable based leaf node, then one can estimate the denominator with another pass of the PC by setting each leaf as 1. In our implementation we get the score for each class in a single forward pass of the PC). The discriminative inference needs the per-assignment conditional $\tilde{P}_\omega(a, y)/\sum_{y'} \tilde{P}_\omega(a, y')$, which divides* inside *the sum over a. The obstruction is the summand: the per-assignment quotient is not the function $\mathcal{S}$ encodes, and a quotient of two circuits has no known smooth and decomposable representation of comparable size. That obstruction is conditional: it lifts if the attribute score factorizes across groups, $\tilde{P}_\omega(a) = \prod_k \psi_k(a_k) > 0$, since the weights $\pi_k(a_k)/\psi_k(a_k)$ are then univariate and the same pass applies (Appendix B), but this condition may not hold in general for real data distributions.*

We therefore compute the discriminative inference only as a baseline, by enumeration or by sampling (Section 6).

4.4 Learning NPCs through Generative Inference

We keep the three-stage recipe of Chen et al. (2025). Stage 1 trains the attribute recognizer. Stage 2 fits the circuit to attribute vectors in two phases, a generative phase on $\log \tilde{P}_\omega(a, y)$ and then a

discriminative phase on $\log P_\omega(y \mid a)$. Stage 3 fine-tunes both modules against a class loss which we modify for each inference method. Specifically, each model is trained against the inference procedure it is evaluated with for fair comparison in Section 6. The generative inference is a differentiable function of both modules' parameters, so we minimize $\mathcal{L}^{\text{exact}} = -\log h^{\text{gen}}(y \mid x)$ with the log scores of Eq. (3) from the same single pass, and we select the epoch on the validation accuracy of that same quantity, so training, model selection and inference target one functional (Appendix D.3.1).

Training the baselines. Prior work's enumeration based inference trains the Stage 3 objective $\mathcal{L}^{\text{enum}} = -\log \sum_a \pi(a \mid x)\, P_\omega(y \mid a)$, which evaluates the circuit at every one of the $\prod_k m_k$ assignments at every optimizer step. Inference can build that table once per split and training cannot, because ω moves, so this objective is infeasible at large K. There we train the baseline against a sampled estimate of it. We draw n assignments $a^{(1)}, \ldots, a^{(n)}$ from $\bar{\boldsymbol{\pi}}(x) = \text{stopgrad}\, \boldsymbol{\pi}(x)$, the recognizer's own distribution with its gradient stopped, and give draw s the weight $\varrho_s = \pi(a^{(s)} \mid x)/\bar{\pi}(a^{(s)} \mid x)$. The objective is

$$\mathcal{L}^{\text{samp}} = -\log \frac{\widehat{\mu}(y)}{\sum_{y'} \widehat{\mu}(y')}, \qquad \widehat{\mu}(y) = \frac{1}{n} \sum_{s=1}^{n} \varrho_s\, P_\omega\big(y \mid a^{(s)}\big). \tag{6}$$

Each ϱ_s equals one in value and still carries the recognizer's gradient, since the measure the draws are taken under is a constant. $\mathcal{L}^{\text{samp}}$ averages $P_\omega(y \mid a)$ over the draws and takes the logarithm after, so it estimates $\mathcal{L}^{\text{enum}}$ itself, while averaging the logarithm over draws estimates a different quantity that Jensen's inequality separates from it at every n (Appendix D.3.1). All three losses are differentiable in the circuit and in the recognizer, and every run here updates both. The trainer released with Chen et al. (2025) updates the recognizer alone and holds the circuit fixed, as their training details state and their method section does not.

Our claim is not that this objective is the most accurate wherever enumeration is affordable. We evaluate each model with the inference procedure its own objective targets. Ours is behind by at most 2.29 ± 1.60 points on the four benchmarks where the enumerated objective runs, and ahead by 10.01 ± 1.02 points on CUB-200 and 18.47 ± 1.19 points on SUN where only the sampled estimator can be trained (Section 6.3). Our objective is exact at every K. On CUB-200 and SUN the enumerated objective cannot be formed at all, and ours is more accurate than the sampled estimator.

5 Performance Guarantee of Generative Inference

We now bound how far the generative inference is from the true class posterior. Define the reweighted attribute posterior

$$q_\omega(a \mid x) \;:=\; \frac{\tilde{P}_\omega(a)\, \pi(a \mid x)}{Z_\omega(x)}, \qquad Z_\omega(x) \;:=\; \sum_{a'} \tilde{P}_\omega(a')\, \pi(a' \mid x). \tag{7}$$

This is a proper distribution whenever $Z_\omega(x) > 0$, which the strict positivity of Section 3.1 gives, and it is invariant to global rescaling of $\tilde{P}_\omega$. As we discussed in the last section, its defining sum has $\prod_k m_k$ terms but is itself a soft-evidence query, so one pass returns it in $O(|\mathcal{S}|)$ time. The posterior q_ω is the attribute distribution the composed model reasons with: the recognizer's belief, reweighted by how plausible the circuit finds each configuration a priori. Substituting $\tilde{P}_\omega(a, y) = P_\omega(y \mid a)\, \tilde{P}_\omega(a)$ into (4) and dividing by $Z_\omega(x)$ regroups the inference into a two-stage mixture,

$$h^{\text{gen}}(y \mid x) \;=\; \sum_a P_\omega(y \mid a)\, q_\omega(a \mid x), \tag{8}$$

in which q_ω carries everything the attribute side contributes and $P_\omega(Y \mid a)$ everything the class-label side does.

Component error terms. The mixture form gives one error term per stage. Define

$$\text{TV}_C(a) := \big\| P_\omega(Y \mid a) - P_D(Y \mid a) \big\|_{\text{TV}}, \tag{9}$$

$$\mathcal{E}_C^{\text{TV}}(x) := \mathbb{E}_{a \sim q_\omega(\cdot \mid x)}\, \text{TV}_C(a), \tag{10}$$

$$\text{TV}_F(x) := \big\| q_\omega(A \mid x) - P_D(A \mid x) \big\|_{\text{TV}}. \tag{11}$$

Table 1: Test accuracy (%), top-1, mean ± s.d. over 3 seeds. Bold marks the best of three by the rule of Appendix D.1. No bold on AwA2, CelebA and MNIST-Add, where the gap is inside its own seed variation. None on GTSRB, where the top two print the same value. "—" marks an attribute space too large to enumerate.

				Baselines		Inference, on its own objective			
Dataset	K	C	$\prod_k m_k$	DNN	CBM	Gen	Disc-Enum	Disc-MC	n
MNIST-Add	2	19	10^2	98.95 ± 0.14	98.91 ± 0.12	98.98 ± 0.12	98.91 ± 0.05	98.84 ± 0.22	100
GTSRB	4	43	3,120	99.86 ± 0.08	99.92 ± 0.05	99.93 ± 0.04	99.93 ± 0.05	99.92 ± 0.01	100
AwA2	4	50	3,402	93.22 ± 0.12	91.91 ± 0.64	87.15 ± 1.76	89.44 ± 0.22	89.40 ± 0.19	100
CelebA	5	127	108	36.78 ± 0.06	36.80 ± 0.07	34.57 ± 0.10	34.93 ± 0.19	34.85 ± 0.06	107
CUB-200	112	200	2^{112}	81.00 ± 1.48	81.75 ± 0.28	**79.18 ± 1.12**	—	69.17 ± 1.93	113
SUN	102	717	2^{102}	51.49 ± 0.95	47.31 ± 0.91	**32.17 ± 2.03**	—	13.70 ± 2.64	129

Equation (9) is the *circuit-side* error on the class simplex, and Equation (10) averages it over the distribution induced by the composed NPC. This average reaches every assignment, since q_ω gives all of them mass, including assignments the data never produces, so at those samples data distribution supplies no $P_D(Y \mid a)$ to compare against, and we set $\mathrm{TV}_C(a) := 0$ (Appendix A). Equation 11 is the error of the *reweighted attribute posterior* over the attribute space.

