Neurosymbolic Object-Centric Learning with Distant Supervision

Stefano Colamonaco
KU Leuven
stefano.colamonaco@kuleuven.be

David Debot
KU Leuven
david.debot@kuleuven.be

Giuseppe Marra
KU Leuven
giuseppe.marra@kuleuven.be

Abstract

Neurosymbolic learning can use symbolic rules to provide supervision for latent concepts from weak labels, but it commonly assumes that the entities referenced by these rules are already specified. Object-centric models decompose images into slot-like representations; however, such slots are not necessarily aligned with the predicates required for symbolic reasoning. We investigate object-centric neurosymbolic learning under distant supervision, where the object-level arguments of a logic program are learned directly from images using only global task labels. We introduce DeepObjectLog, a probabilistic neurosymbolic model that integrates a slot-based perceptual encoder with a probabilistic logic layer. The encoder predicts objectness and class probabilities for candidate object representations, while the logic layer marginalizes over latent objectness and class assignments to compute the likelihood of the observed label. This formulation provides a differentiable task-level learning signal for object-centric perception without requiring per-object labels, masks, bounding boxes, or heuristic set matching. Evaluations across diverse visual reasoning tasks demonstrate that DeepObjectLog achieves superior out-of-distribution generalization to compositional, object-count, and rule shifts compared to neural object-centric and standard neurosymbolic baselines.

1 Introduction

Neurosymbolic AI aims to combine neural representation learning with explicit reasoning over entities, relations, and compositional rules [19, 29, 23]. A central appeal of this paradigm is that symbolic knowledge can provide supervision for latent concepts even when direct labels are unavailable. For example, a model trained only on the sum of two digit images can learn to classify the individual digits, because the addition rule constrains which latent digit assignments are compatible with the observed sum [19]. However, this ability usually relies on an important precondition: the entities over which reasoning should occur are already given. In the digit-addition example, the model is typically provided with two separate digit images. It must learn what each digit is, but not where the digits are, how many there are, or which parts of a larger image correspond to distinct objects.

This precondition limits the applicability of neurosymbolic learning to perceptual inputs whose object structure is not known in advance. In many visual domains, an image may contain an unknown number of entities, with no labels, masks or bounding boxes. Before a symbolic rule can constrain object properties, the model must infer which candidate parts of the input should be treated as task-relevant objects. Thus, the bottleneck is not only learning symbolic concepts from weak supervision, but also learning the object-centric structure that makes symbolic reasoning possible.

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

Object-centric learning addresses this complementary problem by decomposing visual scenes into slot-like representations, each intended to capture a candidate object or scene component [1, 12, 17, 5]. These representations provide a natural interface for neurosymbolic reasoning, since slots can in principle serve as the entities over which rules operate. Yet slots are not automatically aligned with the predicates needed by a symbolic task. A slot may capture a visual region without being identifiable as a task-relevant object, and a set of slots may support a neural classifier without producing the explicit object-level structure required by the symbolic component. When object-level supervision is unavailable, it remains unclear how to align object-centric perception with symbolic reasoning.

We study this interface. Given image-level labels and a program describing how latent object-level predicates determine those labels, we ask whether symbolic task knowledge can guide the learning of task-relevant object-centric structure, as specified in the box below.

Problem. [*Object-centric neurosymbolic learning from distant supervision*]

***Setup*:** We are given a dataset of image–label pairs (x, y), where each image may contain multiple objects but no annotation is provided for object identity, location, mask, or class. We are also given a probabilistic logic program L, a set of rules specifying how latent object-level predicates, e.g. objectness and class, determine the global label y. Notably, the label alone does not reveal how many objects are present, where they are, or which classes they belong to and the given symbolic rules only constrain which decompositions could explain the observation.
***Goal*:** Learn a neurosymbolic model in which perception and reasoning inform each other: the perceptual module proposes candidate objects and their predicates, while the symbolic layer evaluates whether those predicates can explain the observed task label. The learned model should support relational generalization under distribution shifts, including new object compositions, different object counts within the slot capacity of the model, and new supplied symbolic queries or rules over the learned object-level vocabulary.

