# CASPER: A Projection-Based Neurosymbolic Layer for Scalable & Guaranteed Constraint Satisfaction

**Lohith Konathala**
Imperial College London
l.konathala25@imperial.ac.uk

**Luca Andolfi**
Imperial College London
l.andolfi24@imperial.ac.uk

**Eleonora Giunchiglia**
Imperial College London
e.giunchiglia@imperial.ac.uk

## Abstract

Many real-world problems demand exact constraint satisfaction to be meaningfully solved, yet existing neurosymbolic methods face a stark trade-off: those that guarantee satisfaction fail to scale, while those that scale must resort to approximation. Meanwhile, traditional deep learning approaches offer no satisfaction guarantees at all. In this paper, we propose CASPER, a neurosymbolic layer that simultaneously (i) guarantees constraint satisfaction, (ii) scales at inference time, (iii) is fully automated, requiring no problem-specific engineering, and (iv) is fully differentiable, making it composable with any neural predictor both at training and at inference time. CASPER casts constraint satisfaction as a closed-form Euclidean projection onto the constraint-feasible region, computable in a single forward pass. Experiments across MNIST arithmetic (up to 1024 digits), Sudoku solving, and Warcraft pathfinding show that CASPER is the only method that produces guaranteed-valid outputs at scale: on 30×30 Warcraft grids, exact baselines time out and approximate ones produce valid paths less than 7% of the time, while CASPER produces them 100% of the time.

## 1 Introduction

Many structured prediction problems require outputs that are guaranteed to satisfy logical constraints. A clinical decision support system recommending combinations of medications must never prescribe pairs with known contraindications; a hierarchical multi-label classifier must respect that any predicted sub-class implies its parent. In such settings, a single constraint violation invalidates the entire prediction, regardless of how accurate the underlying predictor is on average. Standard neural architectures provide no guarantee that individual predictions satisfy the constraints at inference time.

Neurosymbolic learning addresses this gap by combining neural predictors with symbolic representations of background knowledge, but existing methods exhibit a stark trade-off between guarantee and scalability. Methods that enforce constraints by construction — Semantic Probabilistic Layers [3] and CCN+ [25] — either compile the constraint into a circuit and renormalize the predictor against it for every input (SPL), or encode each $h$-valued categorical variable as $h$ binary indicators (CCN+). Neither approach scales to the structured-prediction regimes our experiments target. Methods that scale—Scallop [30], DeepSoftLog [35], A-NeSI [48], and CTSketch [10]—replace exact symbolic inference with approximations such as top-$k$ proof enumeration, neural surrogates for weighted model counting, or low-rank tensor sketches, but in doing so forfeit the satisfaction guarantee.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Table 1: Comparison of CASPER against baselines on three properties. *Guaranteed Satisfaction* (Sat): method provably produces constraint-compliant outputs. *Scalable* (Scale): method manages to scale over most over the tasks in our experimental analysis, *Automatic* (Auto): no problem-specific engineering required.

| Method | SPL [3] | CCN+ [25] | DSL [35] | Scallop [30] | A-NeSI [48] | CTSketch [10] | CASPER |
|---|---|---|---|---|---|---|---|
| Sat | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Scale | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Auto | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |

We present CASPER (**C**ategorical **A**rgmax **S**atisfaction via **P**roj**E**ction and **R**esolution) [1], a differentiable neurosymbolic layer that enforces logical constraint satisfaction at scale. Given a neural predictor $m_\theta$ and a set of constraints $\Pi$ over categorical variables, CASPER composes with $m_\theta$ as $\tilde{m}_\theta = \text{CASPER} \circ m_\theta$, and processes the variables in sequence projecting each unconstrained output onto the constraint-feasible region of the probability simplex. The variable assignments induced by the argmax of $\tilde{m}_\theta$ are guaranteed to satisfy $\Pi$ by construction. CASPER achieves scalability through two ideas: an offline compilation step that uses the signed resolution rule to reduce multi-variable satisfaction to a sequence of single-variable problems, and a closed-form surrogate of the per-variable projection that requires quadratically fewer sequential steps on a GPU than the exact solution while preserving the satisfaction guarantee. To further exploit GPU parallelism, CASPER parallelizes the computation over different variables whenever possible using the same intuition as in [25], making the fully sequential case the worst-case bound. Table 1 summarises the contrast: CASPER is the only method that simultaneously (i) guarantees constraint satisfaction, (ii) scales at inference time, and (iii) requires no problem-specific engineering beyond the constraint specification.

Our experiments confirm these claims across three benchmarks. On Warcraft pathfinding, CASPER is the only method that produces guaranteed valid paths on the $30 \times 30$ grid, where exact baselines time out and approximate methods produce invalid predictions. On MNIST-Arithmetic, CASPER is the only exact-satisfaction method that scales to sums over more than a thousand digits, with inference time that closes the gap to approximate methods as the problem grows. On Sudoku solving, CASPER reaches $66\%$ post-layer board accuracy, against $25\%$ for the strongest scalable baseline unconstrained.

## 2 Problem Definition

**Notation.** We denote scalar constants by lower-case letters, e.g., $x$, random variables by upper-case letters, e.g., $X$, and ordered sets of constants and random variables by bold letters, e.g., $\mathbf{x}$ and $\mathbf{X}$, respectively. We write $[n]$ for the set $\{1, \ldots, n\}$.

**Setting.** We consider a learning setting with input space $\mathcal{X}$ and structured output space $\mathcal{Y} = [h_1] \times [h_2] \times \cdots \times [h_n]$, i.e., the structured output is given by $n$ categorical variables $\mathbf{Y} = (Y_1, \ldots, Y_n)$, where each $Y_i$ takes values in $[h_i]$. Let $\Pi$ be a set of constraints defining the *feasible assignments* $\mathcal{Y}_\Pi = \{\mathbf{a} \in \mathcal{Y} \mid \mathbf{a} \models \Pi\}$ of the problem, $\mathbf{a} \models \Pi$ indicating that the assignment $\mathbf{a} \in \mathcal{Y}$ satisfies $\Pi$. For ease of exposition, we represent background knowledge using signed clauses with empty order [9]. A *signed clause* is an expression of the form:

$$\mathcal{S}_1 : Y_1 \vee \mathcal{S}_2 : Y_2 \vee \ldots \vee \mathcal{S}_m : Y_m, \tag{1}$$

where, with a slight abuse of notation, $Y_i$ indicates a multiple-valued logic variable (corresponding to the respective random variable), $\mathcal{S}_i \subseteq [h_i]$ represents a set of truth values, and $\mathcal{S}_i : Y_i$ is a *signed literal*. Without loss of generality, from now on we assume that every variable appears at most once in every clause, as any clause of form $\mathcal{S}_1 : Y_i \vee \ldots \vee \mathcal{S}_k : Y_i \vee \varphi$ can always be rewritten as $(\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_k) : Y_i \vee \varphi$. An assignment $\mathbf{a}$ *satisfies* the signed clause in Equation 1 iff for some $i = 1, \ldots, m$ it assigns a value in $\mathcal{S}_i$ to $Y_i$. An assignment $\mathbf{a}$ *satisfies* a conjunction (or a set) of signed clauses if it satisfies all of them. Notice that this is purely a presentational choice: in our setting, signed clauses are in one-to-one correspondence with propositional logic formulas [8]. We adopt this

[1] Code is available at `https://github.com/lkonathala25/casper`.

representation only because it yields cleaner notation and simplifies the theoretical development in the remainder of the paper.

**Example 1.** *Suppose we have an image classification task where we have to decide (i) if the shape shown in the image is a square, circle or triangle, (ii) if it is yellow, red or blue and (iii) if the shape is big (↑) or small (↓). Suppose also we know that shapes that are either big or yellow are always circles. In this problem, our structured output space will be:*

$$\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \mathcal{Y}_3 \qquad \mathcal{Y}_1 = \{\textcolor{yellow}{\blacksquare}, \textcolor{red}{\blacksquare}, \textcolor{blue}{\blacksquare}\} \qquad \mathcal{Y}_2 = \{\bigcirc, \square, \triangle\} \qquad \mathcal{Y}_3 = \{\downarrow, \uparrow\}$$

*and our knowledge can be expressed using two signed clauses:*

$$\{\bigcirc\} : Y_2 \vee \{\downarrow\} : Y_3 \qquad\qquad \{\bigcirc\} : Y_2 \vee \{\textcolor{red}{\blacksquare}, \textcolor{blue}{\blacksquare}\} : Y_1$$

We study models of the form $m_\theta : \mathcal{X} \to \prod_{i=1}^{n} \Delta([h_i])$, where $\Delta([h_i])$ denotes the probability simplex over $[h_i]$. For any input $\mathbf{x} \in \mathcal{X}$, we write $m_\theta(\mathbf{x}) = \big(m_\theta^1(\mathbf{x}), \ldots, m_\theta^n(\mathbf{x})\big)$, where $m_\theta^i(\mathbf{x}) \in \Delta([h_i])$ is the predicted categorical distribution for the $i$-th output variable, and we say that $\mathbf{a} \in \mathcal{Y}$ is the assignment induced by the output $m_\theta(\mathbf{x})$ if for every $i = 1, \ldots n$, $a_i = \operatorname{argmax}_{j \in [h_i]} m_\theta^i(\mathbf{x})_j$.[2] We say that $m_\theta$ *is compliant with* $\Pi$ if, for every $\mathbf{x} \in \mathcal{X}$, the assignment $\mathbf{a} \in \mathcal{Y}$ induced by $m_\theta(\mathbf{x})$ satisfies $\Pi$.

**Goal.** Given any network $m_\theta : \mathcal{X} \to \prod_{i=1}^{n} \Delta([h_i])$ and constraints $\Pi$, our goal is to design a layer (i) that is guaranteed to be compliant with $\Pi$, (ii) that is scalable, and (iii) requires no problem-specific engineering beyond the constraint specification.

## 3 CASPER

Since CASPER reduces any multi-variable problem to a sequence of single-variable ones via an offline compilation step, we present the single-variable case first and then the general case.

### 3.1 Single Variable Case

Let $\mathcal{Y} = [h]$ and suppose that background knowledge trivially expresses that, for every input $\mathbf{x}$, the assignment $\mathbf{a} \in [h]$ induced by the output $m_\theta(\mathbf{x})$ must always belong to the subset $\mathcal{S} \subseteq [h]$, i.e., $\Pi = \{\mathcal{S} : Y\}$.

Given a model $m_\theta : \mathcal{X} \to \Delta([h])$ and an input $\mathbf{x}$, let $\hat{\mathbf{y}} = m_\theta(\mathbf{x})$. To ensure that the final predictor $\tilde{m}_\theta = \text{CASPER} \circ m_\theta$ is compliant with $\Pi$, CASPER must map $\hat{\mathbf{y}}$ to the closest point in the simplex whose argmax lies in $\mathcal{S}$. We measure closeness in squared Euclidean distance: as we show in the rest of this section, this choice yields a closed-form fixed-point characterization of the projection (Equation (4)), an exact decomposition of its surrogate error (Theorem 4), and a principled gradient analysis (Proposition 1). Concretely, we would like to solve:

$$\operatorname{argmin}_{\mathbf{z} \in \Delta([h])} \|\mathbf{z} - \hat{\mathbf{y}}\|_2^2 \qquad \text{subject to } \operatorname*{argmax}_{j \in [h]} \mathbf{z} \in \mathcal{S}. \tag{2}$$

If $|\mathcal{S}| = 1$, then the optimization problem is convex, while if $|\mathcal{S}| > 1$ the feasible region can be decomposed in $|\mathcal{S}|$ convex regions, each corresponding to a candidate maximizer $t \in \mathcal{S}$. To ensure the feasible set is closed and the projection is well-defined, we find a $\epsilon$-*margin solution* to the problem, which for every $t \in \mathcal{S}$, ensures $z_t \geq z_j + \epsilon, \forall j \in [h] \setminus t$. The $\epsilon$-margin solution for each of the convex problems is characterized by the fixed-point system described in Equation (4).

**Theorem 1** (Fixed point solution)**.** *Let $\hat{\mathbf{y}} \in \Delta([h])$ and $\mathcal{S} \subseteq [h]$, the $\epsilon$-margin solution to the optimization problem in Equation* (2) *is equal to $\mathbf{z}^{t^\star}$ with $t^\star$ equal to:*

$$t^\star = \operatorname*{argmin}_{t \in \mathcal{S}} \|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2, \tag{3}$$

*where for every $t \in \mathcal{S}$, $\mathbf{z}^t$ is the solution of the convex minimization problem obtained by fixing $t$ as the unique maximizer and $\epsilon > 0$ as an arbitrarily small constant:*

$$z_j^t = \begin{cases} \hat{y}_j & \text{if } j \notin (\mathcal{M}^t \cup \{t\}) \\ z_t^t - \epsilon & \text{if } j \in \mathcal{M}^t \\ \frac{\hat{y}_t + \sum_{j \in \mathcal{M}^t} \hat{y}_j + |\mathcal{M}^t|\epsilon}{1 + |\mathcal{M}^t|} & \text{if } j = t \end{cases} \qquad \mathcal{M}^t = \{j \in [h] \setminus \{t\} : \hat{y}_j \geq z_t^t - \epsilon\} \tag{4}$$

[2] In presence of multiple identical maximal (resp. minimal) values, we assume that argmax (resp. (argmin)) deterministically returns a single index, with both operators following the same fixed tie-breaking policy.

The proof can be found in Appendix B. The solution to the fixed-point system (4) can be computed in $\mathcal{O}(h \log h)$ sequential operations and in $\mathcal{O}(\log^2 h)$ GPU operations via parallel reductions (see bottom of Appendix B). To maximise the scalability of our approach, we work with an over-approximation of $\mathcal{M}^t$ that is available in closed form directly from $\hat{\mathbf{y}}$. We define

$$\mathcal{A}^t \;=\; \big\{j \in [h] \setminus \{t\} \;:\; \hat{y}_j \geq \hat{y}_t - \epsilon\big\}, \tag{5}$$

which is computable in $\mathcal{O}(h)$ time and requires a single thresholding operation on a GPU.

**Lemma 1.** *For every $t \in [h]$ and every $\hat{\mathbf{y}} \in \Delta([h])$, $\mathcal{M}^t \subseteq \mathcal{A}^t$.*

Proof can be found in Appendix C. Intuitively, this entails that by replacing $\mathcal{M}^t$ with $\mathcal{A}^t$ in Equation (4) we obtain a *surrogate projection* $\tilde{\mathbf{z}}^t$ in which the $t$th element is guaranteed to be the maximum, but for which we might have over-corrected some other elements.

**Theorem 2** (Constraint satisfaction)**.** *Let $m_\theta : \mathcal{X} \to \Delta([h])$, $\mathbf{x} \in \mathcal{X}$ and $\hat{\mathbf{y}} = m_\theta(\mathbf{x})$. Let $\Pi = \{\mathcal{S} : Y\}$ for some $\emptyset \subset \mathcal{S} \subseteq [h]$. If*

$$\textsc{Casper}(\hat{\mathbf{y}}) = \tilde{\mathbf{z}}^{t^\star}, \qquad t^\star = \operatorname*{argmin}_{t \in \mathcal{S}} \|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\|_2^2, \tag{6}$$

*where for every $t \in \mathcal{S}$*

$$\tilde{z}_j^t = \begin{cases} \hat{y}_j & \text{if } j \notin (\mathcal{A}^t \cup \{t\}) \\ \tilde{z}_t^t - \epsilon & \text{if } j \in \mathcal{A}^t \\ \frac{\hat{y}_t + \sum_{j \in \mathcal{A}^t} \hat{y}_j + |\mathcal{A}^t|\epsilon}{1 + |\mathcal{A}^t|} & \text{if } j = t \end{cases} \qquad \mathcal{A}^t = \{j \in [h] \setminus \{t\} : \hat{y}_j \geq \hat{y}_t - \epsilon\}, \tag{7}$$

*then for $\epsilon \to 0^+$, the assignment $\mathbf{a}$ induced by $\textsc{Casper}(\hat{\mathbf{y}})$ satisfies $\Pi$, i.e., $\mathbf{a} \models \Pi$.*

Proof can be found in Appendix D. Thus, for every $t \in \mathcal{S}$, CASPER uses the surrogate projection $\tilde{\mathbf{z}}^t$ instead of the exact fixed-point solution $\mathbf{z}^t$. The surrogate avoids the sorting step required by the exact solution, reducing the number of sequential GPU operations per candidate from $\mathcal{O}(\log^2 h)$ to $\mathcal{O}(\log h)$. Since candidates are processed in parallel, the full single-variable inference step retains $\mathcal{O}(\log h)$ sequential operations. However, replacing $\mathbf{z}^t$ with $\tilde{\mathbf{z}}^t$ does not change which candidate is selected.

**Theorem 3** (Candidate selection)**.** *Let $\hat{\mathbf{y}} \in \Delta([h])$ and $\emptyset \subset S \subseteq [h]$, and let $\mathbf{z}^t$ and $\tilde{\mathbf{z}}^t$ be defined as in Equations (4) and (7). Then, for $\epsilon \to 0^+$,*

$$\operatorname*{argmin}_{t \in S} \|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2 = \operatorname*{argmin}_{t \in S} \|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\|_2^2 = \operatorname*{argmax}_{t \in S} \hat{y}_t.$$

Proof can be found in Appendix E. Hence the surrogate affects only the continuous output of CASPER, not the value it selects; we now characterise the resulting continuous error exactly.

**Example 2.** *(Cont.'ed Ex. 1) Consider the simplified case where $\Pi = \{\blacksquare\} : Y$ (i.e., we know that the shape is always yellow), and suppose that the network outputs:*

$$m_\theta(\mathbf{x}) = (\blacksquare : 0.3,\ \blacksquare : 0.5,\ \blacksquare : 0.2), \qquad \operatorname{argmax}(m_\theta(\mathbf{x})) = \blacksquare$$

*Let $t = \blacksquare$ and (for simplicity) $\epsilon = 0.01$, then:*

$$\mathcal{A}^{\blacksquare} = \{j \in \{\blacksquare, \blacksquare\} : \hat{y}_{\blacksquare} < \hat{y}_j + \epsilon\} = \{\blacksquare\}$$

$$\textsc{Casper}(\hat{\mathbf{y}}) = \left(\blacksquare : \frac{\hat{y}_{\blacksquare} + \hat{y}_{\blacksquare} + \epsilon}{2}, \blacksquare : \tilde{z}_{\blacksquare} - \epsilon, \blacksquare : \hat{y}_{\blacksquare}\right) = (\blacksquare : 0.405, \blacksquare : 0.395, \blacksquare : 0.2)$$

We now give an exact characterization of the distance between $\tilde{\mathbf{z}}^t$ and $\mathbf{z}^t$. The error decomposes into contributions from each spuriously included index, weighted by its *sub-threshold gap* $g_j = z_t^t - \epsilon - \hat{y}_j$.

**Theorem 4** (Surrogate fidelity)**.** *Let $\mathbf{z}^t$ be the solution of the fixed point system (4), $\tilde{\mathbf{z}}^t$ defined as in Equation (6), and $\mathcal{W}^t = \mathcal{A}^t \setminus \mathcal{M}^t$. For each $j \in \mathcal{W}^t$, let $g_j = z_t^t - \epsilon - \hat{y}_j$. Then*

$$\left\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\right\|_2^2 = \sum_{j \in \mathcal{W}^t} g_j^2 - \frac{1}{1 + |\mathcal{A}^t|}\Big(\sum_{j \in \mathcal{W}^t} g_j\Big)^2. \tag{8}$$

The proof can be found in Appendix F. The exact identity in Equation (8) can be bounded by an $\ell_2$ regularizer on the distance between $\hat{\mathbf{y}}$ and $\mathbf{z}^t$.

**Corollary 1.** *Under the hypotheses of Theorem 4,* $\left\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\right\|_2 \leq \left\|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\right\|_2$.

Corollary 1 motivates adding an $\ell_2$ regularizer $\|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\|_2^2$ to the training objective: driving this quantity to zero drives $\tilde{\mathbf{z}}^t \to \mathbf{z}^t$. Beyond controlling the surrogate fidelity, the $\ell_2$ regularizer serves a complementary role during training: it counteracts the rank deficiency of the layer's Jacobian. This rank deficiency might cause problems during training as it results in a structurally identifiable subspace of upstream gradients being mapped to zero during back-propagation.

**Proposition 1** (Gradient attenuation)**.** *Let* $\hat{\mathbf{y}} \in \Delta([h])$ *be such that* $t^\star$ *and* $\mathcal{A}^{t^\star}$ *in* (6) *are locally constant at* $\hat{\mathbf{y}}$*. Let* $\textsc{Casper}(\hat{\mathbf{y}}) = \tilde{\mathbf{z}}^{t^\star}$ *and* $V = \left\{\mathbf{v} \in \mathbb{R}^h : v_j = v_{t^\star},\ j \in \mathcal{A}^{t^\star}\right\}$*. Then the Jacobian* $J = \partial\tilde{\mathbf{z}}^{t^\star}/\partial\hat{\mathbf{y}}$ *equals the orthogonal projector* $P_V$*. Hence, for any differentiable loss* $\mathcal{L}_{\text{task}}$*,*

$$\nabla_{\hat{\mathbf{y}}}\left[\mathcal{L}_{\text{task}} \circ \tilde{\mathbf{z}}^{t^\star}\right] \;=\; P_V\, \nabla_{\tilde{\mathbf{z}}^{t^\star}} \mathcal{L}_{\text{task}}\,;$$

*the back-propagated gradient is the orthogonal projection onto* $V$ *of* $\nabla_{\tilde{\mathbf{z}}^{t^\star}} \mathcal{L}_{\text{task}}$*, with the* $|\mathcal{A}^{t^\star}|$*-dimensional subspace* $V^\perp$ *mapped to zero by* $J$*.*

The proof can be found in Appendix I. The $\ell_2$ regularizer solves this problem by restoring the gradients along the dimensions that are mapped to zero.

**Proposition 2** (Gradient restoration)**.** *Under the assumptions of Proposition 1, consider the composite training objective*

$$\mathcal{L}_{\text{total}}(\hat{\mathbf{y}}) = \mathcal{L}_{\text{task}}\big(\tilde{\mathbf{z}}^t(\hat{\mathbf{y}})\big) + \tfrac{\lambda}{2}\left\|\tilde{\mathbf{z}}^t(\hat{\mathbf{y}}) - \hat{\mathbf{y}}\right\|_2^2, \qquad \lambda > 0.$$

*Then*

$$\nabla_{\hat{\mathbf{y}}}\mathcal{L}_{\text{total}} \;=\; \underbrace{P_V\, \nabla_{\tilde{\mathbf{z}}}\mathcal{L}_{\text{task}}}_{\in V} \;+\; \underbrace{\lambda\left(\hat{\mathbf{y}} - \tilde{\mathbf{z}}^t\right)}_{\in V^\perp}.$$

The proof can be found in Appendix J. Propositions 1 and 2 together explain why the $\ell_2$ regularizer plays a double role in our framework: it both tightens the approximation bound (Corollary 1) and stabilizes optimization by restoring the gradient along the dimensions originally mapped to zero.

### 3.2 General Case

Extending CASPER to multiple variables requires reasoning about how the assignment of one variable restricts the feasible values of the others. We process variables in a user-specified order $Y_1; \ldots; Y_n$ and, at each step, derive the set of values for $Y_i$ consistent with the constraints under the assignments already fixed for $Y_1, \ldots, Y_{i-1}$. The key technical tool is an inference rule over signed clauses, analogous to propositional resolution, that combines two clauses sharing a literal on the same variable into a single entailed clause. Iterating this rule lets us compile $\Pi$ into a per-variable form that exposes exactly the constraints needed at each step of inference, and reduces multi-variable satisfaction to a sequence of single-variable projections of the kind solved in Section 3.1.

We make use of the *signed resolution* rule, a standard inference rule from the multiple-valued logic literature [26, 9]. Given two signed clauses both containing a term on variable $Y_i$: $(\mathcal{S} : Y_i) \vee \varphi$ and $(\mathcal{S}' : Y_i) \vee \varphi'$, where $\varphi$ and $\varphi'$ denote signed clauses over variables other than $Y_i$, the rule derives a new clause entailed by the premises:

$$\frac{(\mathcal{S} : Y_i) \vee \varphi \qquad (\mathcal{S}' : Y_i) \vee \varphi'}{(\mathcal{S} \cap \mathcal{S}' : Y_i) \vee \varphi \vee \varphi'}. \tag{9}$$

Soundness of the rule is a classical result [26]. Given an user-defined ordering over the variables $Y_1; Y_2; \ldots; Y_n$, we can use the intersection rule to compute the set of constraints necessary to determine the feasible assignment for each variable $Y_i$ knowing the initial assignment given by the model $m_\theta$ and the final assignments for the variables $Y_1, \ldots, Y_{i-1}$. The final set of constraints will then be used to build CASPER. We describe the offline compilation below, and we provide the pseudocode for both the compilation and the inference steps in Algorithm 1.

Starting from the end of the ordering:

- Let $\Pi_n = \Pi$,
- For every $i = n, \dots, 1$, let $\bar{\Pi}_i = \{(\mathcal{S}_j : Y_i) \vee \varphi_j\}_{j=1}^m \subseteq \Pi_i$ be the subset of clauses of $\Pi_i$ where $Y_i$ appears, which allows us to iteratively compute the set $\Pi_{i-1}$ as:

$$\Pi_{i-1} = \left(\Pi_i \setminus \bar{\Pi}_i\right) \cup \text{Confl}_i\left(\bar{\Pi}_i\right), \tag{10}$$

in which $\text{Confl}_i(\bar{\Pi}_i)$ is the *conflict clauses set of* $\bar{\Pi}_i$:

$$\text{Confl}_i(\bar{\Pi}_i) = \left\{ \bigvee_{j \in \mathcal{J}} \varphi_j \mid \bigcap_{j \in \mathcal{J}} \mathcal{S}_j = \emptyset, \emptyset \subset \mathcal{J} \subseteq [m] \right\}. \tag{11}$$

The conflict clauses set represents the set of clauses that need to be added to the layer below the $i$th to avoid ending up in an inconsistent state, i.e., avoiding having $(\emptyset : Y_i) \in \Pi_i$. The conflict clauses set can equivalently be characterized as the closure of $\bar{\Pi}_i$ under the intersection rule restricted to unsatisfiable combinations of signed terms.