Theorem 2 (Compositional TV bound)**.** *Assume Assumption 1 and $\tilde{P}_\omega > 0$, which makes $P_\omega(Y \mid a)$ and q_ω well defined. Then for every input x,*

$$\left\| h^{\mathrm{gen}}(\cdot \mid x) - P_D(\cdot \mid x) \right\|_{\mathrm{TV}} \;\le\; \mathrm{TV}_F(x) + \mathcal{E}_C^{\mathrm{TV}}(x), \tag{12}$$

and $\mathbb{E}_X \| h^{\mathrm{gen}}(\cdot \mid X) - P_D(\cdot \mid X) \|_{\mathrm{TV}} \le \mathbb{E}_X\, \mathrm{TV}_F(X) + \mathbb{E}_X\, \mathcal{E}_C^{\mathrm{TV}}(X)$.

Proof sketch. Adding and subtracting $\sum_a P_D(y \mid a)\, q_\omega(a \mid x)$ in the mixture difference splits the gap into a circuit-side term (the conditional mismatch averaged under q_ω) and a recognizer-side term (the posterior mismatch). Summing $|\cdot|$ over y, applying the triangle inequality inside the sum over a, and halving identifies the two terms; Appendix A gives the full derivation. □

The bound is *pointwise*, so it can flag individual inputs on which the composition is untrustworthy, and both terms have computable surrogates on a trained model. We treat those surrogates as a diagnostic of where the composition's error lies, and empirically the recognizer-side surrogate is informative where the attribute space is small and saturates on the four benchmarks where it is large.

6 Experiments

We measure two things on small and large attribute space datasets: (1) inference time for our generative inference method against discriminative inference, and (2) how the different inference procedures compare in accuracy.

6.1 Experimental Setup

Datasets. We use six image benchmarks, which span more than an order of magnitude in both K and C (Table 1). They are MNIST-Add, a pair of MNIST digits (LeCun et al., 1998) labeled by their sum after Manhaeve et al. (2018), GTSRB (Stallkamp et al., 2011), AwA2 (Xian et al., 2019), CelebA (Liu et al., 2015), CUB-200 (Wah et al., 2011) and SUN (Patterson & Hays, 2012). Splits, vocabulary and group cardinalities are in Appendix D. CUB-200 and SUN motivate this work because their 2^{112} and 2^{102} assignments cannot be enumerated.

Baselines. We study three inference procedures for an NPC.

- **Disc-Enum**: the enumerated procedure of Chen et al. (2025), Eq. (2), computed exactly over all $\prod_k m_k$ assignments and available only at small K.
- **Disc-MC-n**: the same target approximated from n attribute samples, and the only discriminative reference we can run on CUB-200 and SUN.

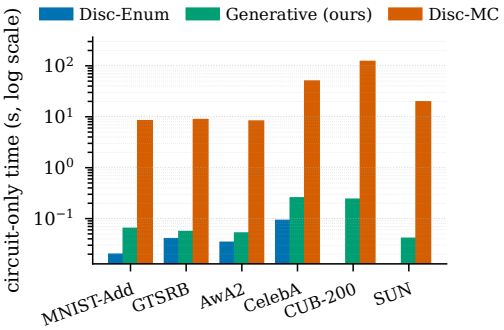

Figure 1: Circuit-only time per test split, log scale, mean of three seeds (values in Table 5). A missing Disc-Enum bar means enumeration is infeasible.

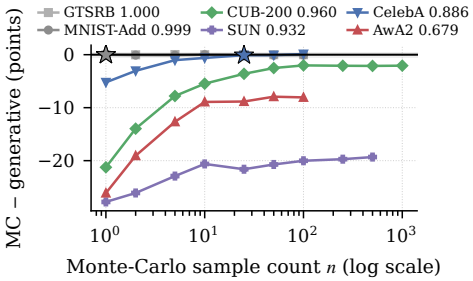

Figure 2: Monte-Carlo minus generative accuracy on one checkpoint, by sample count, at seed 42. Stars mark the smallest n within 0.5 points and the band is that tolerance. Each legend entry is a benchmark and that benchmark's attribute-head concentration $\bar{\pi}_{\max}$, the probability the head puts on its most likely value, averaged over groups and images.

- **Gen**: our inference procedure, computed exactly in one soft-evidence pass.

We measure each procedure on the checkpoint trained against the objective that procedure targets, so training and inference are aligned in every comparison of Table 1. Evaluating all three procedures on one checkpoint instead, with that checkpoint's recognizer and circuit both held fixed, isolates the inference procedure from the objective it was trained against (Appendix E). We also train a concept-bottleneck baseline on the same backbone for the same splits (Appendix D.3.3).

Metrics. *Accuracy* is top-1 test accuracy. *Inference time* is wall-clock seconds to score the full test split on the circuit alone. The recognizer runs once and we cache its outputs, so the measurement leaves out image loading and the backbone, which cost the same for all three procedures.

Implementation. Every model pairs a ResNet (He et al., 2016) attribute head (ResNet-34 on the four small-K benchmarks, ResNet-50 on CUB-200 and SUN) with an EiNet (Peharz et al., 2020) circuit on the RAT-SPN template (Peharz et al., 2019), whose size is linear in K, trained in the three stages of Chen et al. (2025) (Appendix D). CUB-200 follows the data-processing of Koh et al. (2020) and for SUN, we resize the whole frame and supervise on each image's own annotation, because its attributes describe the scene and not a centred object.

6.2 Accuracy and Inference Time

Speed. The generative inference scores a full test split in 0.041 to 0.255 s of circuit time on every benchmark (Figure 1 and Table 5). Split sizes differ, so the comparable quantity is per-example cost, which varies by a factor of only 2.3 across a 56-fold increase in K, over which the number of configurations rises from 10^2 to 2^{112}. This is because the cost of our method is proportional to the circuit size and not the total configuration count which can increase exponentially faster (Theorem 1).

Enumeration, measured the same way, costs 0.31 to 0.72 times the generative pass on the four small-K benchmarks. Its conditional table does not depend on the input, so the implementation builds it once per split (Appendix D.3.2). At small K the generative pass is therefore not faster. But, on CUB-200 and SUN only sampling can compute the discriminative inference, and at the sample counts of Table 1 it costs $508\times$ the generative pass on CUB-200 and $478\times$ on SUN. That ratio is 3.7 to 4.5 times the draw count, so a draw costs more than one pass. A draw also samples the K groups and evaluates the circuit at the resulting hard assignment, once per draw. We set each benchmark's draw count from the baseline's own guarantee: Hoeffding with a union bound over the C classes guarantees $n \geq \log(2C/\delta)/(2\varepsilon^2)$ draws enough to hold every class probability within ε with probability $1-\delta$, which at $\varepsilon = 0.20$ and $\delta = 0.05$ asks for eighty-three to 129 draws, and we run $\max(100, n)$ (Appendix D.3.2). Note that estimation error (ε) lower than 0.2 would scale the number of samples and proportionally the inference cost for the sampled baseline. These ratios depend on n, so Figure 2 varies it for each dataset and even at the smallest budget that matches the generative inference accuracy, sampling costs $4.6\times$ the single pass on CelebA, the only benchmark that reaches parity at any swept budget (Appendix D.3.3).