We propose DeepObjectLog, a neurosymbolic architecture that integrates object-centric perception with probabilistic logical reasoning. The perceptual component extracts a set of candidate object representations from an image. For each candidate, the model predicts an objectness probability and a distribution over symbolic classes. These predictions are passed as probabilistic facts to a ProbLog program [3], which computes the marginal probability of the observed task label by tractably summing over possible objectness and class assignments. This marginal likelihood provides a differentiable signal from the global task label back to the object-centric encoder, allowing logical constraints to influence the selection, classification, and grounding of candidate objects.

We evaluate DeepObjectLog on MultiMNIST-Addition, PokerRules, CLEVR-Addition, and a COCOLogic-based setting [31]. These controlled benchmarks test whether a model can use task-level symbolic structure to learn object-centric representations that support reasoning under distribution shift. Across these settings, DeepObjectLog achieves stronger out-of-distribution task performance than neural object-centric baselines and than a neurosymbolic baseline using the same logic but lacking object-centric perception.

The contributions of this work are threefold. First, we formulate object-centric neurosymbolic learning from distant supervision, where objects are not provided as structured inputs but must be inferred from perception. Second, we introduce a probabilistic interface between slots and logic, in which objectness and class variables are marginalized by probabilistic logical inference, providing a differentiable task-level signal without object annotations, discrete sampling, or set-matching losses. Third, we instantiate this formulation in DeepObjectLog and show that logical feedback improves the task-relevant use of object-centric representations and supports stronger relational generalization, while also analyzing limitations such as inference cost, slot-capacity sensitivity, and cases where task-level labels do not uniquely identify human-named concepts.

A graphical overview of the proposed model is provided in Figure 1, while a detailed explanation of its components is given in Section 3.

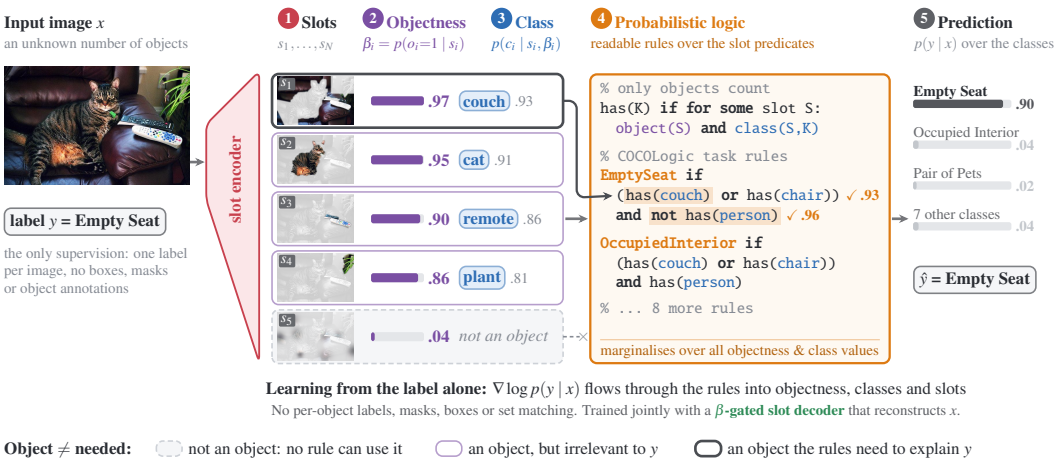

Figure 1: Overview of the proposed model, DeepObjectLog, on a COCOLogic example. A slot encoder extracts N slots, and each slot gets an objectness probability and a class distribution. A logic program marginalizes over all objectness and class assignments to compute the probability of each task label, so the label alone supervises objectness, classes and slots, with no per-object labels, masks, boxes or set matching. Objectness also gates the slot decoder, so a slot that is not an object (s5) is used by neither the logic nor the reconstruction.

2 Related Work

The problem addressed in this study lies at the intersection of two research areas: object-centric representation learning and neurosymbolic reasoning. Both are essential, but neither is sufficient on its own to meet the demands of a system capable of learning relational abstractions directly from raw input when supervision is provided only at the task level.