**Lemma 2** (Equisatisfiability)**.** *Let $\Pi_i$ be a finite and satisfiable set of signed clauses in the variables $Y_1, \dots, Y_i$. Then $\Pi_i$ and $\Pi_{i-1}$ are equisatisfiable, and each assignment to $Y_1, \dots, Y_{i-1}$ satisfying $\Pi_{i-1}$ can be extended to satisfy $\Pi_i$.*

The proof is given in Appendix K. We now show that the complexity of the COMPILE routine is governed by how many variables a conflict clauses set involves. For a signed clause $\varphi$, let $\texttt{scope}(\varphi) = \{Y \mid \exists \ell = (\mathcal{S}_j : Y_j) \in \varphi.\, Y = Y_j\}$, i.e., the scope of $\varphi$ is the set of variables occurring in some of its literals. We define the *interaction graph* of a set of signed clauses $\Pi$ as the undirected graph $\mathcal{G}_\Pi = (\mathbf{Y}, E_\Pi)$, where $E_\Pi = \{(Y_j, Y_z) \mid \exists \varphi \in \Pi.\, \{Y_j, Y_z\} \subseteq \texttt{scope}(\varphi), j \neq z\}$. Intuitively, the interaction graph of $\Pi$ has an edge between each pair of variables co-occurring in a clause in $\Pi$. Given an ordering $d = Y_1; \dots; Y_n$, the *induced* interaction graph $\mathcal{G}^\star_\Pi(d)$ of $\Pi$ along $d$ connects all the neighbours of a variable $Y_i$ that precede it in $d$, recursively from $Y_n$ to $Y_1$ (cf. [14, Def. 3.7]). Formally, $\mathcal{G}^\star_\Pi(d) = (\mathbf{Y}, E_0)$, where $E_n = E_\Pi$ and for $i = n, \dots, 1$ we have:

$$E_{i-1} = E_i \cup \{(Y_j, Y_z) \mid \{(Y_j, Y_i), (Y_i, Y_z)\} \subseteq E_i,\ j \neq z,\ \textit{and } j, z < i\}. \tag{12}$$

Our bound is controlled by the maximum number of variables preceding some $Y_i$ in $d$ that are adjacent to it in $\mathcal{G}^\star_\Pi(d)$. We call this number the *conflict width*. For $i \in [n]$, the set $\mathcal{N}_d(Y_i) = \{Y_j \mid j < i, (Y_j, Y_i) \in E_0\}$ denotes the neighbours of $Y_i$ in $\mathcal{G}^\star_\Pi(d)$ that precede it in $d$; the conflict width of $\Pi$ along $d$ is $w^\star(d) = \max_{i \in [n]} |\mathcal{N}_d(Y_i)|$.

**Theorem 5** (Computational Complexity of COMPILE)**.** *Let $\Pi$ be a finite set of $r$ signed clauses over the variables $Y_1; \dots; Y_n$, $k = \max_{i \in [n]} h_i$ and $w^\star(d)$ the conflict width of $\Pi$ along the ordering $d = Y_1; \dots; Y_n$. COMPILE$(\Pi, Y_1; \dots; Y_n)$ in Algorithm 1 can be implemented to run in*

$$\text{time } \mathcal{O}((r+n) \cdot k^{w^\star(d)+1}) \qquad \text{and} \qquad \text{space } \mathcal{O}(n \cdot k^{w^\star(d)}).$$

Proof in Appendix L. Theorem 5 shows that the cost of compilation grows exponentially with the conflict width $w^\star(d)$: this is the same worst-case as adaptive consistency and directional resolution [13].

Once the clause sets $\Pi_1, \dots, \Pi_n$ have been computed, CASPER is evaluated sequentially according to the variable ordering $Y_1, \dots, Y_n$ as described below and in the inference step of Algorithm 1. Let $\hat{\mathbf{y}} = m_\theta(\mathbf{x})$. We process the variables in the order $Y_1; \dots; Y_n$. At step $i$, the assignments $\mathbf{a}_{<i} = [a_1, \dots, a_{i-1}]$ for the preceding variables have already been determined by steps $1, \dots, i-1$. These assignments induce a feasible set $\mathcal{S}_i(\mathbf{a}_{<i}) \subseteq [h_i]$ consisting of all values of $Y_i$ consistent with the clauses in $\bar{\Pi}_i$ under $\mathbf{a}_{<i}$. By construction of the compilation procedure, every clause in $\Pi_1$ mentions only $Y_1$, and it can always be reduced to $\Pi_1 = \{\mathcal{S}_1 : Y_1\}$ for some $\mathcal{S}_1 \subseteq [h_1]$. Hence for $i = 1$, $\mathcal{S}_1(\mathbf{a}_{<1}) = \mathcal{S}_1$. Substituting $\mathbf{a}_{<i}$ into $\bar{\Pi}_i$ yields a set of constraints on $Y_i$ equivalent to the single signed clause $(\mathcal{S}_i(\mathbf{a}_{<i}) : Y_i)$. If $\arg\max \hat{\mathbf{y}}_i \in \mathcal{S}_i(\mathbf{a}_{<i})$, the prediction already satisfies the constraints and no correction is needed: $a_i = \arg\max \hat{\mathbf{y}}_i$ and $\tilde{\mathbf{z}}_i^{t^\star} = \hat{\mathbf{y}}_i$. Otherwise, we compute the refined prediction $\tilde{\mathbf{z}}_i^{t^\star}$ for $Y_i$ using the closed-form update of Equation (6) with input $\hat{\mathbf{y}}_i$ and constraints $\{\mathcal{S}_i(\mathbf{a}_{<i}) : Y_i\}$, and set $a_i = \arg\max \tilde{\mathbf{z}}_i^{t^\star}$. Iterating for $i = 1, \dots, n$ yields the final refined output $\text{CASPER}(\hat{\mathbf{y}}) = (\tilde{\mathbf{z}}_1^{t^\star}, \dots, \tilde{\mathbf{z}}_n^{t^\star})$. Notice that, at step $i$, computing $\mathcal{S}_i(\mathbf{a}_{<i})$ from the precomputed $\bar{\Pi}_i$ runs in $\mathcal{O}(\log |\bar{\Pi}_i|)$ sequential GPU operations. Hence, each step requires $\mathcal{O}(\log(|\bar{\Pi}_i| \cdot h_i))$ sequential GPU operations. The output of CASPER satisfies $\Pi$ as proven in Appendix M.

**Algorithm 1** Compilation & Inference Step with CASPER

**function** COMPILE($\Pi, Y_1; \ldots; Y_n$)
  $\Pi_n \leftarrow \Pi$
  **for** $i \leftarrow n$ **down to** 1 **do**
    Compute $\bar{\Pi}_i$, $\text{Confl}_i(\bar{\Pi}_i)$
    $\Pi_{i-1} \leftarrow (\Pi_i \setminus \bar{\Pi}_i) \cup \text{Confl}_i(\bar{\Pi}_i)$
  **end for**
  **if** $\Pi_0$ is unsatisfiable **then**
    **return** UNSAT_FLAG
  **else**
    **return** $\bar{\Pi}_1, \ldots, \bar{\Pi}_n$
  **end if**
**end function**

**function** INFER($\hat{\mathbf{y}}, \bar{\Pi}_1, \ldots, \bar{\Pi}_n$)
  **for** $i \leftarrow 1$ **to** $n$ **do**
    Compute $\mathcal{S}_i(\mathbf{a}_{<i}) \subseteq [h_i]$ from $\bar{\Pi}_i$ under $\mathbf{a}_{<i}$
    **if** $\operatorname{argmax} \hat{\mathbf{y}}_i \in \mathcal{S}_i(\mathbf{a}_{<i})$ **then**
      $\tilde{\mathbf{z}}_i^{t^\star} \leftarrow \hat{\mathbf{y}}_i$
    **else**
      Compute $\tilde{\mathbf{z}}_i^{t^\star}$ using Eq. (6) with input $\hat{\mathbf{y}}_i$ and constraint $\{\mathcal{S}_i(\mathbf{a}_{<i}) : Y_i\}$
    **end if**
    Compute $a_i = \operatorname{argmax} \tilde{\mathbf{z}}_i^{t^\star}$
  **end for**
  **return** CASPER$(\hat{\mathbf{y}}) = \left(\tilde{\mathbf{z}}_1^{t^\star}, \ldots, \tilde{\mathbf{z}}_n^{t^\star}\right)$
**end function**

**Theorem 6** (Correctness)**.** *Let $\Pi$ be a finite and satisfiable set of signed clauses. For any datapoint $\mathbf{x}$, model $m_\theta$ and variable ordering, the corresponding output* CASPER$(m_\theta(\mathbf{x}))$ *satisfies $\Pi$ for $\epsilon \to 0^+$.*

Notice that the feasible set $\mathcal{S}_i(\mathbf{a}_{<i})$ depends only on the preceding discrete assignments, rather than on the continuous values produced by the preceding projections. Hence, since the surrogate and exact fixed-point projections select the same candidate at every step (Theorem 3), an early continuous over-correction by the surrogate cannot restrict the feasible sets available to later variables:

**Corollary 2** (Sequential assignment equivalence)**.** *Consider a finite and satisfiable set of signed clauses $\Pi$ and a variable ordering $Y_1; \ldots; Y_n$. Given $\hat{\mathbf{y}} = m_\theta(\mathbf{x})$ for some $\mathbf{x} \in \mathcal{X}$, let $\tilde{\mathbf{a}}$ be the assignment induced by* CASPER$(\hat{\mathbf{y}})$*, $\mathbf{a}$ be the assignment induced by the same procedure when each surrogate projection is replaced by the exact fixed-point projection, and let $\mathbf{a}^\star$ be the sequential feasible argmax assignment, $a_i^\star = \operatorname{argmax}_{t \in \mathcal{S}_i(\mathbf{a}^\star_{<i})} \hat{y}_{i,t}$. Then, for $\epsilon \to 0^+$:*

$$\textit{(i)}\ \mathbf{a} = \tilde{\mathbf{a}} = \mathbf{a}^\star; \qquad \textit{and} \qquad \textit{(ii)}\ \mathcal{S}_i(\mathbf{a}_{<i}) = \mathcal{S}_i(\tilde{\mathbf{a}}_{<i}) = \mathcal{S}_i(\mathbf{a}^\star_{<i}), \forall i \in [n].$$

Proof in Appendix H. Longer sequences therefore only accumulate local continuous discrepancies, characterised at each step by Theorem 4, which do not compound through the feasible sets.

## 4 Experimental Analysis

We conduct our analysis over three tasks: the warcraft pathfinding task [41], 11 variations of the standard MNIST Arithmetic task [36] considering different number of digits to sum and different ways of summing them, and sudoku solving [50]. For each task we apply a timeout (indicated as T/O) of 5s per training-example (consistent with [10]). The experiments for warcraft and MNIST were run on a workstation equipped with 3 NVIDIA RTX 4500 each with 26GB VRAM, while the sudoku solving task was run on a workstation equipped with 1 NVIDIA H200.

### 4.1 Warcraft Pathfinding

The goal is predicting the minimum-cost path from the top-left to the bottom-right corner of an $N \times N$ grid, given an image rendered from Warcraft II terrain tiles where each tile has an associated traversal cost. We consider grid-sizes $N \in \{12, 30\}$ where a path is classed as valid if it is connected, contained within the grid, and terminates at the goal. To exemplify the task, we show on qualitative example in Figure 1. In Table 2, we report the percentage of test instances for which each method predicts an optimal path, i.e., a path with minimum cost, and a consistent path, i.e., a path satisfying the task constraints. These are denoted by Optimal(%) and Consistent(%), respectively. To complement these metrics, we also report a cumulative cost-optimality analysis: for each tolerance factor $\alpha \geq 1$, we measure the fraction of predictions whose cost is

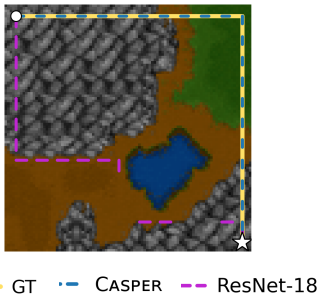



Figure 1: Qualitative example for Warcraft Pathfinding. GT stands for ground truth.

Table 2: Warcraft Pathfinding Task.

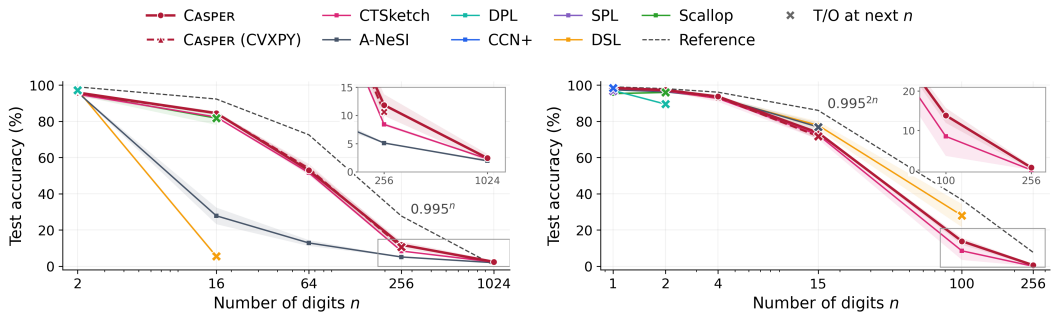
| Method | 12×12 | | 30×30 | |
|---|---|---|---|---|
| | Optimal(%) | Consistent (%) | Optimal(%) | Consistent (%) |
| ResNet-18 | 59.7±1.2 | 62.5±1.0 | 5.1±0.6 | 6.4±1.1 |
| *Exact Methods* | | | | |
| DPL | T/O | T/O | T/O | T/O |
| CCN+ | 74.2±0.6 | **100.0±0.0** | T/O | T/O |
| SPL | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | |
| Scallop | 63.6±5.3 | 66.1±5.7 | T/O | T/O |
| DSL | 65.1±1.3 | 68.3±1.8 | 4.8±0.5 | 5.8±0.7 |
| CTSketch | 60.4±1.1 | 62.8±1.1 | 5.1±0.3 | 6.1±0.6 |
| *Ablation Studies* | | | | |
| CASPER (Train Only) | 57.8±0.9 | 60.5±0.7 | 5.7±0.2 | 9.4±1.6 |
| CASPER (Inference Only) | 78.3±0.8 | **100.0±0.0** | 11.5±0.8 | **100.0±0.0** |
| CASPER (w/o $\ell_2$) | 77.5±0.6 | **100.0±0.0** | 11.0±1.4 | **100.0±0.0** |
| CASPER | 78.7±0.3 | **100.0±0.0** | **12.1±1.1** | **100.0±0.0** |

within a factor $\alpha$ of the optimal path cost in Figure 3 (Appendix N.1). This provides a finer-grained view of how close non-optimal predictions are to the optimum. We compare CASPER against both exact (SPL [3], DPL [36], CCN+ [25]) and approximate baselines (DSL [35], Scallop [30], CTSketch [10]). We evaluate different variants of CTSketch and report the best-performing configuration; the full ablation is given in Appendix N.1. We do not include Dijkstra-based methods such as A-NeSI [48] and I-MLE [39], which predict per-tile costs and recover the path with a shortest-path solver. These are complementary approaches that address a different formulation of the task: the solver performs the combinatorial optimization, while the network learns only the tile costs. In our setting, as in SPL [3], the network instead directly predicts the structured output, with no shortest-path solver, and constraints are enforced on this prediction. All models use the standard ResNet-18 [27] backbone. CASPER and all baselines except CTSketch work with the step-wise formulation of the path constraint (Appendix O), which constrains a time-unrolled walk over grid cells and not the path itself. Its feasible set contains every simple path but also walks that revisit a cell, thus for all methods all metrics are computed on the loop-erased path (Appendix N.1).

We further conduct an ablation study by testing the application of CASPER only at inference time (Inference Only), by training CASPER without the $\ell_2$ regularizer (w/o $\ell_2$), and by training with CASPER and then removing it at inference time (Train Only). Table 2 shows three findings. First, CASPER is the only method that produces $100\%$ valid paths on both grid sizes. Among exact methods, only CASPER runs to completion on $30 \times 30$; SPL, DPL, and CCN+ all time out, while CTSketch's optimal-path rate ($5.1\%$) does not exceed the unconstrained ResNet-18 baseline. CASPER reaches $12.1\%$ optimal on $30 \times 30$, more than doubling CTSketch. Second, the ablation study isolates the contribution of each component of CASPER. Applying CASPER only at inference time already reaches $78.3\%$ optimal on $12 \times 12$ and produces valid paths $100\%$ of the times. This demonstrates that CASPER can be deployed as a post-hoc constraint-enforcement step on existing predictors with no retraining. Conversely, training with CASPER but removing it at inference (Train Only) performs similarly to the unconstrained ResNet-18, showing that the gains stem from enforcing the constraints at inference. Third, the $\ell_2$ regularizer provides a consistent improvement on both grid sizes: $+1.2$ points on $12 \times 12$ and $+1.1$ points on $30 \times 30$, in line with Section 3.1.

### 4.2 MNIST Arithmetic

We next investigate the scalability of CASPER by considering two arithmetic reasoning tasks where the constraint structure grows with the input size: MNIST-Sum and MNIST-Add. For MNIST-Sum, we consider the problem of computing the sum of $n$ handwritten digits sampled from the MNIST dataset [33] with $n = 2^k$ for $k \in \{1, 4, 6, 8, 10\}$. For MNIST-Add, the objective is to compute the sum of two $n$-digit numbers represented as sequences of handwritten digits. As in [10], we test




(a) MNIST-Sum (b) MNIST-Add

Figure 2: Test accuracy as a function of problem size $n$ on MNIST (a) Sum and (b) Add. $\times$ marks the largest $n$ each method completed before timing out or failing.

using $n \in \{2, 15, 100, 256\}$. In both tasks, the constraint requires the final output to follow the sum rules and we train under full supervision. We define a reference accuracy of $0.995^n$ for MNIST-Sum and $0.995^{2n}$ for MNIST-Add, representing the expected accuracy of a MNIST classifier with $99.5\%$ per-digit accuracy. In addition to the previous task baselines, we include A-NeSI [48], and we evaluate a second variant of CASPER that for every variable solves the optimization problem in Equation (2) using *cvxpy* [16, 1] with default solver SCS [40]. All methods use a Le-Net 5 backbone [33].

Table 3: Time (s) per epoch on MNIST-Sum.

| Method / $n$ | 2 | 16 | 64 | 256 | 1024 |
|---|---|---|---|---|---|
| *Exact Methods* | | | | | |
| DPL | 58.4 | T/O | T/O | T/O | T/O |
| CCN+ | 26.9 | T/O | T/O | T/O | T/O |
| SPL | 4.1 | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | | |
| A-NeSI | 11.8 | 24.7 | 66.4 | 232.3 | 995.8 |
| Scallop† | 3100 | 3400 | T/O | T/O | T/O |
| DSL | 33.3 | 386.6 | T/O | T/O | T/O |
| CTSketch | 0.62 | 4.2 | 14.9 | 60.9 | 239.2 |
| CASPER (*cvxpy*) | 33.7 | 94.8 | 257.6 | 815.2 | T/O |
| CASPER | 1.5 | 5.9 | 17.5 | 62.3 | 240.3 |

†Scallop runs on CPU.

We report the results obtained for every dimension in Figure 2 and the time (in seconds) to complete an epoch for each method in Table 3. Figure 2 shows that of the exact methods only CASPER is able to scale from the basic case; DPL, SPL and CCN+ time out for all larger problem sizes. Regarding the approximate methods, on MNIST-Sum we can see that Scallop and DSL fail to scale beyond $n = 16$, with Table 3 showing that Scallop in particular, being CPU bound, incurs a significant increase in training time. Additionally, on MNIST-Sum, for every $n \geq 16$, CASPER outperforms both CTSketch and A-NeSI in terms of accuracy. Regarding MNIST-Add, CASPER is competitive with all approximate methods at all scales, with the exception of DSL and A-NeSI at $n = 15$, and DSL leading at $n = 100$ ($28.0\%$ vs CASPER at $13.8\%$ and CTSketch at $8.5\%$). However, DSL times out for $n = 256$ where CASPER marginally outperforms CTSketch.

Regarding the comparison with CASPER using *cvxpy*, we can see that the *cvxpy* variant performs comparably to the standard CASPER on both MNIST-Sum and MNIST-Add, but times out at $n = 1024$ and $n = 100$, respectively. This confirms the need for the closed-form surrogate: solving the fixed-point system exactly is tractable only at small scale.

### 4.3 Sudoku Solving

We consider the SatNet dataset [50] composed of $10{,}000$ nine-by-nine Sudoku boards constructed with MNIST digits, whose an average number of hints is equal to $36.22$ (min. $31$, max. $42$). The task is to output the completed final board. The pipeline we adopt is the same as [11]: (i) a CNN classifies MNIST hints, (ii) a board solver completes the board and it is followed by (iii) a masking network who blanks out unreliable cells. Finally, (iv) a Prolog solver returns the final board. For (ii) we test two different models: the pretrained neuro-solver from [11] and the LLM Qwen2.5-14B-Instruct [43], which we finetuned over the training SatNet boards. We use the same implementation of CASPER interchangeably on top of both solvers, projecting the cells distribution wrt. the 27 groups, over

9 rows, 9 columns, and 9, three-by-three, boxes. CASPER enforces the non-duplicate constraint individually across each. Table 4, reports the output accuracy of the intermediate solver (*NeuroAcc*), the percentage of CNN errors it can recover from (*RecRate*) and the final Prolog module (*PipeAcc*). On top of improving the final accuracy, CASPER more than doubles the solver performance, while consistently recovering through different solvers almost $95\%$ of the CNN perception errors.

## 5 Related Work

CASPER belongs to a broader effort in neurosymbolic AI [21, 38] to integrate symbolic background knowledge into neural predictors. The earliest neurosymbolic methods incorporate constraints as differentiable training signals [18, 6, 17, 47, 44, 54, 4, 42]. These methods incur no inference-time overhead but cannot guarantee that test predictions satisfy the constraint. A second family treats the symbolic component as a probabilistic program. DeepProbLog [36] and NeurASP [55] perform exact weighted model counting (WMC), but inference is #P-hard. Scallop [30] retains only the top-$k$ proofs, so discarded proofs contribute neither probability mass nor gradient; DeepSoftLog [35] adds soft unification and, on multi-digit addition, retains exact inference through a hand-designed digit-wise carry recursion, outperforming CASPER on MNIST-Add at $n = 100$, although this factorisation does not directly apply to MNIST-Sum, where it times out beyond 16 digits; DeepStochLog [53] replaces probabilistic with stochastic semantics. A complementary line bypasses logic-programming languages: A-NeSI [48] amortises WMC with a learned neural surrogate, introducing approximation and generalisation error; CTSketch [10] encodes the program as a tensor compressed via tensor-train decomposition; Softened Symbol Grounding [34] uses MCMC to sample symbol assignments consistent with the constraint. Across this family, scalability is bought at the cost of exactness or, in the case of DeepSoftLog, generality, and constraint satisfaction cannot be guaranteed at test time. The closest family to ours injects constraints at the predictive layer, guaranteeing that every output is feasible. C-HMCNN [23], CCN [24] and CCN+ [25] clamp predictions into coherence intervals; SPL [3] multiplies a predictive probabilistic circuit by a compiled constraint circuit and renormalises; MultiplexNet [29] parameterises outputs as mixtures over feasible regions. The same principle has recently been applied to controlled language generation, e.g., Ctrl-G [56]. CASPER belongs to this final family but breaks its scalability bottleneck. CASPER natively handles categorical variables via a closed-form Euclidean projection, generalising the simplex projection of Duchi et al. [20] to argmax constraints, that is computable in a single forward pass with parallel depth bounded by the number of variables, independent of constraint complexity. Conceptually, CASPER is also close to differentiable convex-optimisation layers [5, 2], but differently from them it is specialised to categorical structured prediction.

Table 4: Sudoku solving test results.

| **Method** | *NeuroAcc* | *RecRate* | *PipeAcc* |
|---|---|---|---|
| *Board solver* from [11] | | | |
| A-NesI | $18.9_{\pm 7.3}$ | $89.5_{\pm 1.3}$ | $70.0_{\pm 24.7}$ |
| CTSketch | $25.2_{\pm 0.3}$ | $88.2_{\pm 2.3}$ | $81.0_{\pm 1.7}$ |
| **CASPER** | **66.0**$_{\pm 0.0}$ | **94.2**$_{\pm 1.1}$ | **84.7**$_{\pm 0.7}$ |
| *LLM board solver (Qwen2.5-14B-Instruct finetuned)* | | | |
| A-NesI | $25.6_{\pm 0.2}$ | $94.7_{\pm 0.7}$ | $68.3_{\pm 2.4}$ |
| CTSketch | $25.4_{\pm 0.0}$ | $88.6_{\pm 0.4}$ | $80.0_{\pm 0.0}$ |
| **CASPER** | **65.3**$_{\pm 0.5}$ | **95.3**$_{\pm 0.6}$ | **81.8**$_{\pm 1.0}$ |

## 6 Conclusions

In this paper we presented CASPER, a differentiable neurosymbolic layer that enforces hard logical constraints over categorical variables while scaling to thousands of variables. In the single-variable case, constraint satisfaction reduces to a Euclidean projection onto the argmax-feasible region of the simplex, which we solve via a closed-form surrogate with exactly characterised error and gradient behaviour. The general case is handled by an offline compilation step based on signed resolution that reduces multi-variable satisfaction to a sequence of such single-variable projections, parallelizable across variables which are independent according to the constraints. Across our benchmarks, CASPER is the only method that simultaneously guarantees satisfaction and scales. Indeed, it predicts valid paths on 100% of 30×30 Warcraft instances, where all exact baselines time out and the strongest approximate baseline falls below 7%. Further, it is the only exact method that scales to MNIST sums of more than a thousand digits. Finally, it more than doubles the post-layer accuracy of the strongest, unconstrained scalable baseline on Sudoku.