Accuracy. We compare two models that share a training seed, take the difference on each seed, and call it consistent when every seed gives it the same sign and its mean exceeds the spread of those per-seed differences (Appendix D.1). We compare the generative inference against the discriminative inference on each benchmark. Aligning each model with the inference its objective targets is also the right pairing to report: every benchmark and objective gives one trained model, sixteen in all once the enumerated objective is dropped on CUB-200 and SUN, and on fourteen of them the model is most accurate under the inference its own objective targets, and the two exceptions are both on CelebA, where another inference is ahead by at most 0.20 ± 0.15 points. On the four benchmarks where enumeration runs, ours is behind by 2.29 ± 1.60 points on AwA2 and 0.36 ± 0.11 points on CelebA, and level on GTSRB ($+0.00 \pm 0.03$) and MNIST-Add ($+0.06 \pm 0.16$), showing minimal degradation in performance compared to discriminative baselines. On the two where only sampling runs, generative inference is ahead by 10.01 ± 1.02 points on CUB-200 and 18.47 ± 1.19 points on SUN while being much faster than the sampling based discriminative inference, showing the effectiveness of our method for large attribute regime. A concept-bottleneck baseline on the same backbone is more accurate than our inference on four of six benchmarks. Every NPC inference in Table 1 is behind it on those benchmarks, so what the bottleneck costs is not a property of the inference procedure this paper changes.

6.3 The Choice of Stage-3 Objective

Stage 3's class loss need not match the procedure used at test time. Ours is $-\log h^{\text{gen}}(y \mid x)$, exact in one pass at any K, while the objective of Chen et al. (2025) enumerates all $\prod_k m_k$ assignments. We train three objectives on byte-identical Stage-1 heads. These are the enumerated one, the same quantity from ten draws, and ours, and we select each model on the quantity its objective trains against. We then evaluate each model under the inference its own objective targets (Table 6). On the four benchmarks where all three objectives run, the three agree to within 2.29 points. On CUB-200 and SUN the enumerated objective cannot be trained at all. Only the sampled objective and ours can be trained there, and ours is ahead by 10.01 ± 1.02 points on CUB-200 and 18.47 ± 1.19 points on SUN. Ours is also the only one of the three that stays exact at K=112 (Appendix E).

7 Conclusion

The cost of NPC inference comes from the quantity computed, not from the architecture. Scoring each class by joint mass rather than averaging the class conditional makes classification a soft-evidence query, exact in one $O(|\mathcal{S}|)$ pass and usable as a training objective too. That lifts NPC inference from the ten attributes of the published NPC work to 112. Against prior methods, the new inference method is behind by at most 2.29 ± 1.60 points on the four benchmarks where we can run the enumeration, and ahead by up to 18.47 ± 1.19 points on the two where only sampling can be run for previous methods. On all six benchmarks it uses $130\times$ to $508\times$ less circuit time than sampling.

Reproducibility Statement

For every dataset and training configuration reported here we train three models and evaluate each under three Monte-Carlo seeds. Appendix D gives the backbones, the circuit geometries, the training schedules and the image transform we apply to each split at evaluation. Appendix D.1 states the rule every comparison follows. Appendix C gives the pass as pseudocode. The code that trains the models shall be released on acceptance.

Ethics Statement

We study inference algorithms on standard vision benchmarks (MNIST-Add, GTSRB, AwA2, CelebA, CUB-200, SUN) and collect no new human-subjects data. CelebA contains facial images of real identifiable people, and its attribute vocabulary includes a subjective judgement about physical appearance (Appendix D). We use it only for the attribute-recognition and accuracy comparisons already established in the NPC literature, and train no identity or demographic classifier. Its splits shuffle over images rather than over identities. Computing a different quantity from a fixed trained circuit adds no privacy or fairness risk of its own. Appendix E shows that joint training can raise class accuracy while the attribute layer drifts from its annotation, so anyone who uses its stated concepts as an explanation should check both its class accuracy and the fidelity of its attributes.

AI Use Statement

We used large language models to polish our writing and to draft sections of this paper, to search for related work, to develop the research ideas and run the experiments, and to work on the proofs. They also wrote and revised the source code that is not the model itself, which covers the pipeline that trains and evaluates the models and the scripts that clean and reformat the data it consumes. The one synthetic benchmark we report comes from the published generator we cite, not from a language model. We read every item we cite at its source, checked every proof and every number ourselves, and take responsibility for everything in this paper.

References

Steven Braun. Simple-einet: An EinsumNetworks Implementation, 2021. URL `https://github.com/braun-steven/simple-einet`. Software library. The author's own entry gives no year, so this is the repository creation year. Version used: commit 9c71640b6161b4338684d9267745c5299b43bf96 of 26 January 2026.

Weixin Chen and Han Zhao. Understanding and improving adversarial robustness of neural probabilistic circuits. In *NeurIPS*, 2025. arXiv:2509.20549.

Weixin Chen and Han Zhao. Causal neural probabilistic circuits. *arXiv preprint arXiv:2603.01372v1*, 2026.

Weixin Chen, Simon Yu, Huajie Shao, Lui Sha, and Han Zhao. Neural probabilistic circuits: Enabling compositional and interpretable predictions through logical reasoning, 2025. URL `https://arxiv.org/abs/2501.07021`.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report, University of California, Los Angeles, 2020. `http://starai.cs.ucla.edu/papers/ProbCirc20.pdf`.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lió, and Mateja Jamnik. Concept embedding models: Beyond the accuracy-explainability trade-off. In *NeurIPS*, 2022.

Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *NeurIPS*, 2012.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In *NeurIPS*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Mintong Kang, Nezihe Merve Gürel, Linyi Li, and Bo Li. COLEP: Certifiably robust learning-reasoning conformal prediction via probabilistic circuits. In *ICLR*, 2024. doi: 10.48550/arXiv.2403.11348. URL https://openreview.net/forum?id=XN6ZPINdSg.

Eunji Kim, Dahuin Jung, Sangha Park, Siwon Kim, and Sungroh Yoon. Probabilistic concept bottleneck models. In *ICML*, 2023.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *ICML*, 2020.

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

Anji Liu, Mathias Niepert, and Guy Van den Broeck. Image inpainting via tractable steering of diffusion models. In *ICLR*, 2024. URL https://openreview.net/forum?id=NSIVHTbZBR.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *NeurIPS*, 2018.

Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *ICLR*, 2023.

Genevieve Patterson and James Hays. SUN attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, pp. 2751–2758, 2012. doi: 10.1109/CVPR.2012.6247998.

Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *UAI*, 2019.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *ICML*, 2020.

Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey. *ACM Computing Surveys*, 2025. doi: 10.1145/3774643.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *UAI*, 2011.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IJCNN*, 2011.

Moritz Vandenhirtz, Sonia Laguna, Ričards Marcinkevičs, and Julia E. Vogt. Stochastic concept bottleneck models. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2406.19272.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *NeurIPS*, 2021.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019.

Xinyue Xu, Yi Qin, Lu Mi, Hao Wang, and Xiaomeng Li. Energy-based concept bottleneck models: Unifying prediction, concept intervention, and probabilistic interpretations. In *ICLR*, 2024. URL https://arxiv.org/abs/2401.14142.

A Proof of Theorem 2 (TV-in-TV compositional bound)

We write out the steps of the argument sketched in Section 5.