Object-centric learning. Object-centric models aim to decompose visual scenes into discrete object representations [13]. Early approaches [1, 12, 8] use iterative variational inference to discover object structure in an unsupervised manner, but often suffer from scalability limitations. Slot Attention [17] replaces these procedures with a parallel attention-based binding mechanism, enabling permutation-invariant inference over latent slots. This model has since been extended across several domains [15, 7, 25, 28], with slot-based representations being applied to generative modeling and dynamic scenes. Some works attempt to improve the selection of useful slots by adaptively selecting which representations carry meaningful content [10, 9], but these still rely on heuristics or sparsity priors that are disconnected from task-level semantics. A recent study [15] shows that slot-based models can be trained end-to-end with supervision on a downstream task, but it assumes a fixed number of objects and employs a neural classifier without symbolic reasoning, requiring few-shot learning to achieve task generalization. While certain architectures perform downstream tasks like VQA without explicit labels [35], they often lack discrete symbolic grounding. Without task-level logical supervision, current models learn representations independent of the task's symbolic structure, limiting their interpretability and out-of-distribution generalization [34].

Neurosymbolic systems. A complementary limitation appears in most neurosymbolic (NeSy) systems: while they reason over structured entities, they assume that this structure is already available. Thus, they cannot learn to infer object-centric representations from raw data. While NeSy approaches integrate neural perception with symbolic logic for probabilistic reasoning [23, 19], they typically assume inputs consist of pre-segmented objects (e.g. individual images or attribute vectors), bypassing the challenge of structure discovery [36, 21]. Existing extensions incorporate limited spatial reasoning or component-level architectures but usually operate with a fixed number of entities [24, 4]. Consequently, these systems lack object-agnostic parameterization.

Stammer et al. [30] and Shindo et al. [26] employ a pretrained Slot Attention module for object-centric representations, subsequently using them for rule extraction. Skryagin et al. [29] explore the integration of logical constraints into the training of object-centric modules, demonstrating that

logical supervision can improve object decomposition by introducing assumptions of independence across attributes. Finally, some works in abductive learning have attempted to learn object detection from unstructured data, but they typically rely on knowing in advance how many objects are present in each image, which limits their ability to fully handle unstructured inputs [11, 2]. While these approaches demonstrate the potential of combining perception with logic, they still rely on object-level supervision, such as object labels or question-answer pairs, to align neural representations with symbolic targets.

3 Method

To address these limitations, we introduce DeepObjectLog, a neurosymbolic architecture that learns object-centric representations from weak supervision and reasons over their latent properties.

From a high level, the architecture is composed of the following components: (i) **Objects encoder**: starting from a global representation z, the model extracts up to N latent representations s_i, each intended to describe a potential object in the scene; (ii) **Object detector and classifier**: for each extracted object encoding, the model infers a probability β_i that the representation corresponds to a meaningful object, from which an *objectness* flag o_i and a class c_i are extracted; (iii) **Probabilistic logic reasoning**: the system reasons logically over detected objects and their classes to derive a downstream task. This enables the system to interpret objects, understand their relationships, and constrain object configurations in ways that support task-specific inference. (iv) **Image decoder**: finally, each representation is decoded into image space.

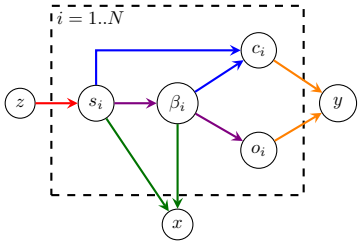

Figure 2: Probabilistic graphical model of DeepObjectLog

3.1 Probabilistic graphical model

Let $z \in \mathbb{R}^{d_z}$ be a latent representation of the image. For each potentially present object i, we denote its associated slot as $s_i \in \mathbb{R}^{d_s}$, and the probability that it represents an actual object as $\beta_i \in [0, 1]$. The corresponding objectness flag is $o_i \in \{0, 1\}$ and the class label corresponding to the object is $c_i \in C \subset \mathbb{N}$. Finally, let $y \in Y \subset \mathbb{N}$ be the downstream class label computed by a symbolic logic theory L, and $x \in [0, 1]^{ch \times h \times w}$ the normalized image. The generative model in Figure 2 factorizes as:

$$p(y, x, \mathbf{s}, \boldsymbol{\beta}, \mathbf{o}, \mathbf{c}, z) \;=\; p(z)\; p_L(y|\mathbf{o}, \mathbf{c})\; p(x|\mathbf{s}, \boldsymbol{\beta}) \cdot \Big[\prod_{i=1}^{N} p(s_i \mid z) p(\beta_i|s_i)\, p(o_i|\beta_i)\, p(c_i|\beta_i, s_i)\Big] \tag{1}$$

where $\mathbf{s} = (s_1, \ldots, s_N),\ \boldsymbol{\beta} = (\beta_1, \ldots, \beta_N),\ \mathbf{o} = (o_1, \ldots, o_N),\ \mathbf{c} = (c_1, \ldots, c_N)$.