## Acknowledgements

This work was supported by the AI for Math Fund, which is managed by Renaissance Philanthropy in partnership with founding donor XTX Markets. Lohith Konathala is funded by the Engineering and Physical Sciences Research Council (EPSRC) and IceBrook through an Industrial Doctoral Landscape Award (IDLA) studentship. The authors acknowledge the use of resources provided by the Isambard-AI National AI Research Resource (AIRR). Isambard-AI is operated by the University of Bristol and is funded by the UK Government's Department for Science, Innovation and Technology (DSIT) via UK Research and Innovation; and the Science and Technology Facilities Council [ST/AIRR/I-A-I/1023]. We thank the anonymous reviewers and area chairs for their helpful comments and feedback.

## References

[1] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[2] Akshay Agrawal, Brandon Amos, Shane T. Barratt, Stephen P. Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9558–9570, 2019.

[3] Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. *Advances in Neural Information Processing Systems*, 35:29944–29959, 2022.

[4] Kareem Ahmed, Eric Wang, Kai-Wei Chang, and Guy Van den Broeck. Neuro-symbolic entropy regularization. In James Cussens and Kun Zhang, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*, Proceedings of Machine Learning Research, pages 43–53. PMLR, 2022. URL `https://proceedings.mlr.press/v180/ahmed22a.html`.

[5] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 136–145. PMLR, 2017.

[6] Samy Badreddine, Artur S. d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303, 2022.

[7] Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lio, Frédéric Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In *Proceedings of ICML*, 2023.

[8] Bernhard Beckert, Reiner Hähnle, and Felip Manyà. Transformations between signed and classical clause logic. In *29th IEEE International Symposium on Multiple-Valued Logic, ISMVL 1999, Freiburg im Breisgau, Germany, May 20-22, 1999, Proceedings*, 1999.

[9] Bernhard Beckert, Reiner Hähnle, and Felip Manyà. *The SAT Problem of Signed CNF Formulas*, pages 59–80. 2000. ISBN 978-94-011-4040-9. doi: 10.1007/978-94-011-4040-9_3. URL `https://doi.org/10.1007/978-94-011-4040-9_3`.

[10] Seewon Choi, Alaia Solko-Breslin, Rajeev Alur, and Eric Wong. Ctsketch: Compositional tensor sketching for scalable neurosymbolic learning. *arXiv preprint arXiv:2503.24123*, 2025.

[11] Cristina Cornelio, Jan Stuehmer, Shell Xu Hu, and Timothy Hospedales. Learning where and when to reason in neuro-symbolic inference. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=en9V5F8PR-`.

[12] David Debot, Pietro Barbiero, Francesco Giannini, Gabriele Ciravegna, Michelangelo Diligenti, and Giuseppe Marra. Interpretable concept-based memory reasoning. In *Proceedings of NeurIPS*, 2024.

[13] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00059-4. URL https://www.sciencedirect.com/science/article/pii/S0004370299000594.

[14] Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN 978-3-031-01566-3. doi: 10.2200/S00529ED1V01Y201308AIM023.

[15] Rina Dechter and Irina Rish. Directional resolution: The davis-putnam procedure, revisited. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994*, pages 134–145. Morgan Kaufmann, 1994.

[16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[17] Michelangelo Diligenti, Soumali Roychowdhury, and Marco Gori. Integrating prior knowledge into deep learning. In *Proceedings of ICMLA*, 2017.

[18] Ivan Donadello, Luciano Serafini, and Artur D'Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1596–1602. AAAI Press, 2017. ISBN 9780999241103.

[19] Priya L. Donti, David Rolnick, and J. Zico Kolter. DC3: A learning method for optimization with hard constraints. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=V1ZHVxJ6dSS.

[20] John C. Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the $l_1$-ball for learning in high dimensions. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, ACM International Conference Proceeding Series, pages 272–279. ACM, 2008. doi: 10.1145/1390156.1390191. URL https://doi.org/10.1145/1390156.1390191.

[21] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 2023.

[22] Eli Gafni and Dimitri P. Bertsekas. Two-metric projection methods for constrained optimization. *Siam Journal on Control and Optimization*, 22:936–964, 1984. URL https://api.semanticscholar.org/CorpusID:15189707.

[23] Eleonora Giunchiglia and Thomas Lukasiewicz. Coherent hierarchical multi-label classification networks. In *Proceedings of NeurIPS*, 2020.

[24] Eleonora Giunchiglia and Thomas Lukasiewicz. Multi-label classification neural networks with hard logical constraints. *J. Artif. Intell. Res.*, 72:759–818, 2021. doi: 10.1613/JAIR.1.12850. URL https://doi.org/10.1613/jair.1.12850.

[25] Eleonora Giunchiglia, Alex Tatomir, Mihaela Cătălina Stoian, and Thomas Lukasiewicz. Ccn+: A neuro-symbolic framework for deep learning with requirements. *International Journal of Approximate Reasoning*, 171:109124, 2024.

[26] Reiner Hähnle. Short conjunctive normal forms in finitely valued logics. *J. Log. Comput.*, 4(6):905–927, 1994. doi: 10.1093/LOGCOM/4.6.905. URL https://doi.org/10.1093/logcom/4.6.905.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[28] Michael Held, Philip Wolfe, and Harlan P. Crowder. Validation of subgradient optimization. *Math. Program.*, 6(1):62–88, 1974. doi: 10.1007/BF01580223. URL https://doi.org/10.1007/BF01580223.

[29] Nicholas Hoernle, Rafael-Michael Karampatsis, Vaishak Belle, and Kobi Gal. MultiplexNet: Towards fully satisfied logical constraints in neural networks. In *Proceedings of Association for the Advancement of Artificial Intelligence*, 2022.

[30] Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. *Advances in Neural Information Processing Systems*, 34:25134–25145, 2021.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 13–18 Jul 2020.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[34] Zenan Li, Yuan Yao, Taolue Chen, Jingwei Xu, Chun Cao, Xiaoxing Ma, and Jian Lü. Softened symbol grounding for neuro-symbolic systems. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=HTJE5Krui0g`.

[35] Jaron Maene and Luc De Raedt. Soft-unification in deep probabilistic logic. *Advances in Neural Information Processing Systems*, 36:60804–60820, 2023.

[36] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

[37] Emanuele Marconato, Stefano Teso, Antonio Vergari, and Andrea Passerini. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/e560202b6e779a82478edb46c6f8f4dd-Abstract-Conference.html`.

[38] Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artif. Intell.*, 328:104062, 2024. doi: 10.1016/J.ARTINT.2023.104062. URL `https://doi.org/10.1016/j.artint.2023.104062`.

[39] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit MLE: backpropagating through discrete exponential family distributions. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 14567–14579, 2021.

[40] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016. URL `http://stanford.edu/~boyd/papers/scs.html`.

[41] Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International conference on learning representations*, 2019.

[42] Connor Pryor, Charles Dickens, Eriq Augustine, Alon Albalak, William Yang Wang, and Lise Getoor. Neupsl: Neural probabilistic soft logic. In *Proceedings of IJCAI*, 2023.

[43] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[44] Davide Scassola, Sebastiano Saccani, Ginevra Carbone, and Luca Bortolussi. Zero-shot conditioning of score-based diffusion models by neuro-symbolic constraints. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 20302–20309. AAAI Press, 2025.

[45] Lennert De Smet, Pedro Zuidberg Dos Martires, Robin Manhaeve, Giuseppe Marra, Angelika Kimmig, and Luc De Raedt. Neural probabilistic logic programming in discrete-continuous domains. In Robin J. Evans and Ilya Shpitser, editors, *Uncertainty in Artificial Intelligence, UAI 2023, July 31 - 4 August 2023, Pittsburgh, PA, USA*, Proceedings of Machine Learning Research, pages 529–538. PMLR, 2023. URL https://proceedings.mlr.press/v216/de-smet23a.html.

[46] Mihaela Catalina Stoian and Eleonora Giunchiglia. Beyond the convexity assumption: Realistic tabular data generation under quantifier-free real linear constraints. In *Proceedings of ICLR*, 2025.

[47] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artif. Intell.*, 302:103602, 2022. doi: 10.1016/J.ARTINT.2021.103602. URL https://doi.org/10.1016/j.artint.2021.103602.

[48] Emile van Krieken, Thiviyan Thanapalasingam, Jakub Tomczak, Frank Van Harmelen, and Annette Ten Teije. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. *Advances in Neural Information Processing Systems*, 36:24586–24609, 2023.

[49] Emile van Krieken, Pasquale Minervini, Edoardo Ponti, and Antonio Vergari. Neurosymbolic diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=HfdzglsZQH.

[50] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.

[51] Runzhong Wang, Yunhao Zhang, Ziao Guo, Tianyi Chen, Xiaokang Yang, and Junchi Yan. Linsatnet: The positive linear satisfiability neural networks. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 36605–36625. PMLR, 2023.

[52] Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 54905–54931. Curran Associates, Inc., 2024. doi: 10.52202/079017-1741. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/62c4718cc334f6a0a62fb81c4a2095a1-Paper-Conference.pdf.

[53] Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. Deepstochlog: Neural stochastic logic programming. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 10090–10100. AAAI Press, 2022. doi: 10.1609/AAAI.V36I9.21248. URL https://doi.org/10.1609/aaai.v36i9.21248.

[54] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR, 2018.

[55] Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In *Proceedings of IJCAI*, 2020.

[56] Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. Adaptable logical control for large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

# A Limitations

CASPER targets a specific corner of the neurosymbolic problem space — guaranteed satisfaction of hard logical constraints over discrete categorical variables — and the design choices that make this corner tractable at scale carry their own limitations, which we discuss here.

The simplex projection at the heart of CASPER is built for finite-domain categorical variables and signed clauses. Constraints over continuous outputs [46, 51, 45, 19], soft constraints encoding preferences, and constraints mixing discrete and continuous quantities all fall outside its scope: handling them would require redesigning the projection geometry rather than re-instantiating it. Within the categorical setting, our approach also relies on a closed-form over-approximation $\mathcal{A}^t$ of the exact active set $\mathcal{M}^t$ (Lemma 1) in order to retain $\mathcal{O}(\log h)$ parallel depth. This approximation affects only the continuous output of CASPER: the surrogate selects the same value as the exact projection at every step (Theorem 3 and Corollary 2), and Theorem 4 characterises the resulting gap exactly, with Corollary 1 bounding it for every input. The $\ell_2$ regulariser (Proposition 2) shrinks this bound during training, but only on average over the training distribution, not on every test input. Inference-only deployments, which never see the regulariser, therefore incur a small but measurable degradation, as the (Inference-Only) ablation in Table 2 shows.

Finally, several active areas of neurosymbolic learning are complementary to ours rather than addressed by it: reasoning shortcuts [37, 38], modelling dependencies between predicted symbols [49], and concept-based interpretability [32, 12, 7]. CASPER composes naturally with methods targeting these axes — for instance, the categorical output of a concept bottleneck model can itself serve as the variables on which CASPER enforces constraints — but a thorough empirical study of such combinations is beyond the scope of this paper.

# B Proof of Theorem 1

**Statement.** *Let $\hat{\mathbf{y}} \in \Delta([h])$ and $\mathcal{S} \subseteq [h]$, the $\epsilon$-margin solution to the optimization problem in*

$$\operatorname{argmin}_{\mathbf{z}\in\Delta([h])} \|\mathbf{z}-\hat{\mathbf{y}}\|_2^2 \qquad \textit{subject to } \operatorname*{argmax}_{j\in[h]} \mathbf{z} \in \mathcal{S}. \tag{13}$$

*is given by $\mathbf{z}^{t^\star}$ with $t^\star$ equal to*

$$t^\star \in \operatorname*{argmin}_{t\in\mathcal{S}} \|\mathbf{z}^t-\hat{\mathbf{y}}\|_2^2, \tag{14}$$

*where for every $t \in \mathcal{S}$, $\mathbf{z}^t$ is the solution of the convex minimization problem obtained by fixing $t$ as the unique maximizer:*

$$z_j^t = \begin{cases} \hat{y}_j & \text{if } j \notin (\mathcal{M}^t \cup \{t\}) \\ z_t^t - \epsilon & \text{if } j \in \mathcal{M}^t \\ \frac{\hat{y}_t + \sum_{j\in\mathcal{M}^t} \hat{y}_j + |\mathcal{M}^t|\epsilon}{1+|\mathcal{M}^t|} & \text{if } j = t \end{cases}, \tag{15}$$

*$\mathcal{M}^t = \{j \in [h] \setminus \{t\} : \hat{y}_j \geq z_t^t - \epsilon\}$ represents the set of active categories given $t$, and $\epsilon > 0$ is an arbitrarily small constant.*

*Proof.* The $\epsilon$-margin feasible space of Equation (13) decomposes as

$$\{z \in \Delta([h]) : \exists\, t \in \mathcal{S},\ z_t \geq z_j + \epsilon\ \forall j \neq t\} = \bigcup_{t\in\mathcal{S}} C^t, \quad C^t = \{z : z_t \geq z_j + \epsilon\}.$$

Since every $C^t$ is an intersection of halfspaces with the simplex, it is closed and convex, and the objective $\|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2$ is convex and continuous. Hence,

$$\min_{\mathbf{z}\in\bigcup_{t\in\mathcal{S}} C^t} \|\mathbf{z}-\hat{\mathbf{y}}\|_2^2 = \min_{t\in\mathcal{S}} \min_{\mathbf{z}\in C^t} \|\mathbf{z}-\hat{\mathbf{y}}\|_2^2.$$

This proves Equation (14) is the $\epsilon$-margin solution of the optimization problem in Equation (13) provided that $\mathbf{z}^t$ is equal to Equation (15).

Consider $t \in \mathcal{S}$. We solve

$$\min_{\mathbf{z}} \|\mathbf{z}-\hat{\mathbf{y}}\|_2^2$$

subject to

$$z_t \geq z_j + \epsilon \quad \forall j \in [h] \setminus \{t\}, \qquad \sum_{j=1}^{h} z_j = 1, \qquad z_j \geq 0 \quad \forall j \in [h]. \tag{16}$$

This yields a well-posed convex problem enforcing uniqueness of the maximizer with margin $\epsilon$.

The Lagrangian is:

$$\mathcal{L} = \sum_{j=1}^{h} (z_j - \hat{y}_j)^2 + \nu \left( \sum_{j=1}^{h} z_j - 1 \right) - \sum_{j \in [h] \setminus \{t\}} \lambda_j (z_t - z_j - \epsilon) - \sum_{j=1}^{h} \mu_j z_j.$$

The KKT conditions (which include the primal feasibility in Equation (16)) give us:

$$\frac{\partial \mathcal{L}}{\partial z_j} = 2(z_j - \hat{y}_j) + \nu + \lambda_j - \mu_j = 0, \qquad \forall j \in [h] \setminus t \qquad \text{(Stationarity (1))}$$

$$\frac{\partial \mathcal{L}}{\partial z_t} = 2(z_t - \hat{y}_t) + \nu - \sum_{j \in [h] \setminus \{t\}} \lambda_j - \mu_t = 0 \qquad \text{(Stationarity (2))}$$

$$\lambda_j \geq 0, \qquad \mu_j \geq 0 \qquad \forall j \in [h] \qquad \text{(Dual Feasibility)}$$

$$\lambda_j (z_t - z_j - \epsilon) = 0 \qquad \forall j \in [h] \setminus \{t\} \qquad \mu_j z_j = 0 \qquad \forall j \in [h] \qquad \text{(Complementary Slackness)}$$

Let

- $\mathcal{Z}^t = \{j \in [h] : \mu_j \neq 0\}$ be the set of categories for which the zero constraint is active,
- $\mathcal{M}^t = \{j \in [h] \setminus t : \lambda_j \neq 0\}$ be the set of categories for which the margin constraint is active, and
- $\mathcal{F}^t = [h] \setminus (\mathcal{M}^t \cup \mathcal{Z}^t \cup \{t\})$ the free set where neither of the constraints is active.

Notice that these sets, together with $\{t\}$, form a partition of $[h]$.

Indeed, the primal feasibility conditions impose that $\mathbf{z} \in \Delta([h])$ and $z_t - z_j \geq \epsilon$ for all $j \neq t$. If $z_t = \epsilon$, then for all $j \in [h] \setminus \{t\}$ we would have

$$z_j \leq z_t - \epsilon = 0.$$

Since $z_j \geq 0$ for all $j \in [h]$ and $\epsilon < 1$, it follows that $z_j = 0$ for all $j \neq t$. Therefore,

$$\sum_{j=1}^{h} z_j = z_t = \epsilon \neq 1,$$

which contradicts the simplex constraint $\mathbf{z} \in \Delta$. Hence $z_t > \epsilon$, and in particular $z_t - \epsilon > 0$.

It follows that no index $j \neq t$ can simultaneously satisfy

$$z_j = 0 \quad \text{and} \quad z_j = z_t - \epsilon,$$

and hence $\mathcal{Z}^t \cap \mathcal{M}^t = \emptyset$.

Then:

- if $j \in \mathcal{Z}^t$ then $z_j = 0$. We know that $\lambda_j = 0$ and we can use the stationarity condition together with the dual feasibility to identify the coordinates belonging to this set as:

  $$\mu_j = \nu - 2\hat{y}_j \geq 0 \qquad \hat{y}_j \leq \tau, \tag{17}$$

  where $\tau = \frac{\nu}{2}$.
- if $j \in \mathcal{F}^t$ then, following the stationarity condition, $z_j = \hat{y}_j - \frac{\nu}{2}$.
- if $j \in \mathcal{M}^t$ then $z_j = z_t - \epsilon$. We know that $\mu_j = 0$, and we can use the stationarity condition to get:

  $$\lambda_j = 2(\hat{y}_j - z_j) - \nu = 2(\hat{y}_j - z_t + \epsilon) - \nu = 2(\hat{y}_j - z_t + \epsilon - \tau)$$

Since we know that for all $j \in \mathcal{Z}^t \cup \mathcal{F}^t$, $\lambda_j = 0$ we can rewrite the first stationarity condition as:

$$2(z_t - \hat{y}_t) + \nu - \sum_{j \in \mathcal{M}^t} \lambda_j = 0$$

$$2(z_t - \hat{y}_t) + \nu - \sum_{j \in \mathcal{M}^t} 2(\hat{y}_j - z_t + \epsilon - \tau) = 0$$

Let $m = |\mathcal{M}^t|$,

$$(z_t - \hat{y}_t) + \tau - \sum_{j \in \mathcal{M}^t} \hat{y}_j + m z_t - m\epsilon + m\tau = 0$$

$$(m+1) z_t = \hat{y}_t + \sum_{j \in \mathcal{M}^t} \hat{y}_j + m\epsilon - (m+1)\tau \tag{18}$$

To determine $\tau$, we impose the simplex constraint $\sum_{j=1}^{h} z_j = 1$ and decompose the sum according to the partition $\{\mathcal{Z}^t, \mathcal{F}^t, \mathcal{M}^t\}$ of $[h] \setminus \{t\}$. By construction,

$$z_j = \begin{cases} 0, & j \in \mathcal{Z}^t, \\ \hat{y}_j - \tau, & j \in \mathcal{F}^t, \\ z_t - \epsilon, & j \in \mathcal{M}^t, \end{cases} \qquad \text{and} \qquad z_t \text{ for } j = t.$$

Hence,

$$1 = z_t + \sum_{j \in \mathcal{F}^t} (\hat{y}_j - \tau) + \sum_{j \in \mathcal{M}^t} (z_t - \epsilon).$$

Let $f = |\mathcal{F}^t|$ and $m = |\mathcal{M}^t|$. Then

$$1 = (m+1) z_t + \sum_{j \in \mathcal{F}^t} \hat{y}_j - f\tau - m\epsilon \qquad\qquad (m+1) z_t = 1 - \sum_{j \in \mathcal{F}^t} \hat{y}_j + f\tau + m\epsilon \tag{19}$$

By setting 18 equal to 19 we obtain

$$\hat{y}_t + \sum_{j \in \mathcal{M}^t} \hat{y}_j + m\epsilon - (m+1)\tau = 1 - \sum_{j \in \mathcal{F}^t} \hat{y}_j + f\tau + m\epsilon$$

$$\sum_{j \in \mathcal{Z}^t} \hat{y}_j = -(h - |\mathcal{Z}^t|)\tau \tag{20}$$

Since by definition $\sum_{j \in \mathcal{Z}^t} \hat{y}_j \geq 0$ and $h - |\mathcal{Z}^t| \geq 1$ (as $t \notin \mathcal{Z}^t$), Equation (20) gives $\tau \leq 0$. However, according to Equation (17), we have that for all $j \in \mathcal{Z}^t$, $0 \leq \hat{y}_j \leq \tau$. Hence, for every $j \in \mathcal{Z}^t$, $\hat{y}_j = 0$, and $\tau = \nu = 0$.

Since $\tau = 0$, we have that for all $j \notin \mathcal{M}^t \cup \{t\}$, $z_j = \hat{y}_j$, hence the sets $\mathcal{F}^t$ and $\mathcal{Z}^t$ are indistinguishable. Hence, we have that:

$$z_j = \begin{cases} \hat{y}_j, & j \notin \mathcal{M}^t \cup \{t\}, \\ z_t - \epsilon, & j \in \mathcal{M}^t, \\ \frac{\hat{y}_t + \sum_{j \in \mathcal{M}^t} \hat{y}_j + m\epsilon}{m+1} & j = t \end{cases} \tag{21}$$

Notice that Equations (22)-(21) define a fixed-point system. Indeed, the set $\mathcal{M}^t$:

$$\mathcal{M}^t = \{j \in [h] \setminus \{t\} \,:\, \hat{y}_j \geq z_t^t - \epsilon\} \tag{22}$$

is defined through $z_t^t$, which in turn depends on $\mathcal{M}^t$. We now show that this system admits a unique solution, which can be found by a one-dimensional sweep over the sorted values of $\hat{\mathbf{y}}$.

Let $\hat{y}^{(1)} \geq \hat{y}^{(2)} \geq \cdots \geq \hat{y}^{(h-1)}$ denote the values $\{\hat{y}_j : j \neq t\}$ sorted in decreasing order, with the convention $\hat{y}^{(0)} := +\infty$ and $\hat{y}^{(h)} := -\infty$. For $a \in \{0, 1, \ldots, h-1\}$, define the *candidate threshold*

$$\theta_a \;=\; \frac{\hat{y}_t + \sum_{k=1}^{a} \hat{y}^{(k)} - \epsilon}{1 + a}. \tag{23}$$

By construction, $\theta_a = z_t^t - \epsilon$ whenever $|\mathcal{M}^t| = a$ and $\mathcal{M}^t$ consists of the top-$a$ indices.

To prove the uniqueness of the solution, we prove that there exists a unique integer $a^\star \in \{0, 1, \ldots, h-1\}$ such that

$$\hat{y}^{(a^\star)} > \theta_{a^\star} \qquad \text{and} \qquad \hat{y}^{(a^\star+1)} \leq \theta_{a^\star}, \tag{24}$$

and that setting $\mathcal{M}^t$ to be the top-$a^\star$ indices of $\{\hat{y}_j : j \neq t\}$ and $z_t^t = \theta_{a^\star} + \epsilon$ yields the unique solution of the fixed-point system (22)–(21).

Indeed, from (23):

$$\theta_{a+1} - \theta_a = \frac{\hat{y}^{(a+1)} - \theta_a}{2+a}, \tag{25}$$

which implies the equivalence

$$\theta_{a+1} > \theta_a \iff \hat{y}^{(a+1)} > \theta_a, \tag{26}$$

and similarly for the reverse inequality.

*Existence.* Define

$$a^\star = \max\big\{\, a \in \{0, 1, \ldots, h-1\} : \hat{y}^{(a)} > \theta_{a-1} \,\big\},$$

where the set is nonempty because $\hat{y}^{(0)} = +\infty > \theta_{-1}$ with the convention $\theta_{-1} = -\infty$. By definition, $\hat{y}^{(a^\star)} > \theta_{a^\star-1}$, and applying (26) at index $a^\star - 1$ yields $\theta_{a^\star} > \theta_{a^\star-1}$. Rearranging (25) gives

$$\theta_{a^\star} = \frac{a^\star \theta_{a^\star-1} + \hat{y}_{(a^\star)}}{1+a^\star},$$

which is a convex combination of $\theta_{a^\star-1}$ and $\hat{y}_{(a^\star)}$; since $\theta_{a^\star-1} < \hat{y}^{(a^\star)}$, we obtain $\hat{y}^{(a^\star)} > \theta_{a^\star}$, the first part of (24). If $a^\star = h-1$, the second part of (24) holds vacuously via $\hat{y}^{(h)} = -\infty$; otherwise, maximality of $a^\star$ gives $\hat{y}^{(a^\star+1)} \leq \theta_{a^\star}$.

*Uniqueness.* Suppose $a_1 < a_2$ both satisfy (24). Hence, $\hat{y}^{(a_1+1)} \leq \theta_{a_1}$. We show by induction that

$$\hat{y}^{(b)} \leq \theta_b \qquad \text{for every } b \geq a_1 + 1, \tag{27}$$

which contradicts the strict inequality $\hat{y}^{(a_2)} > \theta_{a_2}$ required by (24) at $a_2$.