Proof. By Remark 1 and Eq. (8), $q_\omega(\cdot \mid x)$ is a distribution over a and $P_\omega(Y \mid a)$ is a distribution over y for the nonnegative circuit $\tilde{P}_\omega$, and $h^{\text{gen}}(\cdot \mid x)$ is a distribution over y. No step below uses $\sum_{a,y} \tilde{P}_\omega(a, y) = 1$. At an assignment with $P_D(a) = 0$ we read $P_D(Y \mid a)$ as $P_\omega(Y \mid a)$, which is the convention $\mathrm{TV}_C(a) := 0$ of Section 5, and the argument does not depend on that convention. The proof uses $P_D(\cdot \mid a)$ only in a term it adds and subtracts, which cancels, and through its summing to one over y. Fix x. Assumption 1 gives $P_D(y \mid x) = \sum_a P_D(y \mid a)\, P_D(a \mid x)$, and (8) gives $h^{\text{gen}}(y \mid x) = \sum_a P_\omega(y \mid a)\, q_\omega(a \mid x)$. Adding and subtracting $\sum_a P_D(y \mid a)\, q_\omega(a \mid x)$,

$$\begin{aligned} h^{\text{gen}}(y \mid x) - P_D(y \mid x) \;&=\; \sum_a \big[P_\omega(y \mid a) - P_D(y \mid a)\big]\, q_\omega(a \mid x) \\ &\quad + \sum_a P_D(y \mid a)\, \big[q_\omega(a \mid x) - P_D(a \mid x)\big]. \end{aligned} \tag{13}$$

Denote the two summands on the right by $U_C(y)$ and $U_F(y)$. Summing $|\cdot|$ over y and using the triangle inequality inside each summand,

$$\begin{aligned} \sum_y |U_C(y)| \;&\le\; \sum_a q_\omega(a \mid x) \sum_y \big|P_\omega(y \mid a) - P_D(y \mid a)\big| \\ &=\; 2 \sum_a q_\omega(a \mid x)\, \mathrm{TV}_C(a) \;=\; 2\, \mathcal{E}_C^{\mathrm{TV}}(x), \end{aligned} \tag{14}$$

$$\sum_y |U_F(y)| \;\le\; \sum_a \big|q_\omega(a \mid x) - P_D(a \mid x)\big| \underbrace{\sum_y P_D(y \mid a)}_{=1} \;=\; 2\, \mathrm{TV}_F(x). \tag{15}$$

Combining and dividing by 2,

$$\big\|h^{\text{gen}}(\cdot \mid x) - P_D(\cdot \mid x)\big\|_{\mathrm{TV}} \;=\; \tfrac{1}{2} \sum_y \big|h^{\text{gen}}(y \mid x) - P_D(y \mid x)\big| \;\le\; \mathcal{E}_C^{\mathrm{TV}}(x) + \mathrm{TV}_F(x),$$

which is (12). Taking $\mathbb{E}_X$ on both sides gives the averaged version. □

B Proof of Theorem 1 (soft-evidence soundness)

We prove the statement for arbitrary nonnegative per-group weights, which is the form Remark 2 needs, and for the unnormalized score the rest of the paper uses.

Theorem (Restatement)**.** *Let $\mathcal{S}$ be smooth and decomposable over $(A_1, \ldots, A_K, Y)$, encoding a nonnegative function $\tilde{P}_\omega(A, Y) \ge 0$, and let the weights be a product across groups of arbitrary nonnegative per-group weights, $w = \bigotimes_k w_k$. Give each leaf of scope $\{k\}$ the value $z_v = \log \sum_b w_k(b)\, P_v(A_k{=}b)$, give each leaf of scope $\{Y\}$ the weight $\mathbf{1}[Y{=}y]$, and propagate upward, where products add and sums take a weighted logsumexp. The root returns $\log \mathbb{E}_w[\tilde{P}_\omega(A, Y{=}y)]$ in $O(|\mathcal{S}|)$ time. Weighting the Y leaves by 1 instead returns $\log \mathbb{E}_w[\tilde{P}_\omega(A)]$, so the C class scores and the normalizer together cost $O(C|\mathcal{S}|)$, and at $w_k = \boldsymbol{\pi}^{(k)}$ that normalizer is $\log Z_\omega(x)$.*

Proof. Structural induction on the circuit's computation graph, in the general weights the theorem states. Write ϕ_v for the scope of node v and P_v for the nonnegative function its subtree encodes. The induction hypothesis is

$$\exp(z_v) \;=\; \sum_{a_{\phi_v}} \Big(\prod_{k \in \phi_v} w_k(a_k) \Big) P_v(a_{\phi_v}), \tag{16}$$

a weighted sum over the assignments of v's own scope. No step below needs each w_k to sum to one over its group, and no step needs $\sum_{a,y} \tilde{P}_\omega(a, y) = 1$.

Base case (leaves). A leaf v of scope $\{k\}$ outputs $z_v(x) = \log \sum_b w_k(b) P_v(A_k{=}b)$, which is (16) with $\phi_v = \{k\}$.

Product nodes. This is the only step that uses decomposability. A product node combines children of disjoint scopes $\phi_1 \sqcup \phi_2$ and its log value is $z_{\text{parent}} = z_{\text{left}} + z_{\text{right}}$. Because the scopes are disjoint and the weight is a product across groups, the double sum splits. The sum $\sum_{a_{\phi_1 \sqcup \phi_2}} \prod_{k \in \phi_1 \sqcup \phi_2} w_k(a_k)\, P_{\text{left}}(a_{\phi_1}) P_{\text{right}}(a_{\phi_2})$ equals the product of the two single-scope sums, which is $\exp(z_{\text{left}} + z_{\text{right}})$ by hypothesis.

Sum nodes. This is the only step that uses smoothness, and it is where the general weights matter. A sum node computes $z = \log \sum_c \eta_{v,c} \exp(z_c)$ with nonnegative weights $\eta_{v,c}$. Smoothness makes every child's scope the same set ϕ, so the factor $\prod_{k \in \phi} w_k(a_k)$ in (16) is one and the same for every child and comes out of the sum over children:

$$\exp(z) \;=\; \sum_c \eta_{v,c} \sum_{a_\phi} \Big(\prod_{k \in \phi} w_k(a_k) \Big) P_c(a_\phi) \;=\; \sum_{a_\phi} \Big(\prod_{k \in \phi} w_k(a_k) \Big) \sum_c \eta_{v,c} P_c(a_\phi), \tag{17}$$

which is (16) for the parent. Without smoothness the children carry different products, with no common factor to extract. Under normalized evidence the step still holds, because a group absent from a child's scope contributes $\sum_b w_k(b) = 1$.

Root. The root's scope is $A \sqcup \{Y\}$, so (16) there becomes $\exp(z_{\text{root}}) = \sum_{a,y'} \big(\prod_k w_k(a_k)\big) \mathbf{1}[y'{=}y]\, \tilde{P}_\omega(a, y') = \mathbb{E}_w[\tilde{P}_\omega(A, Y{=}y)]$. Weighting the Y leaves by 1 gives $\mathbb{E}_w[\tilde{P}_\omega(A)]$ the same way, so one pass per class plus one more for the normalizer cost $O(C|\mathcal{S}|)$ together. At $w_k = \boldsymbol{\pi}^{(k)}(x)$ these are the numerator and the normalizer of the generative inference.

Time. Each sum node performs $O(1)$ work per incoming edge, each product node one log-addition per incoming edge, and each leaf $O(m_k)$ to contract its group's weight vector, one operation per leaf parameter. $|\mathcal{S}|$ counts sum edges, product edges and leaf parameters together (Section 3.2), so one pass is $O(|\mathcal{S}|)$. □

Relation to prior one-pass results. Theorem 1 of Liu et al. (2024) computes $\sum_x \prod_i w_i(x_i)\, p(x)$ for a smooth and decomposable p in time $O(|p|)$, by putting $\sum_{x_i} f_v(x_i) w_i(x_i)$ at each input node. Set $w_k = \boldsymbol{\pi}^{(k)}$ with $w_Y = \mathbf{1}[Y{=}y]$ for our class score and $w_Y \equiv 1$ for $Z_\omega(x)$. Both quantities of Theorem 1 are then the sum their induction computes. Their circuit is normalized and ours is an unnormalized score, and no step of the induction uses normalization. The compositional atlas of Vergari et al. (2021) reaches the same conclusion by composition. A fully factorized $\boldsymbol{\pi}$ is omni-compatible in their Definition 2.6, their Corollary B.1 multiplies it into any smooth and decomposable circuit in $O(|\mathcal{S}|)$, and their Proposition 2.1, restating Choi et al. (2020), integrates the product in $\Theta(|\mathcal{S}|)$. Our contribution is the identification that the NPC class inference is a query of this form, so the pass applies to it.