This joint distribution consists of the following components:

- $p(z)$ a prior distribution of the latent representation of the scene;
- $p(s_i \mid z)$: the *slot extractor*, which produces the i-th latent slot representation s_i out of N maximum possible objects based on the latent representation z. A slot s_i represents a *proposal* for a potential object.
- $p(\beta_i \mid s_i)$ models the uncertainty of s_i being a meaningful object (and not, e.g., part of the background); it is represented explicitly since object-centric models can condition their processing on such uncertainty [10];
- $p(\mathrm{o}_i \mid \beta_i)$ models the binary flag corresponding to the uncertainty β_i,
- $p(c_i \mid \beta_i, s_i)$: the symbolic classification head, which assigns a discrete class label c_i to the representation s_i, conditioned on its uncertain objectness β_i;
- $p_L\,(y \mid \mathbf{o}, \mathbf{c})$: the symbolic reasoning component, which predicts the final task-specific output y by reasoning on a logic theory L, the inferred objects o_i and their classes c_i;

- $p(x \mid \mathbf{s}, \boldsymbol{\beta})$: the image generator, a probability distribution over images given the set of slots and their predicted objectness.

This formulation makes object structure latent and uncertain, while coupling symbolic prediction and perceptual reconstruction through shared object-centric representations. Notice that, although N fixes the maximum number of slots, all slot-wise modules share parameters and are independent of the actual number of objects, which is inferred through the objectness variables.

In the sections below, we describe the main components of the system in more detail.

3.2 Object independent encoder in the discriminative setting

In this paper, we are interested in a discriminative setting, where we focus on modeling the probability distribution of the object classes c_i by observing the task label y (see Section 4), and using the image x to amortize inference. Following Kori et al. [16], we model the slot encoder distributions $p(s_i|\cdot)$ as deterministic delta distributions. Such an encoder should avoid relying on supervision or assumptions about object count or location, ensuring flexibility across inputs and generalization to novel object numbers and positions. Many possible existing models satisfy such requirements [1, 8] but we leave their comparison for future work. In our implementation, we exploit Slot Attention [17, 37], a differentiable mechanism for object-centric representation learning. In particular, $p(s_i|z) = SlotAttention_i(z; \theta_s, N)$ where θ_s is the real set of parameters of the slot attention model, which are shared across slots. The number N is therefore only a maximum-slot hyperparameter, while downstream objectness variables determine which slots correspond to actual objects.

3.3 Object detector and classifier

The object detection is modelled over two variables: the uncertainty scores $\beta_i \in [0, 1]$ and the objectness flags $o_i \in \{0, 1\}$. The meaning is that o_i is 1 (i.e. it represents an object) with probability β_i. We model directly the uncertainty β_i over o_i as many components in current state-of-the-art object-centric models exploit soft masking schemes using such uncertainty. However, we do not model any probability distribution over β_i; therefore $p(\beta_i|s_i) = \delta(\beta_i - \beta_i^*)$ is a delta distribution centered around the output value of a neural network $\beta_i^* = f_\beta(s_i|\theta_\beta)$ with θ_β its set of weights. We model $p(o_i|\beta_i)$ as a Bernoulli distribution, parameterized by β_i, i.e. $p(o_i|\beta_i) = \beta_i$. The classifier $p(c_i|s_i, \beta_i)$ can be parameterized with either a Bernoulli or with a Categorical distribution, depending on the particular object classification task. In both cases, the distribution is parameterized by a neural network $f_c(\beta_i s_i|\theta_c)$ with weights θ_c. Following Fan et al. [10], the slot representation s_i in the input to the network f_c is also multiplied elementwise by the uncertainty β_i. gating uncertain slots and reducing the influence of background or spurious proposals.