Rearranging (25) yields the identity

$$\theta_{b+1} - \hat{y}^{(b+1)} = \frac{1+b}{2+b}\big(\theta_b - \hat{y}^{(b+1)}\big). \tag{28}$$

*Base case* ($b = a_1 + 1$). Applying (28) at index $a_1$,

$$\theta_{a_1+1} - \hat{y}^{(a_1+1)} = \frac{1+a_1}{2+a_1}\big(\theta_{a_1} - \hat{y}^{(a_1+1)}\big) \geq 0$$

by the hypothesis $\hat{y}^{(a_1+1)} \leq \theta_{a_1}$.

*Inductive step.* Assume $\theta_b \geq \hat{y}^{(b)}$. Since $\hat{y}^{(\cdot)}$ is non-increasing, $\hat{y}^{(b+1)} \leq \hat{y}^{(b)} \leq \theta_b$, so the right-hand side of (28) is non-negative, giving $\theta_{b+1} \geq \hat{y}^{(b+1)}$.

Iterating up to $b = a_2$ gives $\hat{y}^{(a_2)} \leq \theta_{a_2}$, the desired contradiction.

*Consistency with the fixed-point system.* With $a^\star$ as above, set $\mathcal{M}^t$ to be the top-$a^\star$ indices and $z_t^t = \theta_{a^\star} + \epsilon$. Then (21) holds by construction of $\theta_{a^\star}$, and (22) holds by (24): every index in $\mathcal{M}^t$ has $\hat{y}_j \geq \hat{y}_{(a^\star)} > \theta_{a^\star} = z_t^t - \epsilon$, and every index outside has $\hat{y}_j \leq \hat{y}_{(a^\star+1)} \leq \theta_{a^\star} = z_t^t - \epsilon$. □

**Computing $\mathbf{z}^t$.** Given an unconstrained output $\hat{\mathbf{y}}$ and a constraint stating that the final assignment must belong to $\mathcal{S}$, for every $t \in \mathcal{S}$ we can compute $\mathbf{z}^t$ by modifying the sorting method used to project onto the probabilistic simplex [28, 22, 20]:

1. Sort $\{\hat{y}_j : j \neq t\}$ in decreasing order as $\hat{y}_1 \geq \cdots \geq \hat{y}_{h-1}$.
2. Compute the cumulative sums and evaluate $\theta_a$ for $a = 0, 1, \ldots, h-1$ via (23) in $\mathcal{O}(h)$ time.

3. Find the unique $a^\star$ satisfying (24) (equivalently, the largest $a$ with $\hat{y}_{(a)} > \theta_{a-1}$).
4. Set $\mathcal{M}^t$ to be the corresponding top-$a^\star$ indices and return $\mathbf{z}^t$ via (4) with $z_t^t = \theta_{a^\star} + \epsilon$.

This algorithm needs to perform $\mathcal{O}(h \log h)$ sequential operations, dominated by the sort, while it requires only $\mathcal{O}(\log^2 h)$ sequential GPU operations thanks to parallelization. Alternatively, the convex problem can be solved with a generic solver (like [16]), applying an iterative interior-point or ADMM method.

## C Proof of Lemma 1

**Statement.** *For every $t \in [h]$ and every $\hat{\mathbf{y}} \in \Delta([h])$, $\mathcal{M}^t \subseteq \mathcal{A}^t$.*

*Proof.* Let $j \in \mathcal{M}^t$. From $\mathcal{M}^t$ definition we know that $\hat{y}_j \geq z_t^t - \epsilon$. To show that $\mathcal{M}^t \subseteq \mathcal{A}^t$. we need to demonstrate that $z_t^t \geq \hat{y}_t$.

Using the closed form (21) and writing $m = |\mathcal{M}^t|$,

$$z_t^t - \hat{y}_t \;=\; \frac{\hat{y}_t + \sum_{k \in \mathcal{M}^t} \hat{y}_k + m\,\epsilon}{1+m} - \hat{y}_t \;=\; \frac{\sum_{k \in \mathcal{M}^t} \hat{y}_k - m\,\hat{y}_t + m\,\epsilon}{1+m}. \tag{29}$$

If $m = 0$, the right-hand side vanishes and $z_t^t = \hat{y}_t$.

If $m \geq 1$, each $k \in \mathcal{M}^t$ satisfies $\hat{y}_k \geq z_t^t - \epsilon$ yielding $\sum_{k \in \mathcal{M}^t} \hat{y}_k \geq m\,(z_t^t - \epsilon)$. Substituting into (29),

$$z_t^t - \hat{y}_t \geq \frac{m\,(z_t^t - \epsilon) - m\,\hat{y}_t + m\,\epsilon}{1+m} \geq \frac{m}{1+m}\,(z_t^t - \hat{y}_t),$$

which rearranges to $z_t^t \geq \hat{y}_t$.

□

## D Proof of Theorem 2

**Statement.** *Let $m_\theta : \mathcal{X} \to \Delta([h])$, $\mathbf{x} \in \mathcal{X}$ and $\hat{\mathbf{y}} = m_\theta(\mathbf{x})$. Let $\Pi = \{\mathcal{S} : Y\}$ for some $\emptyset \subset \mathcal{S} \subseteq [h]$. If*

$$\textsc{Casper}(\hat{\mathbf{y}}) = \tilde{\mathbf{z}}^{t^\star}, \qquad t^\star = \operatorname*{argmin}_{t \in \mathcal{S}} \|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\|_2^2, \tag{30}$$

*where for every $t \in \mathcal{S}$*

$$\tilde{z}_j^t = \begin{cases} \hat{y}_j & \text{if } j \notin (\mathcal{A}^t \cup \{t\}) \\ \tilde{z}_t^t - \epsilon & \text{if } j \in \mathcal{A}^t \\ \frac{\hat{y}_t + \sum_{j \in \mathcal{A}^t} \hat{y}_j + |\mathcal{A}^t|\epsilon}{1+|\mathcal{A}^t|} & \text{if } j = t \end{cases} \qquad \mathcal{A}^t = \{j \in [h] \setminus \{t\} : \hat{y}_j \geq \hat{y}_t - \epsilon\}, \tag{31}$$

*then for $\epsilon \to 0^+$, the assignment $\mathbf{a}$ induced by $\textsc{Casper}(\hat{\mathbf{y}})$ satisfies $\Pi$, i.e., $\mathbf{a} \models \Pi$.*

*Proof.* We show that for every $t \in \mathcal{S}$, the surrogate $\tilde{\mathbf{z}}^t$ defined by Eq. (30) lies in the simplex $\Delta([h])$ and has $t$ as its unique argmax. Applied to $t = t^\star$, this gives $\mathbf{a} = \operatorname{argmax}_j \tilde{z}_j^{t^\star} = t^\star \in \mathcal{S}$, hence $\mathbf{a} \models \Pi$.

Let $t \in \mathcal{S}$. We first show that $\tilde{\mathbf{z}}^t \in \Delta([h])$:

$$\begin{aligned} \sum_{j \in [h]} \tilde{z}_j^t &= \sum_{j \notin \mathcal{A}^t \cup \{t\}} \hat{y}_j + \sum_{j \in \mathcal{A}^t} (\tilde{z}_t^t - \epsilon) + \tilde{z}_t^t \\ &= \sum_{j \notin \mathcal{A}^t \cup \{t\}} \hat{y}_j + (|\mathcal{A}^t| + 1)\frac{\hat{y}_t + \sum_{j \in \mathcal{A}^t} \hat{y}_j + |\mathcal{A}^t|\epsilon}{1 + |\mathcal{A}^t|} - |\mathcal{A}^t|\epsilon \\ &= \sum_{j \in [h]} \hat{y}_j = 1, \end{aligned}$$

Non-negativity of $\tilde{z}_j^t$ follows from $\hat{\mathbf{y}} \in \Delta([h])$ for the unmodified entries, from $\tilde{z}_t^t$ being a weighted average of non-negative quantities, and from $\tilde{z}_t^t - \epsilon \geq \hat{y}_t - \epsilon \geq 0$ on $\mathcal{A}^t$ provided $\epsilon \leq \hat{y}_t$, which holds for $\epsilon$ sufficiently small. Hence $\tilde{\mathbf{z}}^t \in \Delta([h])$.

To prove $t = \operatorname{argmax}_j \tilde{z}_j^t$, we compare $\tilde{z}_t^t$ against the elements in both $\mathcal{A}^t$ and $[h] \setminus (\mathcal{A}^t \cup \{t\})$.

For $j \in \mathcal{A}^t$: $\tilde{z}_j^t = \tilde{z}_t^t - \epsilon < \tilde{z}_t^t$ for any $\epsilon > 0$.

For $j \notin \mathcal{A}^t \cup \{t\}$: $\tilde{z}_j^t = \hat{y}_j < \hat{y}_t - \epsilon$ by definition of $\mathcal{A}^t$. Since every $k \in \mathcal{A}^t$ satisfies $\hat{y}_k \geq \hat{y}_t - \epsilon$, $\sum_{k \in \mathcal{A}^t} \hat{y}_k \geq |\mathcal{A}^t|(\hat{y}_t - \epsilon)$, and substituting into the closed form for $\tilde{z}_t^t$ yields $\tilde{z}_t^t \geq \hat{y}_t$. Combining, $\tilde{z}_t^t \geq \hat{y}_t > \hat{y}_t - \epsilon > \hat{y}_j = \tilde{z}_j^t$. □

# E Proof of Theorem 3

**Statement.** *Let $\hat{\mathbf{y}} \in \Delta([h])$, $\mathbf{z}^t$ be the solution of the fixed point system* (4) *and $\tilde{\mathbf{z}}^t$ its surrogate solution as defined in Equation* (6)*. Then, for $\epsilon \to 0^+$,*

$$\operatorname*{argmin}_{t \in \mathcal{S}} \left\|\mathbf{z}^t - \hat{\mathbf{y}}\right\|_2^2 = \operatorname*{argmin}_{t \in \mathcal{S}} \left\|\widetilde{\mathbf{z}}^t - \hat{\mathbf{y}}\right\|_2^2 = \operatorname*{argmax}_{t \in \mathcal{S}} \hat{y}_t.$$

*Proof.* Notice that if $|\mathcal{S}| = 1$ the thesis is trivially satisfied. We hence assume $|\mathcal{S}| > 1$. We prove separately that the surrogate and exact projection criteria are strictly improved whenever a candidate $t$ is replaced by a candidate $t'$ satisfying $\hat{y}_{t'} > \hat{y}_t$; the case $\hat{y}_{t'} = \hat{y}_t$ is treated at the end of the proof.

**Exact fixed-point solution.** Suppose by contradiction that $t = \operatorname{argmin}_{s \in \mathcal{S}} \|\mathbf{z}^s - \hat{\mathbf{y}}\|_2^2$ satisfies $\hat{y}_t < \max_{s \in \mathcal{S}} \hat{y}_s$. Then there exists $t' \in \mathcal{S}$ such that $\hat{y}_{t'} > \hat{y}_t$.

Given $s \in \mathcal{S}$, let $C^s = \{\mathbf{z} \in \Delta([h]) : z_s \geq z_j + \epsilon,\ \forall j \neq s\}$, be the *$s$-selection region*, i.e., the subset of $\Delta([h])$ in which coordinate $s$ is maximal with margin at least $\epsilon$.

Consider an alternative candidate solution to the fixed point system $\overline{\mathbf{z}}^{t'} \in \mathcal{C}^{t'}$ constructed from $\mathbf{z}^t$ by swapping the values at the coordinates $t$ and $t'$:

$$\overline{z}_j^{t'} = \begin{cases} z_{t'}^t, & j = t, \\ z_t^t, & j = t', \\ z_j^t, & j \notin \{t, t'\}. \end{cases}$$

We know that $\overline{\mathbf{z}}^{t'} \in \mathcal{C}^{t'}$ because by definition of $\mathbf{z}^t$, $z_t^t \geq z_j^t + \epsilon$, for every $j \neq t$. For every $j \notin \{t', t\}$, this implies

$$\overline{z}_{t'}^{t'} = z_t^t \geq z_j^t + \epsilon = \overline{z}_j^{t'} + \epsilon, \qquad \text{and} \qquad \overline{z}_{t'}^{t'} = z_t^t \geq z_{t'}^t + \epsilon = \overline{z}_t^{t'} + \epsilon.$$

Since only coordinates $t$ and $t'$ differ between $\overline{\mathbf{z}}^{t'}$ and $\mathbf{z}^t$:

$$\begin{aligned} &\|\overline{\mathbf{z}}^{t'} - \hat{\mathbf{y}}\|_2^2 - \|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2 \\ &= \left(z_{t'}^t - \hat{y}_t\right)^2 + \left(z_t^t - \hat{y}_{t'}\right)^2 - \left(z_t^t - \hat{y}_t\right)^2 - \left(z_{t'}^t - \hat{y}_{t'}\right)^2 \\ &= 2\left(z_t^t - z_{t'}^t\right)\left(\hat{y}_t - \hat{y}_{t'}\right). \end{aligned} \tag{32}$$

Because

$$z_t^t - z_{t'}^t \geq \epsilon > 0$$

and

$$\hat{y}_t - \hat{y}_{t'} < 0,$$

Equation (32) implies

$$\|\overline{\mathbf{z}}^{t'} - \hat{\mathbf{y}}\|_2^2 < \|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2.$$

Let $\mathbf{z}^{t'}$ be the minimizer over $\mathcal{C}^{t'}$, since $\overline{\mathbf{z}}^{t'} \in \mathcal{C}^{t'}$,

$$\|\mathbf{z}^{t'} - \hat{\mathbf{y}}\|_2^2 \leq \|\overline{\mathbf{z}}^{t'} - \hat{\mathbf{y}}\|_2^2 < \|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2,$$

contradicting the assumed optimality of $t$.

Thus, no index with a strictly sub-maximal value of $\hat{y}_t$ can minimize the exact projection distance.

**Surrogate solution.** For every $t \in \mathcal{S}$ and $j \in \mathcal{A}^t$, let

$$r_j^t = \hat{y}_j - \hat{y}_t + \epsilon \qquad \text{and} \qquad \alpha_t = \tilde{z}_t^t - \hat{y}_t = \frac{1}{1+|\mathcal{A}^t|} \sum_{j \in \mathcal{A}^t} r_j^t,$$

$r_j^t$ representing the $\epsilon$-*margin violation for coordinate* $j$. For $j \in \mathcal{A}^t$, we have

$$\tilde{z}_j^t - \hat{y}_j = \tilde{z}_t^t - \epsilon - \hat{y}_j = \alpha_t - r_j^t,$$

while the remaining coordinates are unchanged. Hence,

$$\begin{aligned}
\|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\|_2^2 &= \alpha_t^2 + \sum_{j \in \mathcal{A}^t} \left(\alpha_t - r_j^t\right)^2 \\
&= \alpha_t^2 + |\mathcal{A}^t|\alpha_t^2 - \sum_{j \in \mathcal{A}^t} 2\alpha_t r_j^t + \sum_{j \in \mathcal{A}^t} (r_j^t)^2 \\
&= \frac{1}{1+|\mathcal{A}^t|} \left(\sum_{j \in \mathcal{A}^t} r_j^t\right)^2 - \frac{2}{1+|\mathcal{A}^t|} \left(\sum_{j \in \mathcal{A}^t} r_j^t\right)^2 + \sum_{j \in \mathcal{A}^t} (r_j^t)^2 \\
&= \sum_{j \in \mathcal{A}^t} (r_j^t)^2 - \frac{1}{1+|\mathcal{A}^t|} \left(\sum_{j \in \mathcal{A}^t} r_j^t\right)^2 .
\end{aligned}$$

The above equation is equivalent to the computational formula of the sum of squares over the multiset $\mathcal{Q}^t = \{r_j^t : j \in \mathcal{A}^t\} \uplus \{r_t^t\}$ with $r_t^t = 0$, defined as:

$$\mathrm{SS}(\mathcal{Q}^t) = \sum_{q \in \mathcal{Q}^t} (q - \overline{q})^2.$$

$\mathcal{Q}^t$ can be interpreted as the multiset of displacement terms associated with the coordinates modified by the surrogate (i.e., those in $\mathcal{A}^t$ and $t$) when removing the shared displacement $\tilde{z}_t^t - \hat{y}_t$.

Consider $t, t' \in \mathcal{S}$ such that

$$\hat{y}_{t'} > \hat{y}_t, \qquad d = \hat{y}_{t'} - \hat{y}_t > 0.$$

Since $\epsilon \to 0^+$, we may assume that $\epsilon < d$. The corresponding active sets satisfy

$$\mathcal{A}^{t'} \subseteq \mathcal{A}^t, \qquad t \notin \mathcal{A}^{t'}, \qquad t' \in \mathcal{A}^t.$$

Additionally, for every $j \in \mathcal{A}^{t'}$,

$$r_j^t = \hat{y}_j - \hat{y}_t + \epsilon = r_j^{t'} + d,$$

while

$$r_{t'}^t = d + \epsilon.$$

Let

$$\mathcal{P} = \mathcal{Q}^{t'} + d = \{r_j^{t'} + d : j \in \mathcal{A}^{t'}\} \uplus \{d\},$$

where $\mathcal{Q}^{t'} + d$ denotes the multiset obtained by adding $d$ to each element of $\mathcal{Q}^{t'}$. Since the sum of squared deviations is translation invariant,

$$\mathrm{SS}(\mathcal{P}) = \mathrm{SS}(\mathcal{Q}^{t'}).$$

We next relate $\mathcal{P}$ to $\mathcal{Q}^t$ by introducing a new multiset $\mathcal{R}$ obtained by removing the element $d \in \mathcal{P}$ and adding the two elements $0$ and $d + \epsilon$:

$$\mathcal{R} = (\mathcal{P} \setminus \{d\}) \uplus \{0, d + \epsilon\}.$$

The element $0$ is the residual associated with the candidate coordinate $t$, whereas $d + \epsilon = r_{t'}^t$. The goal is to show that

$$\mathrm{SS}(\mathcal{Q}^t) \geq \mathrm{SS}(\mathcal{R}) > \mathrm{SS}(\mathcal{P}) = \mathrm{SS}(\mathcal{Q}^{t'}).$$

Let $n = |\mathcal{P}|$ and $s = \sum_{q \in \mathcal{P}} q$. Since $d \in \mathcal{P}$ and all its other elements are at least $d$, we have $s \geq nd$. Using the computational formula of the sum of squares we obtain

$$\begin{aligned}\mathrm{SS}(\mathcal{R}) - \mathrm{SS}(\mathcal{P}) &= (d+\epsilon)^2 - d^2 - \frac{(s+\epsilon)^2}{n+1} + \frac{s^2}{n} \\ &= \frac{s^2 - 2n\epsilon s + 2n(n+1)d\epsilon + n^2\epsilon^2}{n(n+1)}.\end{aligned}$$

The numerator is increasing in $s$ for $s \geq n\epsilon$, and $s \geq nd > n\epsilon$ since $d > \epsilon$; hence it is minimised at $s = nd$, giving

$$\mathrm{SS}(\mathcal{R}) - \mathrm{SS}(\mathcal{P}) \geq \frac{nd^2 + 2d\epsilon + n\epsilon^2}{n+1} > 0.$$

Finally, $\mathcal{Q}^t$ is obtained from $\mathcal{R}$ by adding the residuals

$$\{r_j^t : j \in \mathcal{A}^t \setminus (\mathcal{A}^{t'} \cup \{t'\})\}.$$

Adding an element to a multiset cannot decrease its sum of squared deviations: if $\mathcal{Q}$ has cardinality $m$ and mean $\overline{q}$, then

$$\mathrm{SS}(\mathcal{Q} \uplus \{x\}) = \mathrm{SS}(\mathcal{Q}) + \frac{m}{m+1}(x - \overline{q})^2 \geq \mathrm{SS}(\mathcal{Q}).$$

Consequently,

$$\mathrm{SS}(\mathcal{Q}^t) \geq \mathrm{SS}(\mathcal{R}) > \mathrm{SS}(\mathcal{P}) = \mathrm{SS}(\mathcal{Q}^{t'}).$$

Equivalently,

$$\|\widetilde{\mathbf{z}}^{\,t'} - \widehat{\mathbf{y}}\|_2^2 < \|\widetilde{\mathbf{z}}^{\,t} - \widehat{\mathbf{y}}\|_2^2.$$

Thus, no candidate $t$ with a strictly sub-maximal value $\hat{y}_t$ can minimise the surrogate projection distance.

Notice that if $\hat{y}_t = \hat{y}_{t'}$, exchanging the coordinates $t$ and $t'$ leaves both projection distances unchanged. Hence,

$$\|\mathbf{z}^t - \hat{\mathbf{y}}\|_2^2 = \|\mathbf{z}^{t'} - \hat{\mathbf{y}}\|_2^2 \quad \text{and} \quad \|\tilde{\mathbf{z}}^{\,t} - \hat{\mathbf{y}}\|_2^2 = \|\tilde{\mathbf{z}}^{\,t'} - \hat{\mathbf{y}}\|_2^2.$$

The same candidate is therefore selected by all three operators according to the common deterministic tie-breaking policy specified in Footnote 1.

Putting our results together we obtain that:

$$\operatorname*{argmin}_{t \in \mathcal{S}} \left\|\mathbf{z}^t - \hat{\mathbf{y}}\right\|_2^2 = \operatorname*{argmin}_{t \in \mathcal{S}} \left\|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\right\|_2^2 = \operatorname*{argmax}_{t \in \mathcal{S}} \hat{y}_t.$$

□

# F Proof of Theorem 4

**Statement.** *Let $\mathbf{z}^t$ be the solution of the fixed point system* (4)*, $\tilde{\mathbf{z}}^t$ defined as in Equation* (6)*, and $\mathcal{W}^t = \mathcal{A}^t \setminus \mathcal{M}^t$. For each $j \in \mathcal{W}^t$, let $g_j = z_t^t - \epsilon - \hat{y}_j$. Then*

$$\left\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\right\|_2^2 = \sum_{j \in \mathcal{W}^t} g_j^2 - \frac{1}{1 + |\mathcal{A}^t|}\Big(\sum_{j \in \mathcal{W}^t} g_j\Big)^2. \tag{33}$$

*Proof.* Let $\mathcal{W}^t = \mathcal{A}^t \setminus \mathcal{M}^t$ denote the spuriously included indices, with cardinality $w = |\mathcal{W}^t|$. Let $m = |\mathcal{M}^t|$, $a = |\mathcal{A}^t| = m + w$. Write $M = \sum_{k \in \mathcal{M}^t} \hat{y}_k$ and $G = \sum_{j \in \mathcal{W}^t} g_j$, where $g_j = z_t^t - \epsilon - \hat{y}_j$ for $j \in \mathcal{W}^t$.

The two solutions read

$$z_t^t = \frac{\hat{y}_t + M + m\epsilon}{1+m}, \qquad \tilde{z}_t^t = \frac{\hat{y}_t + M + \sum_{k \in \mathcal{W}^t} \hat{y}_k + a\epsilon}{1+a},$$
$$z_j^t = \tilde{z}_j^t = \hat{y}_j \qquad \text{for } j \notin \mathcal{A}^t \cup \{t\},$$

$$z_j^t = z_t^t - \epsilon \qquad \tilde{z}_j^t = \tilde{z}_t^t - \epsilon \qquad \text{for } j \in \mathcal{M}^t$$
$$z_j^t = \hat{y}_j \qquad \tilde{z}_j^t = \tilde{z}_t^t - \epsilon \qquad \text{for } j \in \mathcal{W}^t.$$

We now proceed in calculating the difference for each coordinate.

Let us consider the difference at $j = t$. Since $\hat{y}_t + M = (1+m)z_t^t - m\epsilon$ we can simplify the expression for $\tilde{z}_t^t$ as

$$\begin{aligned}
\tilde{z}_t^t - z_t^t &= \frac{(1+m)z_t^t - m\,\epsilon + \sum_{k\in\mathcal{W}^t} \hat{y}_k + a\,\epsilon}{1+a} - z_t^t \\
&= \frac{(1+m)z_t^t - z_t^t(1+a) + \sum_{k\in\mathcal{W}^t} \hat{y}_k + w\,\epsilon}{1+a} \\
&= \frac{-w z_t^t + \sum_{k\in\mathcal{W}^t} \hat{y}_k + w\epsilon}{1+a} \\
&= -\frac{\sum_{k\in\mathcal{W}^t}(z_t^t - \epsilon - \hat{y}_k)}{1+a} = -\frac{G}{1+a}.
\end{aligned}$$

For $j \notin \mathcal{A}^t \cup \{t\}$ the difference is zero.

For $j \in \mathcal{M}^t$,

$$\tilde{z}_j^t - z_j^t = \left[\tilde{z}_t^t - \epsilon\right] - \left[z_t^t - \epsilon\right] = \tilde{z}_t^t - z_t^t = -\frac{G}{1+a}.$$

For $j \in \mathcal{W}^t$,

$$\tilde{z}_j^t - z_j^t = \tilde{z}_t^t - \epsilon - \hat{y}_j = \left[\tilde{z}_t^t - z_t^t\right] + \left[z_t^t - \epsilon - \hat{y}_j\right] = g_j - \frac{G}{1+a}.$$

Summing squared differences (over $\{t\}$, $\mathcal{M}^t$, and $\mathcal{W}^t$):

$$\begin{aligned}
\left\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\right\|_2^2 &= \frac{G^2}{(1+a)^2} + m \cdot \frac{G^2}{(1+a)^2} + \sum_{j\in\mathcal{W}^t} \left(g_j - \frac{G}{1+a}\right)^2 \\
&= \frac{(1+m)\,G^2}{(1+a)^2} + \sum_{j\in\mathcal{W}^t} g_j^2 - \frac{2\,G}{1+a} \sum_{j\in\mathcal{W}^t} g_j + \frac{w\,G^2}{(1+a)^2}.
\end{aligned}$$

Using $\sum_{j\in\mathcal{W}^t} g_j = G$ and $m + w = a$. The total simplifies to

$$\left\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\right\|_2^2 = \sum_{j\in\mathcal{W}^t} g_j^2 - \frac{2\,G^2}{1+a} + \frac{G^2}{1+a} = \sum_{j\in\mathcal{W}^t} g_j^2 - \frac{G^2}{1+a},$$

which is Equation (8).