C THE COMPLETE INFERENCE PROCEDURE

Algorithm 1 writes out the pass of Theorem 1 for a smooth and decomposable circuit over the attributes and the class. Notation follows Section 3.1.

Cost. Each call sweeps the circuit once, so it costs $O(|\mathcal{S}|)$, and the C classes with the normalizer cost $O(C|\mathcal{S}|)$. A hard assignment uses one value per leaf, while this pass uses all m_k, and $|\mathcal{S}|$ counts those values. No step forms the sum over $\prod_k m_k$ assignments.

The same pass as the objective. The score is differentiable in the recognizer's outputs, and in the circuit's parameters when the leaf parameters stay in the autograd graph. Stage 3 minimizes the cross-entropy of $\mathrm{softmax}(\mathrm{score})$ through this pass. That is the objective Section 6.3 compares against the enumerated and the sampled one. Inference detaches the leaf parameters.

Algorithm 1 The generative inference, one pass per class

Require: Smooth and decomposable circuit $\mathcal{S}$ over $(A_1, \ldots, A_K, Y)$ with leaf functions P_v and nonnegative sum weights η; recognizer f_θ; input x

for $k = 1, \ldots, K$ **do**
  $\boldsymbol{\pi}^{(k)}(x) \leftarrow \mathrm{softmax}\big(f_\theta(x)_k\big)$ ▷ one categorical belief per attribute group
end for
function PASS(w_Y)
  for each node v in topological order **do**
    if v is a leaf of scope $\{k\}$ **then**
      $z_v \leftarrow \mathrm{logsumexp}_b\big(\log \pi_k(b \mid x) + \log P_v(A_k{=}b)\big)$ ▷ Eq. (5) at one leaf
    else if v is a leaf of scope $\{Y\}$ **then**
      $z_v \leftarrow \mathrm{logsumexp}_{y'}\big(\log w_Y(y') + \log P_v(Y{=}y')\big)$
    else if v is a product node **then**
      $z_v \leftarrow \sum_c z_c$ over the children of v ▷ disjoint scopes, so the weighted sums factor
    else
      $z_v \leftarrow \mathrm{logsumexp}_c\big(\log \eta_{v,c} + z_c\big)$ over the children of v ▷ one scope, so the evidence factor comes out
    end if
  end for
  return z_v at the root
end function
for $y \in \mathcal{Y}$ **do**
  $\mathrm{score}[y] \leftarrow \textsc{Pass}\big(\mathbf{1}[Y{=}y]\big)$ ▷ $\log \mathbb{E}_{\boldsymbol{\pi}}[\tilde{P}_\omega(A, Y{=}y)]$
end for
$\log Z \leftarrow \textsc{Pass}(1)$ ▷ $\log Z_\omega(x)$ of Eq. (7)
return $\arg\max_y \mathrm{score}[y]$, and $\mathrm{score}[y] - \log Z$ for the normalized prediction

D Detailed Experimental Setup

D.1 Rule for comparing two quantities

Every comparison is between quantities measured on models that share a training seed. The three inference procedures come from one checkpoint, and the Stage-3 objectives are separate runs from one Stage-1 and Stage-2 lineage per seed. We report the mean of the per-seed differences and their standard deviation, and call a difference *consistent* when every seed gives it the same sign and the mean exceeds that standard deviation. We write a difference and a single quantity the same way, as $x \pm s$, and the two s are not the same number. For a difference, s is the spread of the per-seed differences. For a single quantity, it is that quantity's spread across seeds. Two models trained on the same seed move together, and taking the difference seed by seed removes that shared movement, so a column's spread is the wrong number to judge a difference against. A difference quoted with no $\pm$ is not paired. With three seeds this rule describes what the runs did rather than testing significance, so we report how many comparisons come out inconsistent alongside those that do not.

D.2 Model Configurations

Circuit architecture. Every circuit is an Einsum Network (EiNet) (Peharz et al., 2020), a vectorized smooth and decomposable PC that computes each layer with one einsum, from the `simple-einet` implementation (Braun, 2021). We use its `einsum` sum layer rather than its default: the einsum layer weights every ordered pair of the two child regions' channels, while the default weights one matched product per output channel. The class variable enters only at the root layer here, so one pass returns all C scores. Theorem 1 and Algorithm 1 assume no such layout. We check smoothness and decomposability on the trained weights of 228 circuits, covering 114 of the 114 cells this paper reports. Each attribute group lies in exactly one leaf region per repetition, and the leaf regions halve to a single root.

An EiNet follows the RAT-SPN template (Peharz et al., 2019) with depth D, N_S sum units per layer, R repetitions, and a top class layer of C outputs mixed across repetitions. For $D \geq 2$ its

size is $|\mathcal{S}| = O(KN_LmR + RN_S^3 2^D + RN_S^2 C)$, with N_L leaves per region and $m = \max_k m_k$, every group's categorical leaf padded to that common width. Because $2^D \leq K$ this is linear in K. At $D = 1$ the class layer is the only sum layer and takes the leaves directly, so the size is $O(KN_LmR + RN_L^2 C)$ and the leaf channel count has to grow in place of N_S. The implementation holds every circuit parameter as a real logit and normalizes it with a softmax, over the input channels at the sum nodes and over the group's own values at the leaves.

Table 2: Neural backbone configuration, parameter counts from the Stage-1 checkpoints. d_{hidden} is the width of the hidden layer before each attribute group, and 0 means one linear map per group.

Dataset	Backbone	d_{feat}	d_{hidden}	Backbone params	Head params
MNIST-Add	ResNet-34	512	128	21.30M	133.9K
GTSRB	ResNet-34	512	128	21.30M	268.2K
AwA2	ResNet-34	512	128	21.30M	266.7K
CelebA	ResNet-34	512	128	21.30M	330.0K
CUB-200	ResNet-50	2048	0	23.56M	459.0K
SUN	ResNet-50	2048	256	23.56M	53.56M

Table 3: Circuit geometry and size, from the Stage-2 checkpoints. D is depth, N_S sum units and N_L leaves per region, and R repetitions. "n/a" marks a setting that does not apply at depth 1. Counts are from seed 42, and all 3 seeds agree.

Dataset	D	N_S	N_L	R	EiNet params
MNIST-Add	1	n/a	80	40	4.93M
GTSRB	2	40	40	20	4.02M
AwA2	2	5	8	60	133.7K
CelebA	2	8	6	20	178.5K
CUB-200	5	8	4	30	693.8K
SUN	5	8	4	20	1.13M

Tables 2 and 3 give the recognizer's and the circuit's configuration per benchmark, with parameter counts measured from the checkpoints.