3.4 Probabilistic Logic Reasoning over the tasks

The final symbolic prediction y is computed using probabilistic logical reasoning over the set of inferred object class predictions and objectness probabilities. This component is implemented using ProbLog [3], a probabilistic logic programming framework that extends classical logic programming with uncertainty modeling through probabilistic facts. While ProbLog is Turing-equivalent and allows for highly expressive probabilistic modeling, in this work we focus only on the aspects relevant to neurosymbolic object-centric learning and limit our explanation accordingly. For a full overview of the framework, we refer interested readers to De Raedt et al. [3].

Probabilistic Logic Programming. A ProbLog program L is defined over two sets of syntactic constructs: probabilistic facts and rules. A probabilistic fact with syntax `p::f` is an independent Bernoulli distribution $p(f)$ over a binary variable `f` with parameter `p` $= p(f)$. For example, `0.1::alarm` means that variable `alarm` has 0.1 probability of being True. A rule is of the form `h :-` $\mathtt{b}_1, ..., \mathtt{b}_m$, where `h` is the head or conclusion and each $\mathtt{b}_i$ is a body element or premise. Rules act as definition rules: if all the premises $\mathtt{b}_i$ are True, then the conclusion `h` must be True. For example, in the classical Pearl's burglary example, the rule `"call :- alarm, hear"` encodes that if the `alarm` goes off and we `hear` it, then we `call` the police. Given the set of all probabilistic facts `f` and the set of all possible rules R, ProbLog allows to efficiently compute the marginal probability of every symbol y

in the program as:

$$p(y) = \sum_{\mathbf{f}} p(y|\mathbf{f}; R) \prod_{f_i \in \mathbf{f}} p(f_i) \tag{2}$$

where $p(y|\mathbf{f}; R)$ is a deterministic distribution stating whether y can be obtained from $\mathbf{f}$ when applying the rules in R (possibly chaining multiple of them).

ProbLog for object-centric learning. We use ProbLog to model three components of our model:

- The object detectors $p(o_i|\beta_i)$ are modelled as probabilistic facts `object(i)` stating whether the i-th extracted slot is an object;
- The object classifiers $p(c_i|s_i, \beta_i)$ are modelled as probabilistic facts `class(i)` (for a binary classifier) or `class(i,k)` for a categorical classifier with class k. The conditional nature on the objectness o_i is encoded using rules. The rule `task:-object(1),class(1,3)`, for example, states that `task` is true only if the 1st slot is an object and its class is 3.
- The conditional task distribution $p_L(y|\mathbf{c}, \mathbf{o})$ is the deterministic distribution obtained by all the logic rules proving the task y given the objectnesses and the classes of all objects.

Example 1 (Addition of at most two digits) *Let us consider a setting where images contain zero to two digits that can only be* 0 *or* 1*. Consider the following two kinds of probabilistic facts. First,* $\mathtt{p}_{\mathtt{o_{id}}}$ :: `object(ID)` *stating that the probability that the* `ID`*-th slot of the input image contains an actual digit is* $\mathtt{p}_{\mathtt{o_{id}}}$*, as computed by* $p(o_i|\beta_i)$*. Second,* $\mathtt{p}_{\mathtt{c_{id}}}$ :: `class(ID,C)` *providing the probability that the* `ID`*-th object is of class* C*, as computed by* $p(c_i|s_i, \beta_i)$*. We can encode the digit addition task in the following program:*

```
digit(ID, Val) :- object(ID), class(ID, Val).
digit(ID, 0) :- \+ object(ID).
add(Z) :- digit(1, Y1), digit(2, Y2), Z is Y1 + Y2.
```

We can then query, for example, what is the probability that a given image with multiple digits sums to 2, i.e. $y =$ `add(2)`*, given the distributions of the current objectness of the slots and their corresponding classes.*

Note that any marginal task distribution $p(y|\mathbf{s}, \boldsymbol{\beta})$, can be computed by standard marginal inference (Eq. 2) of the task query y:

$$p(y|\mathbf{s}, \boldsymbol{\beta}) = \sum_{\mathbf{c},\mathbf{o}} p(y|\mathbf{c}, \mathbf{o}) \prod_i p(o_i|\beta_i) p(c_i|s_i, \beta_i) \tag{3}$$

It is interesting to note that the use of probabilistic logic eliminates the need for set matching losses typically employed in object-centric systems (e.g., Hungarian loss), as object assignments are implicitly resolved through marginalization.