□

## G Proof of Corollary 1

**Statement.** *Under the hypotheses of Theorem 4,*

$$\left\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\right\|_2 \le \left\|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\right\|_2. \tag{34}$$

*Proof.* We first show that $\mathbf{z}^t$ is the orthogonal projection of $\hat{\mathbf{y}}$ onto the affine subspace

$$V_{\mathcal{M}^t} = \left\{\mathbf{z} \in \mathbb{R}^h : z_j = z_t - \epsilon,\ j \in \mathcal{M}^t\right\}.$$

Substituting the defining constraint into the objective, the projection problem $\min_{\mathbf{z}\in V_{\mathcal{M}^t}} \frac{1}{2}\|\mathbf{z} - \hat{\mathbf{y}}\|_2^2$ reduces to

$$\min_{z_t,\,\{z_j\}_{j\notin\mathcal{M}^t\cup\{t\}}} \frac{1}{2}(z_t - \hat{y}_t)^2 + \frac{1}{2}\sum_{j\in\mathcal{M}^t}(z_t - \epsilon - \hat{y}_j)^2 + \frac{1}{2}\sum_{j\notin\mathcal{M}^t\cup\{t\}}(z_j - \hat{y}_j)^2.$$

The free coordinates decouple: $z_j = \hat{y}_j$ for $j \notin \mathcal{M}^t \cup \{t\}$, and the first-order condition in $z_t$ yields $z_t = \big(\hat{y}_t + \sum_{j\in\mathcal{M}^t} \hat{y}_j + |\mathcal{M}^t|\epsilon\big)/(1 + |\mathcal{M}^t|)$. Together with $z_j = z_t - \epsilon$ for $j \in \mathcal{M}^t$, this is exactly (15), hence $\mathbf{z}^t$ is the orthogonal projection of $\hat{\mathbf{y}}$ onto $V_{\mathcal{M}^t}$.

Since $\mathcal{M}^t \subseteq \mathcal{A}^t$, the approximate solution $\tilde{\mathbf{z}}^t$ satisfies $\tilde{z}^t_j = \tilde{z}^t_t - \epsilon$ for every $j \in \mathcal{M}^t$, hence $\tilde{\mathbf{z}}^t \in V_{\mathcal{M}^t}$. The Pythagorean identity for orthogonal projection onto an affine subspace then yields

$$\big\|\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\big\|_2^2 \;=\; \big\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\big\|_2^2 \;+\; \big\|\mathbf{z}^t - \hat{\mathbf{y}}\big\|_2^2 \;\geq\; \big\|\tilde{\mathbf{z}}^t - \mathbf{z}^t\big\|_2^2,$$

and taking square roots gives (34). □

# H Proof of Corollary 2

**Statement.** *Consider a finite and satisfiable set of signed clauses $\Pi$ and a variable ordering $Y_1; \ldots; Y_n$. Given $\hat{\mathbf{y}} = m_\theta(\mathbf{x})$ for some $\mathbf{x} \in \mathcal{X}$, let $\tilde{\mathbf{a}}$ be the assignment induced by* CASPER$(\hat{\mathbf{y}})$*, $\mathbf{a}$ be the assignment induced by the same procedure when each surrogate projection is replaced by the exact fixed-point projection, and let $\mathbf{a}^\star$ be the sequential feasible argmax assignment, $a_i^\star = \operatorname{argmax}_{t\in\mathcal{S}_i(\mathbf{a}^\star_{<i})} \hat{y}_{i,t}$. Then, for $\epsilon \to 0^+$:*

(i) $\mathbf{a} = \tilde{\mathbf{a}} = \mathbf{a}^\star$;

(ii) $\mathcal{S}_i(\mathbf{a}_{<i}) = \mathcal{S}_i(\tilde{\mathbf{a}}_{<i}) = \mathcal{S}_i(\mathbf{a}^\star_{<i}), \forall i \in [n]$.

*Proof.* We prove (i) separately for the exact and the surrogate procedures, showing in each case that the assignment produced coincides with $\mathbf{a}^\star$.

**Exact fixed-point solution.** For $k \in [n]$, let

$$P(k): \quad a_i = a_i^\star \quad \text{for all } i \leq k.$$

We prove $P(k)$ by induction on $k$, using the following claim.

*Claim.* Let $i \in [n]$. If $\mathbf{a}_{<i} = \mathbf{a}^\star_{<i}$, then $S_i(\mathbf{a}_{<i}) = S_i(\mathbf{a}^\star_{<i})$ and $a_i = a_i^\star$.

*Proof of the claim.* The feasible set $S_i(\mathbf{a}_{<i})$ is obtained by substituting the preceding assignments $\mathbf{a}_{<i}$ into $\bar{\Pi}_i$ (Section 3.2); it depends only on these discrete values and not on the projected vectors computed for $Y_1, \ldots, Y_{i-1}$. Since $\mathbf{a}_{<i} = \mathbf{a}^\star_{<i}$, it follows that $S_i(\mathbf{a}_{<i}) = S_i(\mathbf{a}^\star_{<i}) =: S_i$, which is non-empty by Lemma 2 (cf. the proof of Theorem 6). Moreover, the projection for $Y_i$ receives the unmodified input $\hat{\mathbf{y}}_i$, since the projections for earlier variables do not alter it. Two cases arise.

- If $\operatorname{argmax}_{j\in[h_i]} \hat{y}_{i,j} \in S_i$, no correction is applied (Section 3.2), so $a_i = \operatorname{argmax}_{j\in[h_i]} \hat{y}_{i,j} = \operatorname{argmax}_{t\in S_i} \hat{y}_{i,t} = a_i^\star$.
- Otherwise, by Theorem 3 applied to input $\hat{\mathbf{y}}_i$ and constraint $\{S_i : Y_i\}$, the candidate selected in Eq. (3) is $t_i^\star = \operatorname{argmax}_{t\in S_i} \hat{y}_{i,t}$. By Theorem 1, $t_i^\star$ is the unique maximiser of $\mathbf{z}^{t_i^\star}$, so $a_i = t_i^\star = a_i^\star$.

This proves the claim.

*Base case ($k = 1$).* No variables precede $Y_1$, so $\mathbf{a}_{<1} = \mathbf{a}^\star_{<1}$ holds trivially. By the claim with $i = 1$, $a_1 = a_1^\star$. Hence $P(1)$ holds.

*Inductive step.* Assume $P(k)$ holds for some $k \in [n-1]$, i.e., $\mathbf{a}_{<k+1} = \mathbf{a}^\star_{<k+1}$. By the claim with $i = k+1$, $a_{k+1} = a^\star_{k+1}$. Together with $P(k)$, this gives $P(k+1)$.

*Conclusion.* By induction, $P(n)$ holds, i.e., $\mathbf{a} = \mathbf{a}^\star$.

**Surrogate solution.** Let $\tilde{P}(k)$ be defined as $P(k)$ with $\tilde{a}_i$ in place of $a_i$. The argument above applies verbatim with $\tilde{\mathbf{a}}$ in place of $\mathbf{a}$, Eq. (6) in place of Eq. (3), $\tilde{\mathbf{z}}^{t_i^\star}$ in place of $\mathbf{z}^{t_i^\star}$, and Theorem 2 in place of Theorem 1. Theorem 3 again gives $t_i^\star = \operatorname{argmax}_{t\in S_i} \hat{y}_{i,t}$, now for $\epsilon \to 0^+$, and Theorem 2 guarantees that $t_i^\star$ is the unique maximiser of $\tilde{\mathbf{z}}^{t_i^\star}$. Hence $\tilde{\mathbf{a}} = \mathbf{a}^\star$.

Combining the two cases gives $\mathbf{a} = \mathbf{a}^\star = \tilde{\mathbf{a}}$, which is (i). For (ii), fix $i \in [n]$. By (i), $\mathbf{a}_{<i} = \tilde{\mathbf{a}}_{<i} = \mathbf{a}^\star_{<i}$, and since $S_i(\cdot)$ depends only on the preceding assignments, $S_i(\mathbf{a}_{<i}) = S_i(\tilde{\mathbf{a}}_{<i}) = S_i(\mathbf{a}^\star_{<i})$. □

## I Proof of Proposition 1

**Statement.** *Let $\hat{\mathbf{y}} \in \Delta([h])$ be such that $t^\star$ and $\mathcal{A}^{t^\star}$ in (6) are locally constant in a neighbourhood of $\hat{\mathbf{y}}$. Let $\textsc{Casper}(\hat{\mathbf{y}}) = \tilde{\mathbf{z}}^t$ and $V = \{\mathbf{v} \in \mathbb{R}^h : v_j = v_t,\ j \in \mathcal{A}^t\}$. Then the Jacobian $J = \partial\tilde{\mathbf{z}}^t/\partial\hat{\mathbf{y}}$ is the orthogonal projector $P_V$ onto $V$. Consequently, for any differentiable downstream loss $\mathcal{L}_{\text{task}}$,*

$$\nabla_{\hat{\mathbf{y}}}\big[\mathcal{L}_{\text{task}} \circ \tilde{\mathbf{z}}^t\big] \;=\; P_V\, \nabla_{\tilde{\mathbf{z}}}\mathcal{L}_{\text{task}}, \qquad \big\|P_V\, \nabla_{\tilde{\mathbf{z}}}\mathcal{L}_{\text{task}}\big\|_2 \;\le\; \big\|\nabla_{\tilde{\mathbf{z}}}\mathcal{L}_{\text{task}}\big\|_2,$$

*with the $a$-dimensional subspace $V^\perp$ annihilated entirely.*

*Proof.* We can follow the same steps as in the proof of Corollary 1 to show that $\tilde{\mathbf{z}}^t$ is the orthogonal projection of $\hat{\mathbf{y}}$ onto the affine subspace $V_{\mathcal{A}^t} = \mathbf{p}_0 + V$ for any $\mathbf{p}_0 \in V_{\mathcal{A}^t}$. Hence $\tilde{\mathbf{z}}^t(\hat{\mathbf{y}}) = \mathbf{p}_0 + P_V(\hat{\mathbf{y}} - \mathbf{p}_0)$, and differentiating gives $J = P_V$.

The chain-rule expression $\nabla_{\hat{\mathbf{y}}}[\mathcal{L}_{\text{task}} \circ \tilde{\mathbf{z}}^t] = J^\top \nabla_{\tilde{\mathbf{z}}^t}\mathcal{L}_{\text{task}} = P_V \nabla_{\tilde{\mathbf{z}}^t}\mathcal{L}_{\text{task}}$ follows since $P_V$ is symmetric, and the contraction $\|P_V\mathbf{g}\|_2 \le \|\mathbf{g}\|_2$ holds for any $\mathbf{g} \in \mathbb{R}^{[h]}$ and orthogonal projector.

The coordinate-wise formula comes from the explicit form $(P_V)_{jk} = \frac{1}{1+a}\mathbf{1}_{\{j,k\in\mathcal{A}^t\cup\{t\}\}} + \delta_{jk}\mathbf{1}_{\{j\notin\mathcal{A}^t\cup\{t\}\}}$:

$$\frac{\partial\mathcal{L}_{\text{task}}}{\partial\hat{y}_k} = \begin{cases} \partial\mathcal{L}_{\text{task}}/\partial\tilde{z}^t_k & k \notin \mathcal{A}^t \cup \{t\}, \\ \frac{1}{1+|\mathcal{A}^t|}\left(\frac{\partial\mathcal{L}_{\text{task}}}{\partial\tilde{z}^t_t} + \sum_{j\in\mathcal{A}^t}\frac{\partial\mathcal{L}_{\text{task}}}{\partial\tilde{z}^t_j}\right) & k \in \mathcal{A}^t \cup \{t\}. \end{cases}$$

□

## J Proof of Proposition 2

**Statement.** *Under the assumptions of Proposition 1, consider the composite training objective*

$$\mathcal{L}_{\text{total}}(\hat{\mathbf{y}}) = \mathcal{L}_{\text{task}}\big(\tilde{\mathbf{z}}^t(\hat{\mathbf{y}})\big) + \tfrac{\lambda}{2}\,\big\|\tilde{\mathbf{z}}^t(\hat{\mathbf{y}}) - \hat{\mathbf{y}}\big\|_2^2, \qquad \lambda > 0.$$

*Then*

$$\nabla_{\hat{\mathbf{y}}}\mathcal{L}_{\text{total}} \;=\; \underbrace{P_V\,\nabla_{\tilde{\mathbf{z}}}\mathcal{L}_{\text{task}}}_{\in V} \;+\; \underbrace{\lambda\,\big(\hat{\mathbf{y}} - \tilde{\mathbf{z}}^t\big)}_{\in V^\perp}.$$

*Proof.* The task contribution is $P_V\nabla_{\tilde{\mathbf{z}}}\mathcal{L}_{\text{task}}$ by Proposition 1. Let $\phi(\hat{\mathbf{y}}) = \frac{1}{2}\|\tilde{\mathbf{z}}^t(\hat{\mathbf{y}}) - \hat{\mathbf{y}}\|_2^2$ and apply the chain rule:

$$\nabla_{\hat{\mathbf{y}}}\phi = \frac{\partial\phi}{\partial\tilde{\mathbf{z}}^t}\frac{\partial\tilde{\mathbf{z}}^t}{\partial\hat{\mathbf{y}}} + \frac{\partial\phi}{\partial\hat{\mathbf{y}}} = (J - I)^\top\big(\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\big) = (P_V - I)\big(\tilde{\mathbf{z}}^t - \hat{\mathbf{y}}\big) = \hat{\mathbf{y}} - \tilde{\mathbf{z}}^t,$$

where the last equality uses $\hat{\mathbf{y}} - \tilde{\mathbf{z}}^t \in V^\perp$ (residual orthogonality of the projection in Proposition 1), so $P_V(\hat{\mathbf{y}} - \tilde{\mathbf{z}}^t) = \mathbf{0}$. □

## K Proof of Lemma 2

**Statement.** *Let $\Pi_i$ be a finite and satisfiable set of signed clauses in the variables $Y_1, \dots, Y_i$. Then $\Pi_i$ and $\Pi_{i-1}$ are equisatisfiable, and each assignment to $Y_1, \dots, Y_{i-1}$ satisfying $\Pi_{i-1}$ can be extended to satisfy $\Pi_i$.*

*Proof. Forward direction.* If $\Pi_i$ is satisfiable then $\Pi_{i-1}$ is satisfiable. Indeed, every clause in $\mathrm{Confl}_i(\bar{\Pi}_i)$ is derived from $\bar{\Pi}_i \subseteq \Pi_i$ via the intersection rule (Equation (9)), hence it is entailed by $\Pi_i$). Combined with $\Pi_i \setminus \bar{\Pi}_i \subseteq \Pi_i$, Equation (10) yields $\Pi_i \models \Pi_{i-1}$; in particular, if $\mathbf{a} \models \Pi_i$ then its restriction $\mathbf{a}_{<i}$ (i.e., the assignment $\mathbf{a}$ restricted over the variables $Y_1, \dots, Y_{i-1}$) satisfies $\Pi_{i-1}$, since the latter is over $Y_1, \dots, Y_{i-1}$.

*Backward direction.* Let $\bar{\Pi}_i = \{(\mathcal{S}_j : Y_i) \vee \varphi_j\}_{j=1}^m$ collect the clauses of $\Pi_i$ mentioning $Y_i$, with each $\varphi_j$ a signed clause over $Y_1, \dots, Y_{i-1}$. Let $\mathbf{a}_{<i} \models \Pi_{i-1}$, and assume for contradiction that no $a_i \in [h_i]$ satisfies $(\mathbf{a}_{<i}, a_i) \models \Pi_i$.

Substituting $\mathbf{a}_{<i}$ into the clauses of $\Pi_i$: clauses in $\Pi_i \setminus \bar{\Pi}_i$ are over $Y_1, \ldots, Y_{i-1}$ and lie in $\Pi_{i-1}$, hence are satisfied by $\mathbf{a}_{<i}$; each clause $(\mathcal{S}_j : Y_i) \vee \varphi_j \in \bar{\Pi}_i$ reduces to $(\mathcal{S}_j : Y_i)$ if $\mathbf{a}_{<i} \not\models \varphi_j$, and to a tautology otherwise. Define

$$\mathcal{K} = \{j \in [m] : \mathbf{a}_{<i} \not\models \varphi_j\}, \qquad \bar{\Pi}_i(\mathbf{a}_{<i}) \;=\; \{(\mathcal{S}_k : Y_i) : k \in \mathcal{K}\}.$$

By assumption, no $a_i$ satisfies $\bar{\Pi}_i(\mathbf{a}_{<i})$. If $\mathcal{K} = \emptyset$, any $a_i \in [h_i]$ would do, contradicting the assumption; hence $\mathcal{K} \neq \emptyset$ and $\bigcap_{k \in \mathcal{K}} \mathcal{S}_k = \emptyset$.

By Equation (11), $\mathcal{K}$ witnesses a clause

$$\psi = \bigvee_{k \in \mathcal{K}} \varphi_k \;\in\; \mathrm{Confl}_i(\bar{\Pi}_i) \;\subseteq\; \Pi_{i-1}.$$

Every $k \in \mathcal{K}$ has $\mathbf{a}_{<i} \not\models \varphi_k$ by definition of $\mathcal{K}$, so $\mathbf{a}_{<i} \not\models \psi$, contradicting $\mathbf{a}_{<i} \models \Pi_{i-1}$. Hence some $a_i \in [h_i]$ satisfies $(\mathbf{a}_{<i}, a_i) \models \Pi_i$, which proves the extension claim. Equisatisfiability follows.

□

# L Proof of Theorem 5

**Statement.** *Let $\Pi$ be a finite set of $r$ signed clauses over the variables $Y_1, \ldots, Y_n$, $k = \max_{i \in [n]} h_i$ and $w^\star(d)$ the conflict width of $\Pi$ along the ordering $d = Y_1; \ldots; Y_n$.* COMPILE$(\Pi, Y_1; \ldots; Y_n)$ *in Algorithm 1 can be implemented to run in time $\mathcal{O}((r+n) \cdot k^{w^\star(d)+1})$ and space $\mathcal{O}(n \cdot k^{w^\star(d)})$.*

*Proof.* The proof proceeds by reducing COMPILE to bucket elimination [13]. This suffices to prove the claim, as the latter has worst case time complexity linear in $r + n$ and space linear in $n$, while both grow exponentially — with base $k$ — in the *induced width* $w^\star(d)$ of the variable interaction graph associated to a signed-CNF constraint set $\Pi$, along an ordering $d$. Since in our case $\mathcal{G}^\star_\Pi(d)$ is defined as the induced graph of $\mathcal{G}_\Pi$ along $d$ in the same way as [14], then our conflict width coincides with the induced width.

In particular, we reduce the compilation step of CASPER in Equations (10) and (11) to the bucket elimination algorithm for belief updating over Bayesian networks in [14, Fig. 4.5]. The proof comprises three parts:

1. the encoding of the COMPILE routine in the bucket elimination framework of [13];
2. the reduction; and
3. the transfer of the time and space bounds from bucket elimination to COMPILE.

Together these show that COMPILE admits the same worst-case time and space complexity bounds as classical problems solvable through bucket elimination, like adaptive consistency and directional resolution [13].

*1. Relational Encoding.* Consider a set of variables $\mathbf{U} = \{Y_{j_1}, \ldots, Y_{j_m}\} \subseteq \mathbf{Y}$ with $j_1 < \ldots < j_m$ ordered along the chosen $d$; let $\mathcal{Y}_\mathbf{U} = [h_{j_1}] \times \ldots \times [h_{j_m}]$ and let $\mathbf{u} = (u_{j_1}, \ldots, u_{j_m}) \in \mathcal{Y}_\mathbf{U}$ be an assignment of the values in $\mathcal{Y}_\mathbf{U}$ to variables in $\mathbf{U}$. For a signed clause $\varphi$ so that $\mathtt{scope}(\varphi) \subseteq \mathbf{U}$, we say that $\mathbf{u}$ satisfies it, written as $\mathbf{u} \models \varphi$, if it exists a literal $\ell = (\mathcal{S}_j : Y_j) \in \varphi$ for which it holds that $u_j \in \mathcal{S}_j$. We can now encode each signed clause $\varphi$ by the relation $\mathrm{R}_\varphi = \{\mathbf{u} \in \mathcal{Y}_{\mathtt{scope}(\varphi)} \mid \mathbf{u} \models \varphi\}$. For any two relations $\mathrm{R} \subseteq \mathcal{Y}_\mathbf{U}$ and $\mathrm{R}' \subseteq \mathcal{Y}_{\mathbf{U}'}$, their natural join is the relation $\mathrm{R} \bowtie \mathrm{R}' = \{\mathbf{u} \in \mathcal{Y}_{\mathbf{U} \cup \mathbf{U}'} \mid \mathbf{u}_\mathbf{U} \in \mathrm{R}, \mathbf{u}_{\mathbf{U}'} \in \mathrm{R}'\}$, where $\mathbf{u}_\mathbf{U}$ and $\mathbf{u}_{\mathbf{U}'}$ denote the restriction of $\mathbf{u}$ to the variables in $\mathbf{U}$ and in $\mathbf{U}'$ respectively. On the other hand, for $\mathbf{V} \subseteq \mathbf{U}$, the projection of $\mathrm{R}$ onto $\mathbf{V}$ is $\pi_\mathbf{V}(\mathrm{R}) = \{\mathbf{u}_\mathbf{V} \mid \mathbf{u} \in \mathrm{R}\}$.

*2. Reduction.* Under the relational encoding defined in the previous step, the signed resolution in Equation (9) corresponds to the belief-updating algorithm BE-BEL [14, Fig. 4.5], instantiated over the Boolean relational semiring rather than the sum-product one: indeed, bucket *combination* can be expressed through the natural join $\bowtie$ (taking the role of the product), while bucket *marginalisation* is carried out through the projection operator $\pi$ (instead of the sum). These operations still define a commutative semiring, so the general bucket-elimination scheme of [13] applies. Table 5 summarizes

Table 5: Correspondence between BE-BEL [14, Fig. 4.5] and COMPILE (Algorithm 1) under the relational encoding at step 1 of the proof.

| | BE-BEL | COMPILE |
|---|---|---|
| Variables, ordering | $X_1, \ldots, X_n, \quad d = X_1; \ldots; X_n$ | $Y_1, \ldots, Y_n, \quad d = Y_1; \ldots; Y_n$ |
| Max. domain size | $k = \max_p \lvert D_p \rvert$ | $k = \max_{i \in [n]} h_i$ |
| Input functions | conditional probability table $P \in \mathbf{P}_G$ | relations $\mathrm{R}_\varphi$, $\varphi \in \Pi$ |
| Number of inputs | $r = \lvert \mathbf{P}_G \rvert$ | $r = \lvert \Pi \rvert$ |
| Combination | product $\prod$ | natural join $\bowtie$ |
| Marginalization | sum $\sum_{X_p}$ | projection $\pi_{\mathtt{scope}(\mathcal{B}_i)}$ |
| Bucket | $bucket_p$: functions with latest var. $X_p$ | $\bar{\Pi}_i$: clauses with latest var. $Y_i$ |
| Bucket scope | $S_p$ | $\mathtt{scope}(\mathcal{B}_i)$ |
| Inputs in the bucket | $\psi_p = \prod_{P \in bucket_p} P$ | $\bowtie_{\varphi \in \bar{\Pi}_i \cap \Pi} \mathrm{R}_\varphi$ |
| Messages received | $\lambda_{S_1}, \ldots, \lambda_{S_j}$ | $\mathrm{Confl}_q(\bar{\Pi}_q)$, $q > i$, filed in $\bar{\Pi}_i$ |
| Message computed | $\lambda_p = \sum_{X_p} \psi_p \cdot \prod_{l=1}^{j} \lambda_{S_l}$ | $\pi_{\mathtt{scope}(\mathcal{B}_i)}\big( \bowtie_{\varphi \in \bar{\Pi}_i} \mathrm{R}_\varphi \big) \equiv \mathrm{Confl}_i(\bar{\Pi}_i)$ |
| Message placement | latest variable in $S_p$ | latest variable in $\mathtt{scope}(\mathcal{B}_i)$ (Eq. (10)) |
| Graph | *moral* graph | *interaction* graph $\mathcal{G}_\Pi$ |
| Width parameter | *induced* width $w^*(d)$ | *conflict* width $w^\star(d)$ |

the correspondence between BE-BEL [14, Fig. 4.5] and COMPILE under the relational encoding we build: the message $\lambda_p$ computed in a bucket corresponds to the conflict clauses set $\mathrm{Confl}_i(\bar{\Pi}_i)$, while the messages $\lambda_{S_1}, \ldots, \lambda_{S_j}$ it receives correspond to the conflict clauses produced at earlier elimination steps and re-filed into $\bar{\Pi}_i$ by Equation (10).