D.3 Training Configuration

The training stages. The three stages are those of Chen et al. (2025), and Table 4 gives every optimizer setting. Stage 1 trains the attribute head on all six datasets with the same trainer. Stage 2 pretrains the circuit on attribute vectors sampled from the Stage-1 predictions, ten per image. Its generative phase, Stage 2a, keeps the last epoch. Its discriminative phase, Stage 2b, keeps the best epoch on a validation set built the same way with one draw per image, scored every second epoch and on each of the first three. Stage 3 fine-tunes the circuit and the recognizer together against the exact inference. Weight decay is 4×10^{-5} in Stage 1, none in Stage 2 and 10^{-3} in Stage 3. Batch size is 1024 in Stage 1 (256 on CUB-200 and SUN), 256 in Stage 2 and 64 in Stage 3. Stage 2b uses a learning rate of 0.001 on SUN, and in Stage 3 the circuit and head rates are 2×10^{-3} and 2×10^{-4} on CUB-200 and 10^{-3} and 10^{-4} on SUN, the head rate covering the backbone and the attribute heads together.

Stage 1's loss. Stage 1 averages a per-group loss over the K groups. That loss is a cross-entropy over each group's values, except on AwA2, whose groups carry several values at once and take a binary cross-entropy on every value independently. We divide each cross-entropy by the natural log of its group's cardinality, so a uniform predictor scores 1.0 in every group at any size.

Backbones, transforms and supervision. CUB-200 and SUN use a ResNet-50 (He et al., 2016) from torchvision's `IMAGENET1K_V2` weights under the Koh et al. (2020) pipeline (ColorJitter, RandomResizedCrop and a horizontal flip on train, centre crop on validation and test). MNIST-Add, GTSRB, AwA2 and CelebA use ResNet-34 with a 128-unit head and the plain Resize and ToTensor

Table 4: Training recipe, with the per-dataset rates and batch sizes stated above. Stage 2b and Stage 3 both use a cosine schedule, Stage 3 over its full 150 epochs, and Stage 1 uses ReduceLROnPlateau (factor 0.1, patience 10). "—" marks a rate the stage has no parameters for, since Stage 1 trains the head alone and Stage 2 the circuit alone.

Stage	Optimizer	Circuit LR	Head LR	Epochs	Objective
1	SGD, momentum 0.9	—	0.01	150	per-group CE or BCE
2a	Adam	0.01	—	50	$\log \tilde{P}_\omega(a, y)$
2b	Adam	0.005	—	200	$\log P_\omega(y \mid a)$
3	AdamW	$5{\times}10^{-5}$	$5{\times}10^{-6}$	150	$-\log h^{\text{gen}}(y \mid x)$

path. SUN alone keeps the whole frame at train and test, because its attributes describe the scene and not a centred object. SUN puts a 256-unit hidden layer between the backbone and each group, CUB-200 one linear map per group. CUB-200 and CelebA supervise the head on a per-class summary of the annotations, the usual concept-bottleneck practice, and MNIST-Add, GTSRB, AwA2 and SUN on each image's own row. The two coincide on GTSRB and AwA2, which publish one attribute row per class. Stage 2b and Stage 3 clip gradients to norm 5.0, in Stage 3 over the circuit and recognizer parameters as one vector. Between the two phases of Stage 2 the trainer clamps every leaf and sum logit into $[-20, 20]$, replacing any non-finite logit first. At least one of them acted, so the clamp changes the parameters the discriminative phase starts from.

D.3.1 The Stage-3 objective

The class term is cross-entropy on the exact generative inference,

$$\mathcal{L}(x, y) \;=\; -\log h^{\text{gen}}(y \mid x) \;=\; -\log \operatorname{softmax}_y\big(\log s(x)\big),$$

with the C log scores $\log s(x)$ of Eq. (3) from a single soft-evidence pass, and model selection on the validation accuracy of that same inference. There is no early stopping, and Stage 2 and Stage 3 select the epoch they keep on validation accuracy alone. The full objective adds the attribute loss,

$$\mathcal{L} = -\log h^{\text{gen}}(y \mid x) \;+\; \lambda_c \,\frac{1}{K}\sum_{k=1}^{K} \mathrm{CE}\big(\boldsymbol{\pi}^{(k)}(x),\, t_k\big), \tag{18}$$

with t_k the attribute target for group k. The reported models weight the attribute loss at $\lambda_c = 1$, and Appendix E reports what the default weight does instead.

The softmax parameterization has two consequences. First, before any per-class scale and over the padded product the implementation sums, every class mass equals one, $\sum_a \tilde{P}_\omega(a, Y{=}y) = 1$, because normalized sum weights and normalized leaves multiply and sum to one. The parameterization forces this, so the ratio of 1.000002 that the classes agree to is a check on our arithmetic and says nothing about the trained circuit. Second, equal class masses still allow a non-uniform induced class prior, so they do not prevent $\mathcal{E}_C^{\mathrm{TV}}$ from vanishing. Take any strictly positive $P_D(Y \mid a)$, and set $\tilde{P}_\omega(a, Y{=}y) = P_D(y \mid a)\,\sigma(a)$ for a positive σ solving the C normalizations $\sum_a P_D(y \mid a)\sigma(a) = 1$. Then $P_\omega(Y \mid a) = P_D(Y \mid a)$ everywhere and $\mathcal{E}_C^{\mathrm{TV}}(x) = 0$. Two assignments with $C = 2$, $P_D(1 \mid a_1) = 0.9$ and $P_D(1 \mid a_2) = 0.2$ already give an example. With $\sigma = (6/7, 8/7)$ both class masses sum to one, and the true class prior is $(0.55, 0.45)$ at $P_D(a) = (0.5, 0.5)$. The parameterization does remove the per-class constant.

Where this differs from Chen et al. (2025). They optimize the discriminative inference of Eq. (2) exactly, enumerating all $\prod_k m_k$ assignments, which their attribute counts permit. Section 6.3 runs that objective where it is feasible and a sampled estimator of it where it is not. The remaining differences are with their released artifact rather than with their paper. Their appendix and their trainer freeze the circuit while the recognizer updates, every Stage 3 run here updates both, and their main text describes the stage as updating both. Their Stage 2 learns the circuit's structure with LearnSPN and fits its parameters by maximum likelihood, on one attribute row per training instance taken from the annotation, and a group annotated with several values is replaced by one value drawn uniformly from that group, which affects all four AwA2 groups and three of the five CelebA groups.

Their paper describes drawing each attribute from the annotation's own probability vector. Ours trains a fixed region graph by gradient descent on ten vectors per image drawn from the Stage-1 recognizer, in the generative and then the discriminative phase of Section 4.4.

Test-time evaluation. We run all three inference procedures on the same Stage-3 checkpoint, at batch size sixty-four on the four small-K benchmarks and thirty-two on CUB-200 and SUN. The generative and enumerated inference procedures are deterministic. The sampled one draws per batch from one seeded stream, so we report its spread across Monte-Carlo seeds, and on the four small-K benchmarks it floors each group's distribution at 10^{-8} before drawing, as a guard on the multinomial, with no floor on CUB-200 and SUN. The generative pass has a floor of its own. It pads the evidence tensor to a common width $m = \max_k m_k$ so the leaf sum is one tensor operation, and clamps the tensor at 10^{-45} before the logarithm, which lifts a padded bin and an underflowed real probability alike. Our check covers the padded bins. Where the enumeration runs, it and the pass agree to 1.91×10^{-5} in log space. It does not cover the underflow, because the enumeration applies the same clamp at real bins, and we make no claim about it.

The circuit-only times are a separate measurement at batch size sixty-four over head outputs cached once (Section 6.1), so they exclude image loading and the backbone. That shared cost is larger than the generative pass itself on every benchmark, by $28\times$ to $220\times$ and is smaller than the sampling loop on every benchmark but AwA2. It is the same number on both sides of every ratio, so dropping it moves each ratio away from one. The sampling ratios come out larger than a whole-pipeline comparison would give, and the enumeration ratios come out smaller. The inference procedures make different numbers of circuit calls per batch, one for the generative inference, n for Monte Carlo, and one per split plus a contraction per batch for enumeration, so each timing belongs to the batch size we state for it.