On the Role of Knowledge. The logic rules in our framework describe how object-level concepts should be composed to determine the global label, yet they do not provide any per-instance supervision. This is standard in neurosymbolic learning and reflects the core premise of the paradigm: when symbolic knowledge is available, it should be seamlessly usable by the model. If no knowledge is provided, the model trivially reduces to a purely neural architecture, as standard deep approaches are simply a special case within our framework. Conversely, when partial knowledge exist (e.g., from human experts, domain rules, knowledge bases, or even rules induced by LLMs [27]) the same interface allows it to be incorporated naturally and consistently.

3.5 Masked Decoder

For the image decoder $p(x|\mathbf{s}, \boldsymbol{\beta})$, we follow recent literature in Slot-attention architectures. In particular, we model the image as a mixture of Multivariate Gaussian distributions. Each component is a multivariate Gaussian $\mathcal{N}(\mu, I)$ with unit covariance matrix I and a mean vector μ parameterized with a neural network $f_x(s_i\beta_i; \theta_x)$ with θ_x its parameters. The mixing weights w of the components are parameterized by a neural network $f_w(s_i\beta_i; \theta_w)$. Notice, as we did for the objectness variable o_i and as often done in the slot-attention literature, we decode each slot s_i after a soft masking using β_i.

4 Learning

In the discriminative setting considered in this work, learning is formulated as joint optimization over model parameters and latent explanations. Each training image–label pair (x_k, y_k) is assigned a single latent explanation z_k, and the model is trained to maximize the joint probability of the observed data and their explanations:

$$\max_{\theta_p, \{z_k\}_{k=1}^K} \sum_{k=1}^{K} \log p_{\theta_p}(x_k, y_k, z_k). \tag{4}$$

This is a MAP-style objective, in which the model searches for one high-scoring explanation per training example. Here z_k is the latent explanation for each training example (x_k, y_k) and $\theta_p = \{\theta_s, \theta_\beta, \theta_c, \theta_x, \theta_w\}$ denotes the parameters of the probabilistic model.

Starting from the full joint model $p_{\theta_p}(y, x, \mathbf{s}, \boldsymbol{\beta}, \mathbf{o}, \mathbf{c}, z)$, the score $p_{\theta_p}(x, y, z)$ is obtained by marginalizing the internal variables that are not directly optimized. Since $\mathbf{s}$ and $\boldsymbol{\beta}$ are deterministic in our model, their distributions collapse to point values $\mathbf{s}^\star(z)$ and $\boldsymbol{\beta}^\star(z)$. Using the sifting property of delta distributions and the task marginalization of ProbLog (Equation 3), we obtain

$$p_{\theta_p}(x, y, z) = p(z)\, p_{\theta_p}\big(x \mid \mathbf{s}^\star(z), \boldsymbol{\beta}^\star(z)\big)\, p_{\theta_p}\big(y \mid \mathbf{s}^\star(z), \boldsymbol{\beta}^\star(z)\big). \tag{5}$$

The ideal objective in Equation 4 would require optimizing a separate latent variable z_k for every training example. To avoid this inner optimization, we amortize the latent explanation using a neural encoder q_ϕ, which predicts $z_k \simeq q_\phi(x_k)$. Substituting this amortized estimate into the MAP objective gives the practical training objective

$$\max_{\theta_p, \phi} \sum_{(x,y)\in D} \log p_{\theta_p}(x, y, q_\phi(x)). \tag{6}$$

Expanding the joint score, the loss becomes

$$\max_{\theta_p, \phi} \sum_{(x,y)\in D} \Big[\log p_{\theta_p}\big(y|\mathbf{s}^\star(q_\phi(x)), \boldsymbol{\beta}^\star(q_\phi(x))\big) + \log p_{\theta_p}\big(x|\mathbf{s}^\star(q_\phi(x)), \boldsymbol{\beta}^\star(q_\phi(x))\big) + \log p\big(q_\phi(x)\big) \Big] \tag{7}$$

The first term encourages the inferred object-centric representation to explain the symbolic task label. The second term encourages the same representation to reconstruct the input image. The third term is a prior term on the amortized latent explanation. Thus, the encoder is trained to produce a single latent explanation that jointly accounts for the image, the task label, and the prior.