Along the ordering $d$, the *bucket* of $Y_i$ is the set $\mathcal{B}_i = \bar{\Pi}_i \subseteq \Pi_i$ of the clauses of $\Pi_i$ in which the variable $Y_i$ appears. Since by Equation (10) the variables $Y_{i+1}, \ldots, Y_n$ have already been eliminated, then $Y_i$ is the variable occurring in every clause of $\bar{\Pi}_i$ that comes last along $d$. We call the *scope* of a bucket $\mathcal{B}_i$ the set of the variables — other than the last one — occurring in it, namely:

$$\mathtt{scope}(\mathcal{B}_i) = \big( \bigcup_{\varphi \in \bar{\Pi}_i} \mathtt{scope}(\varphi) \big) \setminus \{Y_i\}.$$

Then the per-bucket message computed by BE-BEL in the bucket of $Y_i$ — namely $\lambda_i = \sum_{Y_i} \psi_i \prod_l \lambda_{S_l}$ with $\psi_i$ being the product of input functions in the corresponding bucket and — over the Boolean semiring — $\lambda_i(\mathbf{u}) = 1$ when some value of $Y_i$ makes every function in the bucket true — becomes the relation $\pi_{\mathtt{scope}(\mathcal{B}_i)}(\bowtie_{\varphi \in \bar{\Pi}_i} \mathrm{R}_\varphi) \subseteq \mathcal{Y}_{\mathtt{scope}(\mathcal{B}_i)}$. We remark that joining the bucket for $\bar{\Pi}_i$ and projecting over its scope rules out the tuples with no consistent value for $Y_i$: by Equation (11), these are exactly the tuples falsifying some clause of $\mathrm{Confl}_i(\bar{\Pi}_i)$ (cf. the proof of Lemma 2). Therefore, the conflict set $\mathrm{Confl}_i(\bar{\Pi}_i)$ from Equation (11) can be equivalently expressed as the set of signed clauses:

$$\{ \bigvee_{Y_j \in \mathtt{scope}(\mathcal{B}_i)} ([h_j] \setminus \{u_j\}) : Y_j \mid \mathbf{u} \in \mathcal{Y}_{\mathtt{scope}(\mathcal{B}_i)} \setminus \pi_{\mathtt{scope}(\mathcal{B}_i)}(\bowtie_{\varphi \in \bar{\Pi}_i} \mathrm{R}_\varphi) \}, \tag{35}$$

containing a maximum of $k^{|\mathtt{scope}(\mathcal{B}_i)|}$ clauses. It follows that, BE-BEL can compute through its bucket-elimination procedure the conflict set in Equation (11). The result is then the same computed by COMPILE in virtue of Equation (10). BE-BEL carries out the same operation as COMPILE across the buckets: indeed, (i) its buckets partition the clauses by the variable coming last in $d$, (ii) the elimination proceeds backwards from $Y_n$ to $Y_1$, and (iii) the derived constraints are re-filed into lower bucket.

Since Lemma 2 — and therefore the correctness of CASPER — depends on $\Pi_{i-1}$ only through the assignments who satisfy it, then COMPILE can compute the set in Equation (35) in place of $\mathrm{Confl}_i(\bar{\Pi}_i)$.

*3. Cost transfer.* Eliminating variable $Y_i$ requires computing the combined bucket $\bowtie_{\varphi \in \bar{\Pi}_i} \mathrm{R}_\varphi$. This is a relation over the $|\mathtt{scope}(\mathcal{B}_i)| + 1$ variables, hence it contains at most $k^{|\mathtt{scope}(\mathcal{B}_i)|+1}$ tuples, where $k = \max_{i \in [n]} h_i$ is the maximum size of the domain of each variable. Since the complexity of BE-BEL depends only on the number of functions combined in each bucket and on the size of the combined bucket, regardless of the particular semiring of choice, the complexity bound transfers. To

conclude the proof we need to show that $|\texttt{scope}(\mathcal{B}_i)| \leq w^\star(d)$ for every $i \in [n]$. To this end, we show that, for every $i = n, \ldots, 1$ it holds that,

$$P(i): \qquad \mathcal{G}_{\Pi_i} \text{ is a subgraph of } (\mathbf{Y}, E_i),$$

with $E_i$ defined as in Equation (12), i.e., any two variables appearing together in a clause of $\Pi_i$ are adjacent in $E_i$. Notice that $P(i)$ implies $|\texttt{scope}(\mathcal{B}_i)| \leq w^\star(d)$. Indeed, for every $Y_j \in \texttt{scope}(\mathcal{B}_i)$ that appears in a clause of $\bar{\Pi}_i \subseteq \Pi_i$ together with $Y_i$, it holds that $(Y_i, Y_j) \in E_i$ by $P(i)$. Observe that $j < i$, since $Y_{i+1}, \ldots, Y_n$ do not appear in $\Pi_i$. As $E_i \subseteq E_0$, we get that $Y_j \in \mathcal{N}_d(Y_i)$, so $\texttt{scope}(\mathcal{B}_i) \subseteq \mathcal{N}_d(Y_i)$ and therefore we obtain the desired bound $|\texttt{scope}(\mathcal{B}_i)| \leq |\mathcal{N}_d(Y_i)| \leq w^\star(d)$.

We now prove $P(i)$ by induction on $i = n, \ldots, 1$. We follow the same argument used in [15, Lemma 2], showing that directional resolution never couples variables that are not adjacent in the induced graph.

*Base case ($i = n$).* For $i = n$, we have $\Pi_n = \Pi$ and $E_n = E_\Pi$. So $\mathcal{G}_{\Pi_n} = \mathcal{G}_\Pi$ and $P(n)$ holds as $\mathcal{G}_\Pi$ is the graph $(\mathbf{Y}, E_n)$.

*Inductive step ($i \to i-1$).* Assume $P(i)$ holds. By Equation (10) every clause $\varphi \in \Pi_{i-1}$ falls in one of two cases:

- $\varphi \in \Pi_i \setminus \bar{\Pi}_i$ is a carried over clause in $\Pi_{i-1}$: any two variables in $\texttt{scope}(\varphi)$ are neighbours in $E_i$ by $P(i)$, hence in $E_{i-1} \supseteq E_i$;
- $\varphi$ is a conflict clause in Equation (35), hence so that $\texttt{scope}(\varphi) \subseteq \texttt{scope}(\mathcal{B}_i)$. Any two distinct $Y_j, Y_l \in \texttt{scope}(\mathcal{B}_i)$ satisfy $j, l < i$ and $\{(Y_j, Y_i), (Y_l, Y_i)\} \subseteq E_i$, as shown above. Therefore, Equation (12) adds the edge $(Y_j, Y_l)$ to $E_{i-1}$ and any two variables in $\texttt{scope}(\varphi)$ are adjacent in $E_{i-1}$.

It follows that $\mathcal{G}_{\Pi_{i-1}}$ is a subgraph of $(\mathbf{Y}, E_{i-1})$ which proves $P(i-1)$. Since we showed that under the relational encoding we provided in step 1. COMPILE reduces to the BE-BEL algorithm, and that $|\texttt{scope}(\mathcal{B}_i)| \leq w^\star(d)$ for every $i \in [n]$, then the proof of [14, Thm. 4.5, Sec. 4.8] yields the bounds:

$$\text{time } \mathcal{O}((r+n) \cdot k^{w^\star(d)+1}) \qquad \text{and} \qquad \text{space } \mathcal{O}(n \cdot k^{w^\star(d)}).$$

□

We remark that [14] states the bound as $\mathcal{O}(r \cdot k^{w^\star(d)+1})$ under the assumption $r > n$, which holds for Bayesian networks but not necessarily for $\Pi$. Also, our result highlights a (somewhat expected) super-polynomial dependence: since COMPILE decides the satisfiability of $\Pi$ (Algorithm 1), and satisfiability of signed-CNF formulae is NP-complete [9], no implementation of COMPILE runs in time polynomial in $r$, $n$ and $k$ unless P = NP.

## M Proof of Theorem 6

**Statement.** *Let $\Pi$ be a finite and satisfiable set of signed clauses. For any datapoint $\mathbf{x}$, model $m_\theta$ and variable ordering, the corresponding output $\textsc{Casper}(m_\theta(\mathbf{x}))$ satisfies $\Pi$.*

*Proof.* By induction on the number $n$ of variables appearing in $\Pi$.

*Base case ($n = 0$):* $\Pi$ contains no variables and is trivially satisfied by any assignment.

*Inductive step ($n > 0$):* Let $Y_n$ be the variable in $\Pi$ with the highest index in the ordering, so $\Pi_n = \Pi$. Since $\Pi_{n-1}$ involves $n-1$ variables and is satisfiable, the inductive hypothesis gives that the partial assignment $\mathbf{a}_{<n}$ induced by $\textsc{Casper}(m_\theta(\mathbf{x}))$ satisfies $\Pi_{n-1}$. Substituting $\mathbf{a}_{<n}$ into $\Pi$ leaves clauses involving only $Y_n$; by Lemma 2, this residual is satisfiable and reduces to a single signed clause $(\mathcal{S}^\star : Y_n)$ by replacing each variable in $Y_1, \ldots, Y_{n-1}$ with the value given by the assignment $\mathbf{a}_{<n}$ induced by $\textsc{Casper}(m_\theta(\mathbf{x}))$. Then, Theorem 2 ensures that $\arg\max_{j \in [h_n]} \textsc{Casper}(m_\theta(\mathbf{x}))_n \in \mathcal{S}^\star$, and hence $\textsc{Casper}(m_\theta(\mathbf{x}))$ satisfies $\Pi_n = \Pi$. □

# N Additional Experimental Results

## N.1 Warcraft Pathfinding.

**Task Details** We consider a shortest-path finding task on Warcraft II terrain tiles as proposed by [41], with the dataset generated using the MIT-licensed `war2edit` toolset[3]; the underlying tile assets remain the property of Blizzard Entertainment and are used here for research purposes. The full rendered terrain image is $96\times96$ pixels at the $N$=12 version and $240\times240$ at $N$=30 one. Given a $N \times N$ grid rendered from Warcraft II terrain tiles, the goal of the task is to predict the minimum-cost paths from the top-left to the bottom-right corner. We test grid sizes, $N \in \{12, 30\}$ and each tile has a traversal cost outlined in Table 6 with higher costs corresponding to harder-to-traverse terrain. The constraint is that the predicted path forms a valid route: connected, contained within the grid, starting at the source (0,0) and ending at $(N-1, N-1)$.

Table 6: Warcraft II Tile Cost Mapping

| Label | Terrain Type | Cost |
|---|---|---|
| Land | Grass / Open Ground | 0.8 |
| Dirt | Dirt / Coast | 1.2 |
| Forest & Mountains | Forest | 5.3 |
| Forest & Mountains | Mountains | 7.7 |
| Water | Water | 9.2 |

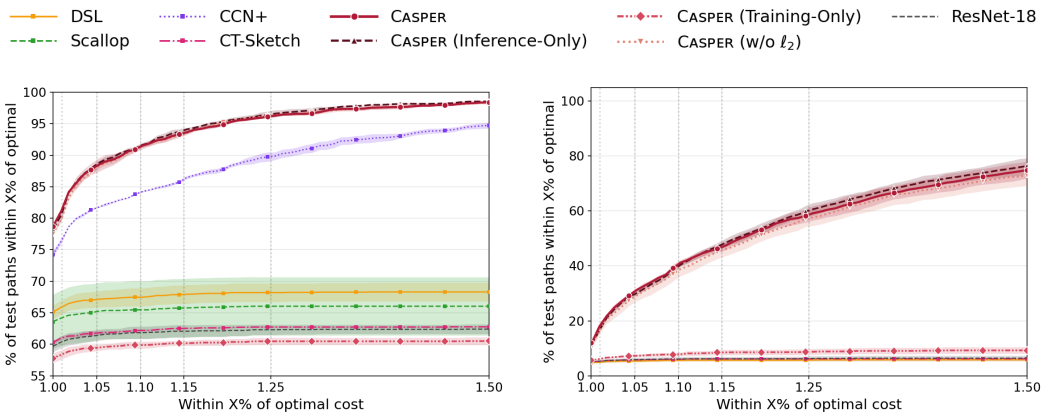
**Dijkstra-based methods.** We do not include A-NeSI [48] or I-MLE [39] among the baselines. These approaches predict a tile-cost map $c_\theta(\mathbf{x})$ and then use a task-specific shortest-path solver to return

$$\operatorname*{argmin}_{p\in\mathcal{P}} \sum_{v\in p} c_{\theta,v}(\mathbf{x}),$$

where $\mathcal{P}$ is the set of valid paths. The solver therefore performs not only feasibility enforcement but also the combinatorial optimization that selects the path, while the network learns the costs rather than directly predicting the structured output. In our formulation, as in SPL [3], the network directly predicts the path, and hence must learn from the image which of the feasible paths is optimal. The two formulations thus differ in their neural output space, optimization objective, and division of labour between the neural and symbolic components. Dijkstra-based decoding is a strong, task-specific alternative that is complementary to our approach; however, it does not address the setting studied here, in which the network must directly predict the structured output and constraints are enforced on those predictions.

**Experimental Setup.** For both grid sizes, $N \in \{12, 30\}$, we use a fixed split of 10k training, 1k validation, and 1k test examples. The neural backbone is a ResNet-18 [27] trained from scratch. We train with Adam at learning rate $5\text{e}{-4}$ under a cosine schedule ($\eta_{\min} = 5\text{e}{-6}$), batch size 128, and 300 epochs with early stopping. In addition to the metrics reported in the main text, and for coherence with previous work [3], we additionally report Hamming accuracy and Exact match. Hamming accuracy is the per-edge accuracy averaged over the grid, and Exact match is the percentage of predicted paths that match the ground-truth shortest path edge-for-edge. We further report Optimal (%), the percentage of predicted paths whose total cost equals the ground-truth cost, along with the fractions within $1\%$, $5\%$, and $10\%$ of the ground-truth cost. For methods that do not always produce consistent paths, these fractions are computed over the full test set (not solely on consistent paths): since inconsistent predictions count as failures regardless of the cost of any partial path they may induce. These metrics are reported in Tables 8 and 9 for the $12 \times 12$ and $30 \times 30$ grids, respectively. To complement the exact-optimality metric, Figure 3 (Appendix N) reports a cumulative cost-optimality analysis: for each tolerance factor $\alpha \geq 1$, we measure the fraction of predictions whose cost is within a factor $\alpha$ of the optimal path cost, providing a finer-grained view of how close non-optimal predictions are to the optimum. Consistency is the percentage of predicted paths that form a valid route: connected, contained within the grid, with correct degree at all nodes. For methods that predict a walk, all metrics are computed on the loop-erased path. All metrics are reported as mean $\pm$ std over 5 seeds.

[3] `https://github.com/war2/war2edit`




(a) $12 \times 12$ (b) $30 \times 30$

Figure 3: Cumulative path cost-optimality for Warcraft Pathfinding

**Path encoding.** CASPER and all structured baselines except CTSketch share a single step-wise encoding of the path constraint ((Appendix O)), compiled once into an artifact from which every method reads its domains and transitions. For an $N \times N$ grid with cells numbered in row-major order, source $s = 0$ and target $t = N^2 - 1$, the variables $X_0, \ldots, X_L$ with $L = N^2 - 1$ denote the cell occupied at each tick. A cell $v$ belongs to the domain $D_i$ of $X_i$ only if it is reachable from $s$ by tick $i$ ($d(s, v) \leq i$), has matching parity ($i \equiv d(s, v) \bmod 2$), and can still reach $t$ in time ($d(v, t) \leq L - i$); $X_0 = s$ and $X_L = t$ are fixed. Transitions require $X_{i+1} \in \mathrm{succ}(X_i) \cap D_{i+1}$, where $\mathrm{succ}(u)$ is the set of grid neighbours of $u$, except that $t$ can only move to itself. No all-different constraint is imposed, so walks may revisit cells; loop erasure turns a predicted walk into a simple path.

**Implementation Details.** We compare CASPER against the remaining six neurosymbolic baselines once again spanning exact and approximate methods: DeepProbLog (DPL) [36], Semantic Probabilistic Layers (SPL) [3], CCN+ [25], CTSketch [10], Scallop [30], and DeepSoftLog (DSL) [35]. Both CASPER and CCN+ enforce path connectivity through a differentiable layer built on the steps encoding. For CASPER, the per-cell probabilities $P(v \text{ on path})$ are read on the domains $D_i$; a walk selects the highest-scoring allowed successor at each step, and loop erasure turns it into a path. This path sets a target category for each cell, and each cell's distribution is projected with the smallest $\ell_2$ edit that makes its target the argmax. CCN+ instead encodes the steps constraint as propositional clauses over one Boolean atom per step and cell in $D_i$, requiring at least one cell per step and that $X_i = u$ implies one of its allowed successors (the at-most-one constraint is omitted); the clauses are compiled into a PiShield shield fed with the same scores as CASPER, and prediction is a greedy walk over the shielded scores followed by loop erasure. Scallop and DSL incorporate path validity through their respective probabilistic logic backends during training. Scallop encodes the steps constraint as a differentiable Datalog program over the relations $\texttt{at}(i, v)$, $\texttt{dom}(i, v)$ and $\texttt{nxt}(i, j)$ read from the compiled artifact, with grid edges as probabilistic input facts given by the network's per-node outputs: a walk at cell $u$ at tick $i$ can move along an edge to $v$ if $v \in D_{i+1}$, remains at $t$ once reached, and `valid_path` holds when $\texttt{at}(L, t)$ holds. Training minimizes $-\log P(\text{valid_path})$ alongside the supervised loss. Each proof is one valid walk, whose probability is the product of its edge probabilities; following the default settings, we retain only the top-3 highest-probability walks per fact. This signal is a soft, approximate penalty applied during training rather than a hard constraint. At inference, the prediction is the raw per-cell argmax, without the encoding, so nothing couples these choices into a connected route. The constraint therefore acts on the probability of a valid walk during training, not on this independent decoding, and does not transfer to inference, hence Scallop's low consistency.

SPL compiles the steps constraint into an SDD with one Boolean variable per legal (tick, from, to) move, each taking the probability of the grid edge it crosses, with sink-to-sink moves neutral; prediction uses SPL's greedy decoding through the circuit. CTSketch approximates the per-node path-connectivity constraint via a Tensor-Train (TT) decomposition of a symbolic one-hot tensor $\phi_{\text{conn}} \in \{0, 1\}^{2\times2\times2\times2\times3}$. The four binary modes of $\phi_{\text{conn}}$ encode the occupancy of the four

incident edges of a node (left, right, up, down in canonical order), and the final mode indexes three admissible routing classes: degree-zero (off-path node), degree-one (path endpoint), and degree-two (interior through-node); edge-degree configurations of three or higher receive zero weight in every class, producing a large penalty in log-space. We deploy CTSketch in two variants that differ in how the terrain image is presented to the backbone and how path-consistency is enforced.

For CASPER, we set the surrogate threshold $\epsilon = 1\mathrm{e}{-3}$ and add an auxiliary $L_2$ loss between the unprojected and projected per-cell distributions weighted by $\lambda = 1.0$, as motivated by Proposition 2. The step variables are processed in order, $X_0; \ldots; X_L$. For DSL, we embed the steps constraint in a symbolic functor [35] that runs max-product propagation over the steps encoding, one pass per tick ($L = N^2 - 1$ passes: 143 at $N = 12$ and 899 at $N = 30$): after each pass, cells outside $D_i$ are set to zero and $t$ may remain at itself, giving $P(\text{valid}) = \text{reach}_L(t)$. As for Scallop, this term enters only the training loss, and prediction is the raw per-cell argmax. CCN+ and Scallop are run at $N = 12$ only: CCN+'s clause encoding does not scale to $N = 30$, and Scallop exceeds the time-out limit at $N = 30$. For SPL, we run with a configuration consistent with [3]. Lastly, for CTSketch we run two distinct modalities: *CTSketch* and *CTSketch Grid* (cf. Table 8, 9).

*Single modality (CTSketch).* The full rendered terrain image is forwarded through the ResNet-18 backbone, yielding independent per-edge logits for all $2N(N-1)$ grid edges. The symbolic tensor $\phi_{\text{conn}}$ is factorised offline into a chain of TT cores via TT-SVD. At each of the $N^2$ grid nodes, the predicted probability $p_e$ of each incident edge is assembled into a 2-state distribution $[1 - p_e,\, p_e]$ (boundary edges absent from the grid are clamped to $[0, 1]$). These distributions are contracted with the shared TT-core chain to produce a per-node log-score vector over the three routing classes. The symbolic connectivity loss $\mathcal{L}_{\text{CTS}}$ is defined as the mean negative log-likelihood of the ground-truth routing class, determined by the node's degree in the ground-truth shortest path (zero, one, or two). The full training objective is given by $\mathcal{L} = \mathcal{L}_{\text{BCE}} + \lambda_{\text{CTS}}\, \mathcal{L}_{\text{CTS}}$, with $\lambda_{\text{CTS}} = 1.0$. No post-processing is applied at inference time; predicted paths are obtained by thresholding the per-edge sigmoid outputs.

*Grid modality (CTSketch Grid).* The terrain image is partitioned into a $3 \times 3$ grid of non-overlapping tiles, each covering $\lfloor N/3 \rfloor \times \lfloor N/3 \rfloor$ terrain nodes ($4 \times 4$ nodes per tile at $N{=}12$). The ResNet-18 backbone is applied independently to each tile and predicts only the edges whose both endpoints lie within that tile. The binary cross-entropy loss and the CTS loss $\mathcal{L}_{\text{CTS}}$ are correspondingly restricted to within-tile edges. An additional global regulariser $\mathcal{R}_{\text{global}}$, weighted by $\lambda_{\text{global}} = 1.0$, penalises cross-tile inconsistencies at evaluation time by scoring boundary-node degree compatibility between adjacent tiles. At inference time, within-tile edge predictions are thresholded, and a deterministic stitching procedure fills the missing cross-tile edges: starting from the top-left tile, the algorithm greedily identifies pairs of adjacent tiles with degree-one boundary nodes and connects them.

**CTSketch Ablation.** We study the sensitivity of both CTSketch modalities introduced above to the TT approximation rank on the $12 \times 12$ grid, evaluating ranks $r \in \{3, 10, 30\}$ with TT-SVD decomposition across 5 random seeds per configuration.

*Single modality.* The connectivity tensor $\phi_{\text{conn}}$ is a sparse binary array of shape $(2, 2, 2, 2, 3)$ whose intrinsic TT-rank is 3: a TT-SVD at $r{=}3$ already achieves a reconstruction error below $5 \times 10^{-7}$ in the $\ell_\infty$ norm, making the factorisation effectively lossless. As a direct consequence, all three rank settings yield nearly indistinguishable training dynamics and final performance. We adopt $r = 30$ as the final configuration reported in the main results, as it achieves the best aggregate performance and provides a large margin above the minimum admissible rank, ensuring numerical stability of the TT contraction throughout training.

*Grid modality.* Across all three ranks, CTSketch (Grid) yields consistently degenerate path-level outcomes: exact-match and consistency figures remain near $0.2\%$ irrespective of $r$ (cf. Table 8, 9). Within-tile edge quality is comparatively higher, as reflected by Hamming accuracies of $\approx 91.5\%$, indicating that the backbone learns meaningful per-tile edge distributions under the tile-local CTS supervision. The failure to produce valid global paths is therefore attributable not to the quality of the TT approximation—which is exact at every tested rank—but to the structural limitation the method exhibits during global reconciliation: tile-local policies are optimised independently, with no training signal governing the entry and exit points at tile boundaries, so the greedy cross-tile connector cannot reliably assemble a globally consistent shortest path.

Table 7: Computational efficiency on Warcraft Shortest Path across grid-sizes.

| | Peak Memory (GB) | | Epoch Time (s) | |
|---|---|---|---|---|
| **Method** | 12×12 | 30×30 | 12×12 | 30×30 |
| *Exact Methods* | | | | |
| DPL | T/O | T/O | T/O | T/O |
| SPL | T/O | T/O | T/O | T/O |
| CCN+ | 9.93 | T/O | 202.8 | T/O |
| *Approximate Methods* | | | | |
| Scallop | 0.32 | T/O | 1682.4 | T/O |
| DSL | 0.48 | 2.45 | 616.2 | 1072.5 |
| CTSketch | 0.71 | 1.82 | 1.98 | 17.97 |
| CTSketch (Grid) | 0.76 | 1.93 | 2.03 | 17.67 |
| CASPER (Ours) | 0.69 | 3.33 | 11.8 | 80.8 |

Table 8: Warcraft Pathfinding Task (12×12). Each metric is presented as a percentage (%). EM stands for Exact-Match.

| **Method** | **EM** | **Optimal** | | | | **Hamming** | **Consistent** |
|---|---|---|---|---|---|---|---|
| | | == | ≤ 1% | ≤ 5% | ≤ 10% | | |
| ResNet-18 | 58.5±1.4 | 59.7±1.2 | 60.4±1.2 | 61.4±1.1 | 61.8±1.2 | 98.3±0.1 | 62.5±1.0 |
| *Exact Methods* | | | | | | | |
| CCN+ | 68.6±0.8 | 74.2±0.6 | 76.5±0.5 | 81.6±0.1 | 84.2±0.2 | 97.5±0.0 | **100.0**±0.0 |
| SPL | T/O | T/O | T/O | T/O | T/O | T/O | T/O |
| DPL | T/O | T/O | T/O | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | | | | |
| CTSketch | 59.8±1.1 | 60.4±1.1 | 61.0±1.2 | 61.8±1.2 | 62.3±1.1 | 98.4±0.1 | 62.8±1.1 |
| CTSketch (Grid) | 0.2±0.1 | 0.2±0.1 | 0.2±0.1 | 0.2±0.1 | 0.2±0.1 | 91.5±0.1 | 0.2±0.1 |
| Scallop | 63.0±5.5 | 63.6±5.3 | 64.2±5.4 | 65.1±5.5 | 65.5±5.6 | **98.5**±0.1 | 66.1±5.7 |
| DSL | 64.0±1.8 | 65.1±1.3 | 65.9±1.3 | 67.1±1.6 | 67.5±1.8 | **98.5**±0.1 | 68.3±1.8 |
| *Ablation Studies* | | | | | | | |
| CASPER (Train Only) | 56.6±0.7 | 57.8±0.9 | 58.3±0.9 | 59.4±0.7 | 59.9±0.7 | 98.3±0.1 | 60.5±0.7 |
| CASPER (Inf. Only) | 73.6±0.9 | 78.3±0.8 | 80.7±1.1 | **88.4**±0.5 | **91.6**±0.3 | 98.1±0.0 | **100.0**±0.0 |
| CASPER (w/o $\ell_2$) | 72.6±0.8 | 77.5±0.6 | 80.0±0.3 | 87.8±0.4 | 91.3±0.4 | 98.1±0.0 | **100.0**±0.0 |
| CASPER | **73.9**±0.3 | **78.7**±0.3 | **81.3**±0.2 | 88.1±1.3 | 91.4±0.5 | 98.1±0.0 | **100.0**±0.0 |

**Computational Efficiency** Table 7 reports peak GPU memory and per-epoch training time for each method on both grid sizes $N \in \{12, 30\}$. CASPER's memory usage and epoch-time grow gracefully with increasing grid-size. All other exact methods, which guarantee path consistency, are unable to run at $N = 30$ without breaching the 5s per-sample time-out limit. Of the approximate methods, CTSketch is the most memory and time efficient at $N = 30$; however, as outlined in Table 9, this comes at the cost of both optimality (5.1% cost-optimal paths vs 12.1% for CASPER) and consistency (6.1% consistency vs 100% for CASPER).