Table 5: Circuit-only seconds over the full test split, mean $\pm$ s.d. over 3 training seeds, the numbers behind Figure 1. *infeasible* marks an attribute space too large to enumerate. The last column is the shared head and loading cost, excluded from the others.

Dataset	K	$\prod_k m_k$	Disc-Enum	Generative	Disc-MC	Shared head
MNIST-Add	2	10^2	0.020 ± 0.001	0.064 ± 0.002	8.34 ± 0.14	1.78 ± 0.04
GTSRB	4	3,120	0.040 ± 0.004	0.056 ± 0.003	8.79 ± 0.31	2.74 ± 0.18
AwA2	4	3,402	0.034 ± 0.003	0.052 ± 0.003	8.22 ± 0.12	11.46 ± 1.75
CelebA	5	108	0.092 ± 0.003	0.255 ± 0.002	50.21 ± 0.29	16.72 ± 0.51
CUB-200	112	2^{112}	*infeasible*	0.240 ± 0.023	121.91 ± 9.16	10.50 ± 0.29
SUN	102	2^{102}	*infeasible*	0.041 ± 0.003	19.63 ± 0.63	2.46 ± 0.49

D.3.2 Cost of enumeration where it runs

The circuit's class conditional $P_\omega(Y \mid \mathbf{a})$ does not depend on the input, so enumeration evaluates the circuit once per assignment for the whole test split, and each input then costs one contraction over $\prod_k m_k$ columns. Among the benchmarks where it runs at all, $C \prod_k m_k$ is below $|\mathcal{S}|$ on MNIST-Add, GTSRB and CelebA and above it on AwA2. At small K enumeration is therefore not slower than the single pass, and it stops being faster where that contraction overtakes the circuit. It takes 0.31 to 0.72 times the generative pass on those four benchmarks, which is less on all of them. Both timers run each procedure once without timing it, so neither includes initialization. The sweep of Appendix D.3.3 times complete inference passes instead, at batch size 16, because it asks what each procedure costs a user at equal accuracy. Counting the sampling work alone, its parity budget costs $3.6\times$ on CelebA. The single pass costs the same at any $\prod_k m_k$, so it still runs at 2^{112} or 2^{102}.

Monte-Carlo sample count. By Hoeffding with a union bound over C classes, $n \geq \log(2C/\delta)/(2\varepsilon^2)$ draws suffice to estimate every class probability at a single input to within ε with probability $\geq 1 - \delta$. The condition is sufficient, so it sets a budget rather than showing a smaller one fails. At $\varepsilon = 0.20$ and $\delta = 0.05$ it asks for between eighty-three and 129 draws across our six datasets, and we run the sampled inference at $\max(100, n)$. The floor binds on AwA2, GTSRB and MNIST-Add, and the other three run at their own requirement (107 on CelebA, 113 on CUB-200 and 129 on SUN), meeting the target on six of six benchmarks. Every benchmark meets that target. No

benchmark exceeds its requirement by more than seventeen draws, so the reported time ratios hold at the budget this baseline's own guarantee prescribes. We cannot afford a tighter target. At $\varepsilon = 0.01$ the same bound asks for 332 to 399 times that budget.

D.3.3 THE DRAW-COUNT SWEEP

Figure 2 varies n after training and reports the smallest budget at which sampling matches the generative inference. MNIST-Add and GTSRB are above 99.1% already at one sample, so they are uninformative here. Sampling reaches parity on CelebA at n=25 (within 0.09 points). AwA2, CUB-200 and SUN never reach parity. AwA2 is 7.93 points behind at n=50, CUB-200 is 2.02 points behind at n=100 and SUN is 19.32 points behind at n=500. Both sides of that ratio are complete inference passes. The figure's legend orders the heads by $\bar{\pi}_{\max} = \frac{1}{NK}\sum_i \sum_k \max_v \pi_k(v \mid x_i)$, the mean over test images and groups of the largest probability the Stage-3 attribute head puts on any value in a group. The draws estimate an average over the attribute posterior, so the budget follows how concentrated that posterior is rather than how many assignments it spans. Ordering the six benchmarks by $\bar{\pi}_{\max}$ also orders their single-draw penalty, from 0.02 points on GTSRB ($\bar{\pi}_{\max}$=1.000) to 26.12 on AwA2 (0.679). The 3 pairs that run the other way are CUB-200 (0.960, -21.25) against CelebA (0.886, -5.22), SUN (0.932, -27.79) against CelebA (0.886, -5.22) and SUN (0.932, -27.79) against AwA2 (0.679, -26.12). The size of the assignment space does not set that budget. GTSRB has $3{,}120$ assignments and reaches parity at n=1, while CelebA has 108 and reaches parity at n=25.

Baselines. We train the DNN and CBM columns of Table 1 ourselves, at Chen et al. (2025)'s hyperparameters, with a resize to 224×224 followed by ToTensor with no normalization and no augmentation. Their class loss fails at 200 classes, because it trains C independent logistic regressions and selects on a validation subset accuracy that is 0 at every epoch. We therefore also train the same architecture with softmax cross-entropy and select on validation top-1. On CUB-200 and SUN we train both baselines again at our own ResNet-50 under Koh et al. (2020)'s preprocessing, which is ColorJitter, RandomResizedCrop and a horizontal flip on train, and a centre crop on validation and test, at 299 pixels, with normalization. We use their validation and test transform exactly, a centre crop of the raw image with no resize before it, and their normalizer, a mean of 0.5 and a standard deviation of 2 in every channel, not the ImageNet statistics. On CUB-200 we supervise our recognizer on Koh et al.'s published per-image attribute labels and the baselines on the shipped column. Those two label sets share 106 of 112 attributes and binarize the annotator responses by different rules.

E STAGE-3 OBJECTIVE AND THE CONCEPT LAYER

Two ways to compare the table. The first comparison takes each checkpoint with the inference its own objective targets, which is the diagonal of Table 6. The bold cells there show where an objective's own inference is among the best on its row, which is what makes that diagonal the right cell to report. Where all three run they agree to within 2.29 points. On CUB-200 and SUN enumeration does not fit, so we compare only the other two. The second comparison takes every checkpoint with the generative inference, the one we deploy. Of the differences larger than the spread across seeds, six of six favor the exact objective, and none favors another objective by more than the spread across seeds. A model trained against the exact objective therefore stays accurate under either procedure on two of the five benchmarks where the inference matters, and the ordering reverses on CUB-200, MNIST-Add and SUN. With the model fixed at our objective's checkpoint and only the inference procedure varying, the generative one is ahead on three ($+18.09 \pm 1.62$ points on SUN, $+5.04 \pm 3.24$ on AwA2 and $+2.57 \pm 0.69$ on CUB-200) and behind on one. On the four benchmarks where no run uses the learned loss scale described below, the three runs differ only in the objective and in the criterion we select on. On CUB-200 and SUN the two runs lie on opposite sides of that scale, so there the comparison moves two things at once.

Section 4.4 defines all three objectives, and this paragraph gives what each one costs. Ours costs one soft-evidence pass, $O(|\mathcal{S}|)$ at any K. The enumerated objective costs $O(\prod_k m_k \cdot |\mathcal{S}|)$ per optimizer step, since it evaluates the circuit at every assignment and ω moves each step, so it cannot amortize the table the way inference does. By the delta method the bias of $\mathcal{L}^{\text{samp}}$ falls as $O(1/n)$ wherever $\mathbb{E}_{\boldsymbol{\pi}}[P_\omega(y \mid A)]$ stays away from zero.