5 Experiments

We focus on evaluating the generalization capabilities of DeepObjectLog, as generalization under distributional shifts offers the clearest evidence that a model is truly learning to decompose scenes into meaningful components and reason over them [13, 6]. We pose the following research questions:

Q1: Compositional generalization: Can the model handle novel combinations of objects seen during training?
Q2: Task generalization: Can the model generalize to new tasks not present in the training data?
Q3: Object count generalization: Can the model handle object configurations larger (extrapolation) or smaller (interpolation) than those seen during training?
Q4: Stronger learning signal: Does our method provide a stronger object-level learning signal than purely neural task prediction?
Q5: Foundation models: How do pretrained Vision-Language Models (VLMs) compare to our approach on structured reasoning tasks?

5.1 Setup

Datasets. We consider 4 datasets. For additional details, we refer to Appendix B. *(1) MultiMNIST-Addition* (MM-A) is our controlled diagnostic for learning object detection and predicates from task labels only. Each image contains up to three MNIST digits, and the only training label is their

sum; digit identities, locations, and object counts are not used for training. We evaluate three shifts: compositional generalization, by training on 75% of digit combinations with 0–3 digits and testing on the remaining 25%; extrapolation, by testing on 4- and 5-digit images with either seen or unseen sums; and interpolation, by training on 0-, 1-, and 3-digit images and testing on 2-digit images. *(2) PokerRules* tests whether learned object predicates can support relational rules beyond digit arithmetic. Images contain multiple playing cards and labels are poker-hand categories defined by symbolic relations among card ranks. We evaluate supplied rules for hand types not seen during training and extrapolation to scenes with more cards. This setting also probes an important limitation: some task labels identify relational structure, such as equality of ranks, more strongly than human-named card classes. *(3) CLEVR-Addition (CLEVR-A)* tests objectness and counting in scenes with more varied visual structure. We form pairs of CLEVR images and label each pair by the total number of objects across the two images. The evaluation includes object-count extrapolation, including seen and unseen output sums. With *(4) COCOLogic* [31] we study how task-level logical supervision interacts with partial object-level annotations in a task with more natural images.

Baselines. We compare our approach against four neural baselines: a standard CNN, the Slot Attention model, MESH [37], and CoSA [15]. Notably, CoSA is a recent work and, to the best of our knowledge, the only existing method that has explicitly attempted tasks of this type. We further include a Transformer predictor (TF), which replaces the MLP over slots with a Transformer encoder. Additionally, for tasks that do not require generalization to larger numbers of objects, we include comparisons with a standard neurosymbolic (NeSy) implementation following the setup described in Misino et al. [24] for the classification task. For working with real-world images, we additionally include DINOSAUR [25]. In the experiment with COCOLogic we use DINOSAUR both as a competitor and as a backbone for our method. Full details of the baselines and implementations are provided in Appendix B.

Metrics. We report task accuracy, defined as the proportion of correctly classified images, and concept accuracy, which measures the correctness of the inferred object representations. Note that concept accuracy is not computable for fully neural baselines, as they do not model objects explicitly. In the COCOLogic setting, we report balanced accuracy, as done in the original paper [31].

6 Results

We evaluate DeepObjectLog's ability to generalize across the four core research questions introduced above. Additional experiments and ablation studies are provided in Appendix C.

Q1: DeepObjectLog generalizes more robustly to novel combinations of familiar objects. In the MM-A task (Table 1), our model achieves a Task Accuracy of 90% on out-of-distribution (OOD) compositions, higher than the neural and object-centric baselines. This demonstrates the model's ability to exploit the provided logic knowledge to learn task-relevant object predicates that are more independent on the particular combinations seen in the training set. The comparison with NeSy helps isolate the role of object-centric perception: although NeSy uses the same symbolic reasoning component, it achieves only 35.76% in this setting.

Q2: DeepObjectLog allows for generalization to novel classes. In the PokerRules task (Table 2), our model achieves an OOD class accuracy of 72%, compared with 0.46% for NeSy; neural baselines are not directly applicable in this setting because their output spaces are fixed by the training labels. Similar results are also reported for the MM-A and the CLEVR-A tasks (see Appendix C). These results are consistent with the model learning object-level predicates that can be reused by new supplied rules, rather than only fitting the original image-level labels, as the Concept Accuracy results shows in Table 1. This object-centric interface makes it possible to apply new symbolic rules over the learned predicates: by grounding predictions in symbolic concepts, the same perceptual model *can be reused for different reasoning tasks on the same input domain* simply by modifying the logical program, with performance depending on how well the learned predicates align with the concepts required by the new rule.