### N.2 MNIST-Arithmetic

**Task Details** We consider two fully supervised tasks based on the MNIST Dataset [33] which is distributed under the CC BY-SA 3.0 license. For MNIST-Sum, we consider the problem of computing the sum of $n$ handwritten digits sampled from the dataset, and test using sums of size $n = 2^k$ for $k \in \{1, 4, 6, 8, 10\}$. Each sample consists of $n$ single-digit images $(x_1, \ldots, x_n)$ with labels

Table 9: Warcraft Pathfinding Task (30×30). Each metric is presented as a percentage (%). EM stands for Exact-Match.

| Method | EM | Optimal | | | | Hamming | Consistent |
|---|---|---|---|---|---|---|---|
| | | == | $\leq 1\%$ | $\leq 5\%$ | $\leq 10\%$ | | |
| ResNet-18 | 5.1±0.6 | 5.1±0.6 | 5.3±0.7 | 5.9±0.9 | 6.1±0.9 | 97.5±0.0 | 6.4±1.1 |
| *Exact Methods* | | | | | | | |
| CCN+ | T/O | T/O | T/O | T/O | T/O | T/O | T/O |
| SPL | T/O | T/O | T/O | T/O | T/O | T/O | T/O |
| DPL | T/O | T/O | T/O | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | | | | |
| CTSketch | 5.1±0.3 | 5.1±0.3 | 5.3±0.3 | 5.7±0.5 | 5.9±0.6 | **97.7**±0.4 | 6.1±0.6 |
| CTSketch (Grid) | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | 96.7±0.8 | 0.0±0.0 |
| Scallop | T/O | T/O | T/O | T/O | T/O | T/O | T/O |
| DSL | 4.8±0.5 | 4.8±0.5 | 4.9±0.5 | 5.4±0.6 | 5.6±0.6 | 97.6±0.1 | 5.8±0.7 |
| *Ablation Studies* | | | | | | | |
| CASPER (Train Only) | 5.7±0.2 | 5.7±0.2 | 6.1±0.2 | 7.2±0.5 | 7.8±0.8 | 97.3±0.0 | 9.4±1.6 |
| CASPER (Inf. Only) | 7.4±0.4 | 11.5±0.8 | 16.6±0.8 | 29.7±1.0 | 39.7±0.9 | 97.2±0.3 | **100.0**±0.0 |
| CASPER (w/o $\ell_2$) | 7.0±0.1 | 11.0±1.4 | 16.7±1.9 | 28.6±2.4 | 38.3±3.1 | 97.0±0.3 | **100.0**±0.0 |
| CASPER | **7.8**±0.6 | **12.1**±1.1 | **18.0**±1.3 | **30.4**±0.6 | **40.5**±1.4 | 97.1±0.2 | **100.0**±0.0 |

$(Y_1, \ldots, Y_n) \in \{0, \ldots, 9\}^n$ and sum $T \in \{0, \ldots, 9n\}$. The enforced constraint is:

$$\sum_{i=1}^{n} Y_i = T.$$

For MNIST-Add, we consider the problem of computing the sum of two $n$-digit numbers represented as sequences of handwritten digits. We test using $n \in \{2, 15, 100, 256\}$. Each sample consists of $2n$ single-digit images $(x_1, \ldots, x_{2n})$ encoding two operands, with labels $(A_1, \ldots, A_n) \in \{0, \ldots, 9\}^n$ and $(B_1, \ldots, B_n) \in \{0, \ldots, 9\}^n$, and an $(n+1)$-digit sum with labels $(S_0, S_1, \ldots, S_n) \in \{0, \ldots, 9\}^{n+1}$, where $A_1$, $B_1$, and $S_0$ denote the most significant digits. The constraint requires the predicted $(n+1)$-digit output to equal the sum of the two input numbers:

$$\sum_{i=1}^{n} A_i \cdot 10^{n-i} + \sum_{i=1}^{n} B_i \cdot 10^{n-i} = \sum_{i=0}^{n} S_i \cdot 10^{n-i}.$$

**Experimental Setup** To ensure a fair comparison with existing baselines we follow the dataset construction of [10]: 5k training and 1k test examples for every $n$ on MNIST-Sum, and $\lfloor 60000/2n \rfloor$ training and $\lfloor 10000/2n \rfloor$ test examples on MNIST-Add. To test scalability we sweep $n \in \{2, 16, 64, 256, 1024\}$ for MNIST-Sum and $N \in \{2, 15, 100, 256\}$ for MNIST-Add. The neural backbone is shared across both tasks: a LeNet-5 digit encoder which we train with Adam [31] at a learning rate of $1\text{e}{-3}$ (reduced to $3\text{e}{-4}$ for MNIST-Add at $n \in \{100, 256\}$) under a cosine schedule ($\eta_{\min} = 1\text{e}{-5}$). We set batch size 16 for MNIST-Sum and 64 for MNIST-Add, and for each task size $n$ we record mean $\pm$ std over 5 seeds for task accuracy (predicted output exactly matches ground truth) and consistency (predicted digits sum to the predicted output). We present the full set of results for MNIST-Sum in Table 10 and for MNIST-Add in Table 12; these results are visualised in Figure 2. Table 13 further breaks down MNIST-Add accuracy into operand and sum accuracy, reporting both exact-match and per-digit (macro) accuracy.

**Implementation Details** We compare CASPER against seven neurosymbolic baselines covering both exact and approximate methods: DeepProbLog (DPL) [36], Semantic Probabilistic Layers (SPL) [3], CCN+ [25], CTSketch [10], A-NeSI [48], Scallop [30], and DeepSoftLog (DSL) [35]. All methods are trained and evaluated in a fully-supervised setting but differ in how the per-digit predictions are aggregated to give the final output. For MNIST-Sum, CASPER, A-NeSI, and CCN+

Table 10: Test accuracy (%) on the MNISTSum task ($\text{sum}_n$) for $n \in \{2, 16, 64, 256, 1024\}$. T/O = timeout. The reference approximates the accuracy of a neural predictor for MNIST images with 0.995 accuracy, computed as $0.995^n$.

| **Method** | $\textbf{sum}_2$ | $\textbf{sum}_{16}$ | $\textbf{sum}_{64}$ | $\textbf{sum}_{256}$ | $\textbf{sum}_{1024}$ |
|---|---|---|---|---|---|
| *Exact Methods* | | | | | |
| DPL | $97.2_{\pm 0.5}$ | T/O | T/O | T/O | T/O |
| SPL | $96.7_{\pm 0.5}$ | T/O | T/O | T/O | T/O |
| CCN+ | $\mathbf{97.4}_{\pm 0.7}$ | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | | |
| A-NeSI | $96.0_{\pm 0.2}$ | $27.8_{\pm 4.6}$ | $12.9_{\pm 1.6}$ | $5.1_{\pm 0.1}$ | $2.0_{\pm 0.2}$ |
| Scallop | $95.5_{\pm 1.1}$ | $81.7_{\pm 3.6}$ | T/O | T/O | T/O |
| DSL | $97.3_{\pm 0.8}$ | $5.5_{\pm 0.9}$ | T/O | T/O | T/O |
| CTSketch | $95.0_{\pm 1.1}$ | $82.4_{\pm 0.4}$ | $51.7_{\pm 1.3}$ | $8.4_{\pm 0.3}$ | $2.3_{\pm 0.1}$ |
| CASPER (*cvxpy*) | $95.5_{\pm 0.6}$ | $84.4_{\pm 0.5}$ | $\mathbf{54.0}_{\pm 2.5}$ | $10.6_{\pm 2.0}$ | T/O |
| CASPER (Ours) | $95.9_{\pm 0.5}$ | $\mathbf{84.5}_{\pm 1.1}$ | $52.9_{\pm 2.3}$ | $\mathbf{11.8}_{\pm 1.9}$ | $\mathbf{2.4}_{\pm 0.7}$ |
| Reference | 99.0 | 92.3 | 72.6 | 27.7 | 0.6 |

use a learned MLP sum-head, while SPL reads the sum directly from the joint distribution over digits and sum encoded in its compiled SDD. CTSketch, Scallop, DSL, and DPL take a per-digit argmax at inference and compute the sum deterministically, so their predictions are consistent by construction. For MNIST-Add, CASPER and CTSketch use a carry-chain decomposition, processing digit positions sequentially and propagating the carry from each position into the next. Consistent with [48], A-NeSI uses an autoregressive MLP head over the multi-digit output. CCN+ uses a MLP head, as its binary one-hot representation does not support carry-chain decomposition. Scallop, DPL, and DSL evaluate the addition relation as a logic program, while SPL compiles it into a SDD over the joint distribution of digits and sum.

For MNIST-Sum, Scallop uses top-3 proofs with learning rate $1\text{e}{-3}$; CTSketch uses tensor-train decomposition with rank 2 and tree depth $1, 4, 6, 8, 10$ for $n = 2, 16, 64, 256, 1024$ respectively; DSL uses neural embeddings for $n = 16$ with learning rate $1\text{e}{-3}$ and weight decay $1\text{e}{-5}$; and A-NeSI uses parameters consistent with [48]. For MNIST-Add, Scallop again uses top-3 proofs with learning rate $1\text{e}{-3}$; CTSketch uses tensor-train decomposition with rank 2; and A-NeSI and DSL use the hyperparameters from their respective papers. The original CTSketch implementation reconstructs predicted digits via base-10 expansion over int64, we observe overflow for $n \geq 19$ and report results using a fixed arbitrary-precision implementation. A-NeSI's symbolic function overflows int64 for $n \geq 19$; we therefore report A-NeSI only up to $n = 15$.

Table 11 reports an ablation conducted on the MNIST-Sum task to isolate the contribution of training with CASPER versus applying it only at inference: digit accuracy is essentially unchanged across configurations, confirming that CASPER does not degrade the perception backbone, while inference-only application already recovers most of the sum-accuracy gain.

**Computational Efficiency** Table 14 reports peak GPU memory and per-epoch training time on MNIST-Sum across all scales. Among exact methods, only CASPER scales beyond $n = 2$, and it reaches $n = 1024$ while remaining competitive with the approximate methods on both memory and time, using substantially less memory than A-NeSI and running considerably faster at the largest scale. CTSketch is the most resource-efficient baseline, but at the cost of accuracy (Table 10). The cvxpy variant times out at $n = 1024$, confirming that the closed-form surrogate is necessary for scaling while retaining the constraint satisfaction guarantee.

### N.3 Sudoku solving.

**Task Details** We consider the sudoku dataset from SatNet [50], consisting of $10{,}000$ nine-by-nine Sudoku boards whose hint cells are rendered as $28\times28$ grayscale MNIST digits and whose blank cells are represented as uniformly grey images. The average number of given hints per board is $36.22$ (with

Table 11: MNIST-Sum Ablation Study.

| Metric | No CASPER | CASPER (Inference) | CASPER (Train + Inference) |
|---|---|---|---|
| $n = 16$ | | | |
| Digit Acc. (%) | 98.8±0.1 | 98.9±0.2 | **99.0**±0.1 |
| Sum Acc. (%) | 4.3±0.6 | 84.0±2.9 | **85.8**±1.6 |
| Consistency (%) | 4.3±0.6 | **100.0**±0.0 | **100.0**±0.0 |
| Avg. $L_2$ Edit | N/A | 0.9 | 0.3 |
| $n = 64$ | | | |
| Digit Acc. (%) | 98.9±0.0 | **98.8**±0.1 | **98.8**±0.0 |
| Sum Acc. (%) | 1.7±0.6 | **49.0**±1.9 | 47.4±0.2 |
| Consistency (%) | 1.7±0.2 | **100.0**±0.0 | **100.0**±0.0 |
| Avg. $L_2$ Edit | N/A | 0.8 | 0.4 |
| $n = 256$ | | | |
| Digit Acc. (%) | **98.7**±0.1 | **98.7**±0.0 | 98.5±0.4 |
| Sum Acc. (%) | 0.8±0.1 | 7.8±0.2 | **10.3**±1.1 |
| Consistency (%) | 0.7±0.4 | **100.0**±0.0 | **100.0**±0.0 |
| Avg. $L_2$ Edit | N/A | 0.9 | 0.7 |

Table 12: Test accuracy (%) on the MNISTAdd task (add$_n$). Accuracy is the exact-match accuracy of the predicted sum: all $n+1$ digits must be correct. T/O = timeout; ERR = numerical error. The reference approximates the accuracy of a neural predictor with 0.995 accuracy per digit, computed as $0.995^{2n}$.

| Method | add$_1$ | add$_2$ | add$_4$ | add$_{15}$ | add$_{100}$ | add$_{256}$ |
|---|---|---|---|---|---|---|
| *Exact Methods* | | | | | | |
| DPL | 97.1±0.2 | 89.5±0.6 | T/O | T/O | T/O | T/O |
| SPL | 96.7±0.6 | T/O | T/O | T/O | T/O | T/O |
| CCN+ | 98.4±0.2 | T/O | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | | | |
| A-NeSI | 97.4±0.2 | 96.3±0.4 | 93.1±0.7 | 76.9±2.3 | ERR | ERR |
| Scallop | 95.3±0.7 | 95.9±0.4 | T/O | T/O | T/O | T/O |
| DSL | 98.1±0.2 | 96.6±0.4 | 92.6±0.7 | **77.8**±2.2 | **28.0**±6.4 | T/O |
| CTSketch | 98.1±0.2 | 96.8±0.2 | 92.6±1.8 | 72.5±2.3 | 8.5±4.9 | 0.0±0.0 |
| CASPER (*cvxpy*) | **98.5**±0.1 | 97.2±0.2 | 93.3±0.3 | 71.8±1.6 | T/O | T/O |
| CASPER (Ours) | 98.4±0.1 | **97.3**±0.2 | **93.7**±0.7 | 73.7±0.3 | 13.8±1.9 | **0.6**±0.5 |
| Reference | 99.0 | 98.0 | 96.1 | 86.0 | 36.7 | 7.7 |

a minimum of 31, and maximum 42). The task is to predict the complete $9\times9$ solution, inserting one digit from $\{1,\ldots,9\}$ per cell—from the 81-cell image grid, while abiding by the standard Sudoku rules: a board is accepted as correct only when every row, every column, and every $3\times3$ sub-grid contains each digit exactly once. The dataset can be downloaded from the official repository. No explicit license for the dataset was provided.

**Experimental Setup** The 10,000 boards are split into 9,000 training, 500 validation, and 500 test examples following the standard SatNet partition. All hyperparameter and checkpoint selection is performed on the validation split; final results are reported on the fixed test split. Every method is run with 10 random seeds and we report mean and standard deviation across seeds for each metric we consider. The primary metric is *PipeAcc*: the fraction of test boards for which the complete pipeline output exactly matches the ground-truth solution. We additionally report the *NeuroAcc*: board accuracy of the argmax of the soft board distribution $P\in\mathbb{R}^{81\times9}$ produced by component (ii) *before* masking and symbolic completion, isolating the contribution of the learned upstream components from that of the Prolog solver. Finally, we report the percentage of CNN errors, that the board solver

Table 13: Operand and sum accuracy (%) on MNISTAdd. *Exact*: no digit errors; *Macro*: per-digit accuracy. ▮ exact, ▮ approximate methods; unlisted methods time out (A-NeSI: ERR for $n \geq 100$).

| | | Operand 1 | | Operand 2 | | Sum | |
|---|---|---|---|---|---|---|---|
| $n$ | **Method** | **Exact** | **Macro** | **Exact** | **Macro** | **Exact** | **Macro** |
| **add$_1$** | DPL | 98.4±0.2 | 98.4±0.2 | 98.5±0.2 | 98.5±0.2 | 97.1±0.2 | 97.9±0.2 |
| | SPL | 98.5±0.4 | 98.5±0.4 | 98.1±0.4 | 98.1±0.4 | 96.7±0.6 | 97.6±0.4 |
| | CCN+ | **99.2**±0.1 | **99.2**±0.1 | 99.0±0.1 | 99.0±0.1 | 98.4±0.2 | **98.9**±0.1 |
| | A-NeSI | 98.6±0.2 | 98.6±0.2 | 98.7±0.2 | 98.7±0.2 | 97.4±0.2 | 98.2±0.2 |
| | Scallop | 97.7±0.5 | 97.7±0.5 | 97.5±0.5 | 97.5±0.5 | 95.3±0.7 | 96.6±0.5 |
| | DSL | 99.0±0.1 | 99.0±0.1 | 99.1±0.1 | 99.1±0.1 | 98.1±0.2 | 98.6±0.1 |
| | CTSketch | 99.1±0.1 | 99.1±0.1 | 99.0±0.1 | 99.0±0.1 | 98.1±0.2 | 98.6±0.1 |
| | CASPER (*cvxpy*) | **99.2**±0.1 | **99.2**±0.1 | **99.2**±0.2 | **99.2**±0.2 | **98.5**±0.1 | **98.9**±0.1 |
| | CASPER (Ours) | **99.2**±0.1 | **99.2**±0.1 | **99.2**±0.1 | **99.2**±0.1 | 98.4±0.1 | 98.8±0.0 |
| **add$_2$** | DPL | 94.9±0.4 | 97.4±0.2 | 94.2±0.5 | 97.0±0.2 | 89.5±0.6 | 95.2±0.3 |
| | A-NeSI | 98.0±0.3 | 99.0±0.1 | 98.2±0.3 | 99.1±0.1 | 96.3±0.4 | 98.3±0.2 |
| | Scallop | 98.2±0.3 | 99.1±0.1 | 97.6±0.3 | 98.8±0.2 | 95.9±0.4 | 98.2±0.2 |
| | DSL | 98.4±0.2 | 99.2±0.1 | 98.2±0.3 | 99.1±0.1 | 96.6±0.4 | 98.6±0.2 |
| | CTSketch | 98.2±0.1 | 99.1±0.1 | 98.6±0.2 | **99.3**±0.1 | 96.8±0.2 | 98.4±0.1 |
| | CASPER (*cvxpy*) | 98.5±0.2 | **99.3**±0.1 | 98.4±0.1 | 99.2±0.1 | 97.2±0.2 | **98.8**±0.1 |
| | CASPER (Ours) | **98.6**±0.2 | **99.3**±0.1 | **98.7**±0.1 | **99.3**±0.1 | **97.3**±0.2 | 98.7±0.1 |
| **add$_4$** | A-NeSI | 96.2±0.5 | 99.0±0.1 | 96.7±0.5 | **99.2**±0.1 | 93.1±0.7 | 98.0±0.2 |
| | DSL | 95.8±0.6 | 98.9±0.1 | 96.6±0.5 | **99.2**±0.1 | 92.6±0.7 | 97.8±0.2 |
| | CTSketch | 96.0±1.1 | 99.0±0.3 | 96.5±0.6 | 99.1±0.1 | 92.6±1.8 | 97.9±0.4 |
| | CASPER (*cvxpy*) | 96.2±0.6 | 99.0±0.2 | 96.1±0.8 | 99.0±0.2 | 93.3±0.3 | 98.0±0.0 |
| | CASPER (Ours) | **96.6**±0.3 | **99.1**±0.1 | **97.0**±0.7 | **99.2**±0.2 | **93.7**±0.7 | **98.1**±0.3 |
| **add$_{15}$** | A-NeSI | 87.1±1.9 | 99.0±0.1 | **87.4**±1.8 | **99.1**±0.1 | 76.9±2.3 | 97.4±0.3 |
| | DSL | **88.9**±1.7 | **99.2**±0.1 | 86.5±1.9 | 99.0±0.1 | **77.8**±2.2 | **97.7**±0.3 |
| | CTSketch | 84.8±3.2 | 98.9±0.2 | 85.4±0.2 | 99.0±0.1 | 72.5±2.3 | 97.2±0.2 |
| | CASPER (*cvxpy*) | 85.8±0.7 | 99.0±0.1 | 82.7±1.2 | 98.7±0.1 | 71.8±1.6 | 97.0±0.2 |
| | CASPER (Ours) | 88.6±1.0 | **99.2**±0.0 | 86.0±0.6 | 99.0±0.0 | 73.7±0.3 | 97.6±0.1 |
| **add$_{100}$** | DSL | **38.0**±6.8 | 98.7±0.2 | **46.0**±7.1 | 98.7±0.2 | **28.0**±6.4 | 96.6±0.4 |
| | CTSketch | 30.5±0.7 | 98.9±0.0 | 30.8±4.6 | 98.9±0.1 | 8.5±4.9 | 96.9±0.3 |
| | CASPER (Ours) | 37.9±2.6 | **99.0**±0.0 | 39.4±1.8 | **99.0**±0.1 | 13.8±1.9 | **97.2**±0.2 |
| **add$_{256}$** | CTSketch | 4.6±2.2 | **98.8**±0.0 | 5.7±0.7 | **98.9**±0.0 | 0.0±0.0 | **96.9**±0.1 |
| | CASPER (Ours) | **8.9**±3.0 | **98.8**±0.2 | **6.1**±0.6 | 98.8±0.2 | **0.6**±0.5 | 96.6±0.6 |

is able to detect and recover from, out of the total number of CNN errors across the test set (*RecRate*). Results are shown in Table 4.

At evaluation time, methods that produce a scalar mask-logit bias (CASPER and its LLM variant) select the bias from the grid $\{-2.0, -1.0, 0.0, 1.0, 2.0\}$ that maximises board accuracy on the validation set; for the remaining methods (CTSketch and A-NesI) the bias is swept over $\{-1.0, 0.0, 1.0\}$ and the results for the best performing evaluation run is reported.

### Implementation Details

**Shared backbones.** All six methods share the same three-component backbone, initialised from the publicly released NASR checkpoint [11]. Specifically these are as follows:

- The *perception module* is a convolutional network that processes each of the 81 cell images independently and outputs per-cell log-posteriors over the ten classes $\{0, 1, \ldots, 9\}$ (class 0 denotes a blank cell).

Table 14: Computational efficiency on MNIST-Sum across methods and scales.

| | Peak Memory (MB) | | | | | Epoch Time (s) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Method /** $n$ | 2 | 16 | 64 | 256 | 1024 | 2 | 16 | 64 | 256 | 1024 |
| *Exact Methods* | | | | | | | | | | |
| DPL | 20.1 | T/O | T/O | T/O | T/O | 58.4 | T/O | T/O | T/O | T/O |
| SPL | 39.8 | T/O | T/O | T/O | T/O | 4.1 | T/O | T/O | T/O | T/O |
| CCN+ | 56.7 | T/O | T/O | T/O | T/O | 26.9 | T/O | T/O | T/O | T/O |
| *Approximate Methods* | | | | | | | | | | |
| A-NeSI | 80.0 | 209.5 | 1014.1 | 5680.1 | 13157.0 | 11.8 | 24.7 | 66.4 | 232.3 | 995.8 |
| Scallop † | **18.9** | 57.1 | T/O | T/O | T/O | 3100.0 | 3400.0 | T/O | T/O | T/O |
| DSL | 20.6 | 35.4 | T/O | T/O | T/O | 33.3 | 386.6 | T/O | T/O | T/O |
| CTSketch | 19.0 | **28.6** | **106.1** | **367.7** | **1421.0** | **0.6** | **4.2** | **14.9** | **60.9** | **239.2** |
| CASPER (*cvxpy*)‡ | 20.3 | 31.7 | 123.7 | 529.4 | T/O | 33.7 | 94.8 | 257.6 | 815.2 | T/O |
| CASPER(Ours) | 20.3 | 31.7 | 123.7 | 529.4 | 1866.5 | 1.5 | 5.9 | 17.5 | 62.3 | 240.3 |

† Scallop runs on CPU. ‡ The *cvxpy* projection runs on CPU

- The *board-solver* is a Vision Transformer with embedding dimension 192, depth 4, 3 attention heads, and MLP expansion ratio 4; it maps the 81 per-cell ten-dimensional inputs to a soft board distribution $P \in \mathbb{R}^{81\times 9}$ over $\{1,\ldots,9\}$.
- The *masking network* shares the same Transformer configuration with a scalar output per cell; sigmoid-activating those logits yields the per-cell keep probability used to gate Prolog.

**LLM fine-tuning.** For the LLM-backed variants, we fine-tune Qwen2.5-14B-Instruct on the 9,000 SatNet training boards using LoRA [52] with rank $r = 16$, scaling $\alpha = 16$, no dropout, applied to all attention and MLP layers. The base model is loaded in 4-bit NormalFloat quantisation with bfloat16 activations (via Unsloth). We train for 3 epochs with AdamW-8bit, learning rate $2\times10^{-4}$, cosine decay schedule, warmup ratio 0.05, weight decay 0.01, and an effective batch size of 16 (providing two examples per device with eight gradient-accumulation steps). Each board is formatted under the Qwen2.5 chat template: the system message states the 'ALLDIFFERENT' rules and specifies a structured output format; the user message provides the full $9\times9$ grid with zeros marking blank cells. The model is trained to output a `<WORKING>/<SOLUTION>` block containing exactly nine lines of nine space-separated digits. At inference time, outputs that violate the format, contain out-of-range digits, or alter any given cell are treated as failures.