The loss scale. On CUB-200 and SUN the class loss is taken at a learned positive scale. The cross-entropy receives e^t times the circuit's class log scores, for one scalar t trained alongside the circuit from 0, so the objectives as displayed are the $t = 0$ case. Stage 2b's cross-entropy takes it too. Of the Stage-3 objectives the base trainer implements, the enumerated one, the exact generative one and the single-draw control carry the scale and the sampled conditional one does not. A global positive scale cannot move an $\arg\max$, so it changes no reported accuracy, and we state it because the objective we display is then not the one the trainer minimizes. The other four benchmarks carry no such scale. Separately, the reported CUB-200 configuration carries a learned class prior, rescaling each class output by a learned scalar, so its class masses differ and its circuit is unnormalized, the case Remark 1 covers.

Estimator. $\boldsymbol{\pi}$ enters $\mathcal{L}^{\text{samp}}$ only through the measure the draws come from, so sampling from $\boldsymbol{\pi}$ directly would give the recognizer no gradient. We sample from the detached $\bar{\boldsymbol{\pi}}$ and reweight, at ten draws, which puts $\boldsymbol{\pi}$ back into the expression. The label normalization contributes a baseline term that centres the gradient, so we need no separate variance reduction. The trainer also implements a single-draw control, which takes one draw from $\boldsymbol{\pi}$ and never reweights it. Its pathwise gradient is identically zero, so no class gradient reaches the recognizer, and we do not report it beside the three objectives: a comparison against it would move the objective and the module that receives gradient at the same time.

Comparing the objectives. Under the generative inference the difference between two objectives is larger than the spread across seeds on six of the fourteen pairwise comparisons (on AwA2, CelebA, CUB-200 and SUN), the largest being the sampled objective against the exact objective on SUN, -19.92 ± 1.20 points. Of those differences, none go against the exact objective.

Concept weight. A class loss does not require the attribute layer to stay faithful to the annotation. At λ_c=0.005, the weight at which CUB-200 and SUN were trained before this ablation, mean per-attribute accuracy is 77.0% and 80.8% against 95.3% and 96.0% after Stage 1. At λ_c=1 it is 96.0% and 94.2%. Holding the layer costs class accuracy on AwA2 and SUN, by -2.32 points and -4.05 points. On CelebA and CUB-200 the weight holds more of the layer while class accuracy changes by less than the spread across seeds, so the smaller weight costs attribute accuracy without a measurable gain in class accuracy. We use λ_c=1 on every benchmark.

Effect of each objective on the attribute layer. The objectives differ in how much class gradient reaches the recognizer, and the attribute layer tracks that difference. The sampled objective erodes it (5.1 points on CUB-200, 8.4 on SUN), ours erodes it further (18.3 and 15.1), and the one-draw control does not erode it at all ($+2.7$ and -0.1 points). The one-draw control and the default-weight configuration were trained at λ_c=0 on the four small-K datasets and λ_c=0.005 on CUB-200 and SUN. At those weights the layer loses more than 1.0 point on AwA2, CelebA, CUB-200 and SUN, up to 18.3 points, and holds to within 0.0 points on MNIST-Add and GTSRB. Setting λ_c=1 keeps more of the layer than the default everywhere the default erodes it, and does not cost class accuracy. The cost of doing so differs by objective. On SUN the weight costs the exact objective 4.05 points and moves the sampled objective by $+0.98$.

Selection and inference agreement. Table 6 reports, for every setting, the accuracy each inference procedure gives on that setting's own checkpoint and the attribute layer it leaves behind. That last column is the checkpoint's own head scored against the Stage-1 value printed under the dataset name, and bold marks a drop of more than five points. All settings share byte-identical Stage-1 heads and start Stage 3 from the same Stage-2 circuit, and each checkpoint is the epoch that scored best under the quantity that setting optimizes. The one-draw control is the exception. We select it on a ten-sample average of log joint scores, which is neither the objective it trains nor the column matched to it. Each objective is most accurate under the quantity it optimizes. The generative and sampled inference procedures agree on a checkpoint only while its attribute layer is intact. On CUB-200 they differ by 75.68 points at the default λ_c, where joint training moved the attribute layer by 18.3 points, against 2.57 at λ_c=1, and on SUN by 36.08 against 18.09. A checkpoint trained on the exact objective loses at most 18.09 points when evaluated a way it was not trained for, against up to 10.47 points for the others. The ordering reverses on CUB-200, MNIST-Add and SUN. There the steadiest of the other objectives loses 1.65 points on CUB-200, 0.00 points on MNIST-Add and 1.45 points on

Table 6: Stage-3 objectives, all three at λ_c=1, each row on its own Stage-3 checkpoint, mean ± s.d. over 3 training seeds. Bold marks every class accuracy in a row that no other beats by the rule of Appendix D.1, and a row with no bold is one where none of the three separates. † marks a configuration whose attribute accuracy ends more than five points below its Stage-1 value. "—" and *infeasible* mark attribute spaces too large to enumerate.

		Class accuracy			Attribute
Dataset	Stage-3 objective	Gen	Disc-Enum	Disc-MC	accuracy
MNIST-Add	exact (ours)	98.98 ± 0.12	98.95 ± 0.07	98.95 ± 0.09	99.5 ± 0.0
(Stage 1: 99.4)	enumerated (Chen et al., 2025)	98.92 ± 0.03	98.91 ± 0.05	98.91 ± 0.04	99.5 ± 0.0
	sampled	**98.91 ± 0.16**	98.83 ± 0.23	**98.84 ± 0.22**	99.4 ± 0.1
GTSRB	exact (ours)	99.93 ± 0.04	99.93 ± 0.04	99.94 ± 0.04	100.0 ± 0.0
(Stage 1: 99.9)	enumerated (Chen et al., 2025)	99.93 ± 0.05	99.93 ± 0.05	99.94 ± 0.04	100.0 ± 0.0
	sampled	99.91 ± 0.01	99.91 ± 0.01	99.92 ± 0.01	100.0 ± 0.0
AwA2	exact (ours)	**87.15 ± 1.76**	82.11 ± 4.44	81.99 ± 4.13	97.9 ± 0.0
(Stage 1: 98.7)	enumerated (Chen et al., 2025)	79.09 ± 0.23	**89.44 ± 0.22**	**89.42 ± 0.18**	97.5 ± 0.1
	sampled	78.93 ± 1.11	**89.38 ± 0.27**	**89.40 ± 0.19**	97.6 ± 0.1
CelebA	exact (ours)	34.57 ± 0.10	**34.77 ± 0.13**	34.67 ± 0.09	80.4 ± 0.3
(Stage 1: 81.2)	enumerated (Chen et al., 2025)	34.03 ± 0.17	**34.93 ± 0.19**	**34.87 ± 0.15**	80.7 ± 0.0
	sampled	34.15 ± 0.28	**34.90 ± 0.03**	34.85 ± 0.06	80.8 ± 0.1
CUB-200	exact (ours)	**79.18 ± 1.12**	—	76.60 ± 1.80	96.0 ± 0.4
(Stage 1: 95.3)	enumerated (Chen et al., 2025)		*infeasible*		
	sampled	67.52 ± 1.67	—	**69.17 ± 1.93**	96.8 ± 0.0
SUN	exact (ours)	**32.17 ± 2.03**	—	14.08 ± 1.24	94.2 ± 0.4
(Stage 1: 96.0)	enumerated (Chen et al., 2025)		*infeasible*		
	sampled	12.25 ± 2.66	—	**13.70 ± 2.64**	94.2 ± 0.4

SUN, against ours at 2.57, 0.03 and 18.09. A checkpoint trained against the inference we deploy can therefore be evaluated either way. A checkpoint trained against a discriminative inference cannot.