The fine-tuned LLM produces one-hot-like logits—high confidence on the predicted digit and zero elsewhere—carrying no per-cell uncertainty. To recover calibrated soft probabilities compatible with downstream masking, the frozen neuro-solver from [11] is used as an auxiliary uncertainty prior for CASPER. Concretely, we introduce a new *fusion head*, implemented as a residual MLP (with a total of ≈21,700 parameters) operating on the individual cells receiving both the LLM-solved boards and the frozen standard neuro-solver soft probabilities, modelling prior prediction uncertainty. Across all LLM-backed methods the CNN and neuro-solver are frozen throughout; only the fusion head (and, where applicable, the masking network) receive gradient updates.

**CASPER.** *Neuro-board solver from [11].* Only the masking network is trained; the CNN and board solver remain at their pre-trained weights throughout. At each training step, the solver distribution is projected through CASPER constraint layer. The masking network is trained with a per-cell weighted binary cross-entropy loss: the target for cell $j$ is 1 if and only if the projected digit equals the ground-truth, and 0 otherwise. Incorrectly projected cells (the minority class, typically $<10\%$) receive loss weight 20 to compensate for the severe class imbalance. We use AdamW with learning rate $10^{-4}$, weight decay 0.3, cosine LR schedule, gradient clipping at norm 1.0, batch size 200, up to 10 epochs, and early stopping on validation board accuracy at patience 5.

*LLM board solver.* The fusion head and masking network are trained jointly. Training is cache-only: a full-data generation pass is run once to populate an on-disk cache of LLM outputs. The loss combines the same weighted BCE mask term ($\lambda_{\text{mask}} = 1.0$) and a label-smoothed NLL calibration loss on the

fused distribution ($\lambda_{\text{temp}} = 0.5$, smoothing $\alpha = 0.3$). Label smoothing is necessary because the LLM is already highly accurate: without it, the NLL drives the fusion temperature toward zero, collapsing the soft distribution. With $\alpha = 0.3$ the analytical optimum lies at a finite temperature, preserving uncertainty. All other hyperparameters match the previous variant, with the training budget extended to 20 epochs and early stopping patience 7. The mask logit bias is tuned at evaluation on the grid $\{-2.0, -1.0, 0.0, 1.0, 2.0\}$.

**CTSketch.** *Neuro-board solver from [11].* The full model (CNN, board-solver, masking network) is trained end-to-end. The training objective is the CTS tensor-sketch loss, following the code from [10]: a differentiable rank-2 Tucker approximation of the nine-ary ALLDIFFERENT constraint-satisfaction probability, summed over all 27 groups (rows, columns, and boxes) and combined with a Prolog-based per-board reward signal. We use AdamW with learning rate $5 \times 10^{-6}$, weight decay $0.3$, a linear learning-rate ramp over all $9$ training epochs (warmup covering the full training duration), and batch size $200$.

*LLM board solver.* Only the fusion head and masking network are trained; the CNN, board-solver, The CTS sketch loss and Prolog reward are applied directly to the fused distribution output by the fusion head. We use AdamW with learning rate $10^{-4}$, weight decay $0.3$, cosine LR schedule, gradient clipping (norm $1.0$), batch size $200$, $20$ epochs, and early stopping on validation reward at patience $7$.

**A-NesI.** *Neuro-board solver from [11].* We apply the A-NesI framework [48] with pairwise encoding of the ALLDIFFERENT constraints. The inference model $q$ is a two-layer MLP with hidden size $128$ that predicts binary constraint satisfaction for each cell pair; it is optimised with Adam at learning rate $10^{-3}$ (MSE loss). The perception backbone (CNN and board-solver) is updated via the log-$q$ perception loss with Adam at learning rate $5 \times 10^{-5}$. Each training step draws $100$ samples from the board distribution $P$. The masking network is frozen during A-NesI training and is used solely at evaluation time. We train for $3$ epochs with batch size $50$.

*LLM board solver.* The fusion head takes the role of the perception backbone in the A-NesI training loop: it is updated via the log-$q$ perception loss with Adam at learning rate $5 \times 10^{-5}$, while $q$ is trained identically to the neuro counterpart (Adam, lr $10^{-3}$, MSE). The masking network is trained jointly from epoch $1$ via a separate AdamW optimiser (learning rate $10^{-4}$, weight decay $0.3$, gradient clipping at norm $1.0$). The checkpoint selection uses A-NesI inference success (fraction of validation batches for which $q$ predicts all pairwise constraints satisfied) as the criterion. Training runs for $20$ epochs with early stopping patience $7$.

## O Choice of the Encoding

Theorem 5 bounds the cost of COMPILE by

$$\text{time } \mathcal{O}\big((r+n) \cdot k^{w^\star(d)+1}\big), \qquad \text{space } \mathcal{O}\big(n \cdot k^{w^\star(d)}\big),$$

with $r$ the number of clauses, $n$ the number of variables, $k = \max_i h_i$ the largest domain, and $w^\star(d)$ the conflict width of $\Pi$ along the ordering $d$, which coincides with the induced width of the interaction graph of $\Pi$ along $d$ [14]. For highly interconnected problems, $w^\star(d)$ grows with the problem size $N$, which is intractable for compilation, so we instead consider alternative problem settings that keep $w^\star(d)$ bounded.

In Sudoku solving, compiling the joint constraint set over the whole board is intractable, because every cell shares a group with many others, so the interaction graph is highly interconnected. We instead enforce the all-different constraint on each of the 27 groups (9 rows, 9 columns and 9 boxes) individually, so that each constraint set involves only the 9 cells of one group.

In Warcraft, specifying the task as path-finding with local constraints between neighbouring cells gives an interaction graph where $w^\star(d) \propto N$, making intractable to compile. We instead formulate it as walk-finding, describing the route as a sequence of cells, which gives $w^\star(d) = 1$ for every $N$. This is a different problem-setting since walks may revisit a cell, so a simple path is recovered by loop erasure. All baselines are run in the same walk-finding setting (excl. CTSketch), with loop erasure applied to every method's output before computing the metrics.

**Warcraft: walk-finding formulation.** Let $G = (V, E)$ be the four-neighbour grid graph on a $g \times g$ board, $n = g^2 = |V|$, with source $s$ (top-left) and target $t$ (bottom-right). Write $d(u, v)$ for

the graph (Manhattan) distance. Recall that $G$ is bipartite under the checkerboard two-colouring: every edge joins cells of different colours, hence a walk of length $\ell$ from $s$ can only end on a cell $v$ whose distance $d(s, v)$ has the same parity as $\ell$ ($\ell \equiv d(s, v) \pmod 2$; e.g. a cell at even distance is reachable only in an even number of moves).

**Variables and transitions.** Fix the horizon $L = n - 1$, the maximum number of edges of any simple path. Introduce multi-valued variables $X_0, \ldots, X_L$ with intended reading "$X_i$ = the cell occupied at time $i$". The *absorbing transition relation* is

$$T(u, v) \iff \big(u \neq t \wedge \{u, v\} \in E\big) \ \vee \ u = v = t,$$

i.e. every step is a real move, except that the target, once reached, repeats forever (and can never be left). (There is no general waiting move; this is what makes the parity argument below available.) The formula is the chain

$$\Phi \ = \ \bigwedge_{i=0}^{L-1} T(X_i, X_{i+1}),$$

endpoint pinning being absorbed into the domains below.

**From the relation to signed clauses.** $T$ is not a primitive of the constraint language; it is unrolled value by value. Writing $[S]{:}X$ for the signed literal "$X \in S$", each conjunct $T(X_i, X_{i+1})$ becomes one clause per value $u \in D_i$,

$$[\, D_i \setminus \{u\}\,]{:}X_i \ \vee \ [\,\mathrm{succ}(u) \cap D_{i+1}\,]{:}X_{i+1}, \qquad \mathrm{succ}(u) = \begin{cases} N(u) & u \neq t, \\ \{t\} & u = t, \end{cases}$$

i.e. the implication $X_i{=}u \Rightarrow X_{i+1} \in \mathrm{succ}(u)$ written as a disjunction. The edge set $E$ never appears as an object: adjacency ($N(u)$ = the up/down/left/right cells of $u$) is materialized inside the enumerated successor lists. A guard $[\emptyset]{:}X_i$ (empty when $D_i = \{u\}$ is a singleton) is dropped, leaving a unary clause. For the trimmed $2 \times 2$ instance the whole formula is four clauses:

$$[1, 2]{:}X_1, \qquad [2]{:}X_1 \vee [3]{:}X_2, \qquad [1]{:}X_1 \vee [3]{:}X_2, \qquad [3]{:}X_3,$$

(step 0: $D_0 = \{0\}$ forces $X_1 \in N(0) \cap D_1 = \{1, 2\}$; step 1: from $u = 1$ or $u = 2$ the only admissible successor in $D_2 = \{3\}$ is the target; step 2: absorption). Optionally, AllDifferent clauses over the non-target positions restrict the models to simple paths; we omit them (`allow_cycles`) because they connect every pair of step variables and destroy the chain structure — simplicity is instead recovered downstream (loop erasure) or left to the learner.

**Domain trimming.** The untrimmed encoding gives every $X_i$ the full domain $V$. The trimmed encoding declares instead

$$D_i \ = \ \big\{\, v \in V \setminus \{t\} \ : \ \underbrace{d(s, v) \leq i}_{\text{forward}}, \ \underbrace{i \equiv d(s, v) \pmod 2}_{\text{parity}}, \ \underbrace{d(v, t) \leq L - i}_{\text{backward}} \,\big\} \cup \big\{\, t \ : \ i \geq d(s, t) \,\big\}.$$

In particular $D_0 = \{s\}$ and $D_L = \{t\}$: the endpoint constraints are domain facts, not clauses.

**Reading the three tests.** Since $D_L = \{t\}$, the walk must have *arrived by the deadline* $L$; at time $i$ it therefore has exactly $L - i$ transitions left. The three tests bound where it can possibly stand at that moment:

- *Forward* ($d(s, v) \leq i$): the past constrains the present — in $i$ moves from $s$ no walk outruns the graph distance, so cells farther than $i$ from the source are unoccupiable.
- *Backward* ($d(v, t) \leq L - i$): the future constrains the present — from $v$ even the most direct route to $t$ costs $d(v, t)$ moves, so a cell with $d(v, t) > L - i$ admits *no* completion by the deadline, whatever the walk does afterwards. (At $i = L$ this test alone yields $D_L = \{t\}$; one step earlier only $t$ and its neighbours survive.)
- *Parity* ($i \equiv d(s, v) \bmod 2$): within reach is not enough — every move flips the checkerboard colour, so the step count and the distance must also agree modulo 2.

The forward and backward tests are the two cones of a discrete space–time "light cone" — reachable-from-$s$ and can-still-reach-$t$ — and $D_i$ is their intersection restricted to the correct colour class. Domains are accordingly narrow near both ends of the horizon and widest in the middle. Deleting a cell that fails any test removes no walk, since every walk satisfies all three by the prefix/suffix argument of Proposition 3; the backward family is also precisely what bucket elimination would otherwise have to derive as resolvents (see below), while the forward and parity families are stronger facts that the target-first elimination would never produce.

**Proposition 3** (Exactness of the trimming)**.** *An assignment $(x_0, \dots, x_L) \in D_0 \times \cdots \times D_L$ satisfies $\Phi$ if and only if it is an absorbing $s$–$t$ walk: a four-neighbour walk that starts at $s$, first reaches $t$ at some time $\ell \le L$, and repeats $t$ thereafter. Moreover the trimming deletes no such walk: the models of the trimmed formula and of the untrimmed formula (with absorbing transitions and pinned endpoints) coincide. Consequently* every *simple $s$–$t$ path — of every length, the maximum being $L = n - 1$ — is a model, obtained by padding it with repetitions of $t$; the remaining models are exactly the walks that revisit a cell before reaching $t$.*

*Proof sketch.* ($\Leftarrow$) Let $(x_i)$ be an absorbing walk with arrival time $\ell$. For $i < \ell$ the prefix $x_0 \cdots x_i$ is a genuine walk of length $i$ from $s$, so $d(s, x_i) \le i$ and, by bipartiteness, $i \equiv d(s, x_i) \pmod 2$; the suffix $x_i \cdots x_\ell$ reaches $t$ in $\ell - i \le L - i$ moves, so $d(x_i, t) \le L - i$. For $i \ge \ell$, $x_i = t$ and $i \ge \ell \ge d(s, t)$. Hence every position lies in its trimmed domain, and consecutive positions satisfy $T$ by definition of a walk. ($\Rightarrow$) Conversely, a satisfying assignment moves along edges until its first visit to $t$ (self-repetition is licensed only at $t$), and $x_L = t$ because $D_L = \{t\}$; it is therefore an absorbing $s$–$t$ walk. Both directions never invoke the trimming tests negatively, which proves the coincidence of model sets. Finally, a simple path padded at $t$ is an absorbing walk, and an absorbing walk with pairwise-distinct non-target positions is a padded simple path. □

**Why this compiles with zero resolvents.** The primal graph of $\Phi$ is the chain $X_0 - X_1 - \cdots - X_L$ (induced width 1), so bucket elimination never couples more than two step variables. Its only derivations would be the backward-reachability constraints "$X_i$ must lie within distance $L - i$ of $t$" propagated from the pinned endpoint — precisely the *backward* trimming test, which the domains already encode; the forward and parity tests are stronger facts that resolution would not derive at all. The trimmed input is therefore its own compilation fixpoint. Measured compilation costs (single worker, AMD EPYC 9335):

| grid | variant | clauses | resolvents | time | artifact |
|---|---|---|---|---|---|
| $12 \times 12$ | untrimmed (waiting) | 20,448 | 1,441 | 38.2 s | 11 MB |
| $12 \times 12$ | **trimmed** | 8,839 | **0** | **2.0 s** | 1.9 MB |
| $12 \times 12$ | trimmed, dataset horizon $L$=32 | 864 | 0 | 0.13 s | 113 KB |
| $30 \times 30$ | untrimmed (waiting) | 808,200 | 25,201 | 32.1 h* | 2.8 GB |
| $30 \times 30$ | **trimmed** | 379,315 | **0** | **968 s** | 612 MB |
| $30 \times 30$ | trimmed, dataset horizon $L$=80 | 10,806 | 0 | 9.1 s | 8.1 MB |

*Of the 32.1 hours, essentially all is spent in the $2(g-1) = 58$-bucket backward-reachability zone at $\approx 30$ minutes per bucket; the remaining 842 buckets produce no resolvents and pass in a fraction of the time. The 25,201 resolvents are precisely the growing balls around the target, derived one family at a time in width-900 literal arithmetic — exactly the information the backward trimming test precomputes for free. The trimmed encoding eliminates this zone entirely at every grid size (zero resolvents at $12 \times 12$ and $30 \times 30$ alike), leaving compilation dominated by parsing and subsumption bookkeeping.

**Worked example:** $2 \times 2$**.** Cells row-major: $s = 0 = (0, 0)$, $1 = (0, 1)$, $2 = (1, 0)$, $t = 3 = (1, 1)$; $d(s, t) = 2$, $L = 3$. The trimmed domains are

$$D_0 = \{0\}, \qquad D_1 = \{1, 2\}, \qquad D_2 = \{3\}, \qquad D_3 = \{3\}.$$

(E.g. $0 \notin D_1$ by parity; $0 \notin D_2$ because $d(0, t) = 2 > L - 2 = 1$; $t \notin D_1$ because $1 < d(s, t)$.) The formula has exactly two models,

$$(0, 1, 3, 3) \qquad \text{and} \qquad (0, 2, 3, 3),$$

i.e. precisely the two simple $s$–$t$ paths — at this size the walk relaxation is already exact, and the model count equals the path count.

**Worked example:** $3 \times 3$**.** Cells $0..8$ row-major, $s = 0$, $t = 8$, $d(s,t) = 4$, $L = 8$. Distances from $s$ are $(0,1,2,1,2,3,2,3,4)$ and to $t$ are $(4,3,2,3,2,1,2,1,0)$. The trimmed domains are

$$\begin{array}{ll} D_0 = \{0\} & D_1 = \{1,3\} \\ D_2 = \{0,2,4,6\} & D_3 = \{1,3,5,7\} \\ D_4 = \{0,2,4,6,8\} & D_5 = \{1,3,5,7,8\} \\ D_6 = \{2,4,6,8\} & D_7 = \{5,7,8\} \\ D_8 = \{8\} & \end{array}$$

— 29 values in total against 81 untrimmed (the parity test alone halves each interior domain; the backward test additionally removes 0 from $D_6$ and $1,3$ from $D_7$; $t$ enters only from step $d(s,t) = 4$ on). Sample models:

- a geodesic simple path, padded at the target: $(0,1,2,5,8,8,8,8,8)$;
- a maximal simple path using the full horizon (no padding): $(0,1,2,5,4,3,6,7,8)$;
- a cyclic walk retained by `allow_cycles` — it revisits 0 and 3 before arriving: $(0,1,4,3,0,3,6,7,8)$; loop erasure maps it to $(0,3,6,7,8)$.

All three satisfy every domain membership and transition. Excluded by construction are: assignments that leave the target after reaching it (absorption), assignments placing any cell off its parity class (e.g. $x_2 = 1$), jumps between non-adjacent cells, and prefixes that stray too far to return (e.g. $x_6 = 0$: $d(0,t) = 4 > L - 6$).

**Sequence-level supervision and head design.** Feeding every step variable the *same* network tile distribution leaves the sequence dimension unsupervised: the greedy walk then has no reason to progress, and empirically it oscillates inside the predicted corridor and reaches $t$ only when the backward cone forces it (arrival at the last admissible step for $100\%$ of test samples), leaving path quality intact but the raw sequence degenerate. The remedy is to let the network output *one categorical distribution per step variable* and to supervise it with the ground-truth optimal path *as a sequence*, the absorbing target repeated after arrival; the training loss applies the cross-entropy *after* the projection (on the layer-adjusted distributions), plus the mean-normalized $\ell_2$ projection regularizer. This has one structural cost: a dense head needs one logit per (step, admissible cell) pair, i.e. $\sum_i |D_i|$ outputs — benign at $12 \times 12$ (6,234 logits at the provable horizon $L$=106) but $O(Ln)$ in general (379k logits, a 194M-parameter final layer, at $30 \times 30$ with $L$=899). Two mitigations: (i) shrink the horizon — either to the provable chordless bound of Section O or, more aggressively, to the longest optimal path *observed in the dataset* (at $12 \times 12$: $L = 32$ versus the provable 106; the artifact shrinks to 869 logits and 864 clauses); (ii) factorize the head as $\mathrm{logit}(i,v) = \langle u_i, e_v(x) \rangle$ with per-cell embeddings $e_v(x)$ taken from the stride-8 feature map (one feature per grid tile) plus broadcast global context, and learned per-step queries $u_i$ — $O((L{+}n)d)$ parameters in place of $O(Ln)$. Empirically, sequence supervision removes the procrastination outright (mean arrival drops from the horizon to the geodesic within two epochs). Measured at $12 \times 12$ (CE after projection, $\lambda = 1$ mean-normalized, paper recipe). We report the *walk-cost* metric — every entered tile is paid at every visit, so wandering is penalized directly and a walk counts as optimal only if it is a wander-free optimal simple path — alongside the loop-erased strict metrics. "simple" = fraction of emitted sequences with no revisit. Dense-head rows are mean $\pm$ std over six seeds; factorized rows are single-seed:

| head | $L$ | walk-opt.% | simple | erased-opt.% | exact | ms/sample |
|---|---|---|---|---|---|---|
| dense | 32 | $68.8 \pm 2.1$ | 97.9% | $69.5 \pm 2.0$ | $0.599 \pm 0.024$ | 0.09 |
| dense | 40 | $68.5 \pm 2.1$ | 97.5% | $69.1 \pm 2.0$ | $0.608 \pm 0.044$ | 0.11 |
| dense | 106 | $66.2 \pm 4.8$ | 98.3% | $66.7 \pm 4.9$ | $0.579 \pm 0.059$ | 0.15 |
| factorized | 106 | 58.1 | — | 58.1 | 0.532 | 0.30 |
| factorized | 32 | 59.1 | — | 59.1 | 0.497 | 0.11 |

Sequence supervision removes the procrastination outright: 97–99% of emitted sequences are already simple, median arrival equals the geodesic, and the wandering-penalized and loop-erased metrics coincide to within one point — the layer outputs clean optimal paths with no post-processing credit. Across horizons the differences are within seed noise (single-seed comparisons at this task overstate contrasts by up to 7 points), so the *dataset* horizon is preferred purely on cost: the smallest head and the cheapest projection at equal accuracy. The factorized head trades roughly 10 points for its $O((L{+}n)d)$ parameter budget — the price of input-independent step queries — making it the scalable but not yet accuracy-matched option for larger grids.

**Sequence supervision at $30 \times 30$.** The same protocol at $30 \times 30$ (three seeds per configuration; horizons: dataset maximum $L{=}80$, dataset maximum plus slack $L{=}100$, provable chordless bound $L{=}674$):

| head | $L$ | walk-opt.% | ratio med | simple | exact | params | s/epoch |
|---|---|---|---|---|---|---|---|
| dense | 80 | $7.0 \pm 0.6$ | 1.235 | 94.1% | 0.055 | 16.7M | 11 |
| dense | 100 | $7.0 \pm 1.1$ | 1.263 | 95.0% | 0.053 | 21.3M | 13 |
| dense | 674 | $6.6 \pm 0.2$ | 1.349 | 98.4% | 0.058 | 154M | 48 |
| factorized | 80 | $0.0 \pm 0.0$ | 1.539 | 0.0% | 0.046 | 11.2M | 15 |
| factorized | 100 | $0.0 \pm 0.0$ | 2.485 | 0.0% | 0.004 | 11.2M | 21 |

Three conclusions. *First*, the dense head scales: even the 154M-parameter output layer at the chordless bound trains stably (48 s/epoch, 2 ms projections), and accuracy is horizon-invariant — the dataset horizon again wins on cost alone. *Second*, the factorized head fails outright at this scale (0% wander-free optimal, no simple sequences): input-independent step queries cannot drive 900-way step decisions, so the affordable head is the one that does not work, while the "expensive" dense head is entirely affordable. *Third*, the $\approx 7\%$ walk-opt plateau, shared by the greedy oracle walks and every horizon, is a *perception* ceiling, not a decoding one: an *exact* dynamic-programming decode over the down/right-monotone DAG on the same network's probabilities attains only 6.3% (median ratio 1.23), despite searching its restricted space optimally, while the masks contain a true-weight optimal path 38% of the time. The network's coarse corridor is right; its fine cost ranking between neighbouring corridors is not, and no decoder over the same probabilities — restricted or not — can recover information the scores do not carry. Improving the $30 \times 30$ operating point is therefore a backbone problem (e.g. cost regression in place of membership probabilities), orthogonal to the layer's guarantees.

# P Broader Impacts

The guarantees provided by CASPER are highly relevant in safety-critical domains, such as clinical decision support tools. However, while CASPER guarantees that its outputs satisfy the given constraints, it does not guarantee that they are correct. First, the constraints themselves may contain errors, which CASPER would enforce just as reliably as correct knowledge. Second, a prediction can satisfy every constraint and still be wrong; since all outputs appear valid, such constraint-compliant errors may go unnoticed.

## 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 abstract and introduction claim CASPER (i) guarantees constraint satisfaction by construction, (ii) scales at inference time, (iii) requires no problem-specific decomposition and (iv) is fully differentiable. These properties are substantiated by Theorem 6 (correctness) and verified by the experimental analysis detailed in Section 4.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: In Appendix A we discuss the limitations of CASPER namely we consider the problem of enforcing hard constraints over categorical variables. We do not address several orthogonal problems to ours: constraint satisfaction over continuous variables, reasoning shortcuts, modelling dependencies between predicted symbols and concept base interpretability.

   Guidelines:

   - The answer [N/A] 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: All theoretical results are stated with explicit assumptions in Section 3 and proven in Appendices B–M. Theorems 1–2 and Lemma 1 establish the single-variable projection and its satisfaction guarantee; Theorem 3 shows that the surrogate and exact projections select the same value; Theorem 4, Corollary 1, and Propositions 1–2 characterise the surrogate error and gradient behaviour; Lemma 2 and Theorem 6 extend correctness to the multi-variable case via the compilation step, whose cost is bounded by Theorem 5; and Corollary 2 shows that the surrogate does not alter the sequence of assignments. Each result is numbered and cross-referenced from the main text.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: We provide a detailed description of our method in Section 3, our key experimental results in Section 4 and all remaining details (task-formulation, experimental setup and implementation details) specific to each task in Appendix N.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: Code and instructions to reproduce all experimental results are publicly available at `https://github.com/lkonathala25/casper`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: All relevant details for our experimental analysis are covered, for each task, in Section 4.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: We report the error bars in all the tables and charts present in our paper.

Guidelines:

- The answer [N/A] 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: [Yes]

   Justification: We cover the compute resources used in our experiment analysis in Section 4.

   Guidelines:

   - The answer [N/A] 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: We confirm we have read the NeurIPS code of ethic and we comply with it.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: We discuss the potential positive and negative societal impacts of CASPER, particularly in safety-critical domains such as clinical decision support, in Appendix P.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper proposes a method that does not present any particular risk of misuse.

Guidelines:

- The answer [N/A] 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 datasets used in the experimental analysis are cited: MNIST-Arithmetic [32], Warcraft Pathfinding [40] and SatNet (Sudoku Solving) [49] where the associated licenses are outlined in Appendix N.

Guidelines:

- The answer [N/A] 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: [N/A]

    Justification: This paper does not release any new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The development of the core methodology of this paper does not involve LLMs as any important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.