Q3: DeepObjectLog adapts to new structural configurations. When evaluated on images containing a number of objects not seen during training, our model generalizes more effectively than neural baselines. In the MM-A task, the model maintains substantially higher accuracy on extrapolation test sets with four and five digits, while competitors show significant degradation. The interpolation results, reported in Appendix C, also show a similar trend. Similarly, in the PokerRules and

Table 1: Task Accuracy (overall image) and Concept Accuracy (object-level decomposition) in MM-A. Results are on a test set of in-distribution compositions (**Test**) and on out-of-distribution compositions (**OOD**) . We also report extrapolation performance on images with four or five digits OOD counts. Dashes indicate cases where a model cannot provide results due to architectural limitations.

MM-A	Task Acc.		Concept Acc.		Extrapolation	
	Test	OOD	Test	OOD	4 digits	5 digits
CNN	79.90±0.17	3.43±0.15	-	-	22.16±0.90	13.16±0.96
SA	**98.90**±0.34	13.30±3.14	-	-	36.23±4.44	9.76±6.11
MESH	98.86±0.47	18.26±1.33	-	-	37.50±1.15	12.00±0.51
Tf	95.20±0.42	52.28±9.11	-	-	50.09±8.12	18.42±9.85
CoSA	93.00±1.92	36.2±15.10	-	-	52.20±14.01	25.06±12.82
NeSy	90.70±4.02	35.76±6.24	55.43±26.40	23.83±3.82	-	-
Ours	94.26±2.00	**90.00**±3.01	**85.16**±2.45	**65.46**±5.70	**69.73**±10.74	**44.06**±3.85

Table 2: Task Accuracy on PokerRules for both in-distribution test classes and OOD classes. A dash indicates cases where a model cannot provide results due to architectural limitations.

	Test	OOD class	In-distribution class	OOD class
PokerRules			Extrapolation: 5 cards	
CNN	81.96±1.10	-	22.03±5.27	-
SA	99.30±1.21	-	35.86±8.72	-
MESH	**99.93**±0.11	-	37.80±4.91	-
Tf	97.41±0.15	-	15.20±7.41	-
CoSA	95.46±3.90	-	44.26±17.16	-
NeSy	80.23±2.11	0.46±0.05	-	-
Ours	97.90±1.17	**72.23**±16.72	**78.53**±4.68	**78.23**±19.24

CLEVR-A tasks, the model shows stronger extrapolation than competitors both on known classes and previously unseen class configurations, where other approaches either degrade substantially or are not applicable. This result is consistent with the intended role of objectness: slots can be activated or suppressed depending on the input, allowing the same symbolic rule to operate over different numbers of predicted objects.

Q4: Logical reasoning provides a stronger task-level signal for object-centric learning. As shown in Figure 3, on COCOLogic, at 0% object-level supervision DINOSAUR cannot provide concept predictions, while our method already achieves non-trivial concept accuracy, demonstrating its ability to learn object structure purely from the task signal. This advantage is maintained as the level of supervision increases. Task accuracy remains comparable across methods; notably, this is achieved while preserving the interpretability and guarantees of our logical reasoning component. For reference, a Concept Bottleneck Model in [31] achieves a task accuracy of 58.84, which is in line with our results.

Q5: DeepObjectLog compares favorably to pretrained Vision-Language Models on MM-A. We evaluated LLaVA-1.5-7B-hf and Qwen2.5-VL-7B-Instruct on MM-A by providing the task description and possible answers. Under this prompting setup, both models struggled: the best competitor achieved only 59.93% accuracy on OOD compositions. Under the same benchmark evaluation, DeepObjectLog reached 90% accuracy. While this comparison is not strictly fair, since these models are not explicitly trained for this task, it demonstrates that the benchmark is non-trivial even for strong general-purpose VLMs. Details and results are provided in Appendix C.

7 Conclusions

We propose DeepObjectLog, a neurosymbolic model that integrates object-centric perception with probabilistic logical reasoning to jointly infer and reason over a variable number of latent objects. Our results show that (1) the proposed method strongly outperforms neural and neurosymbolic baselines

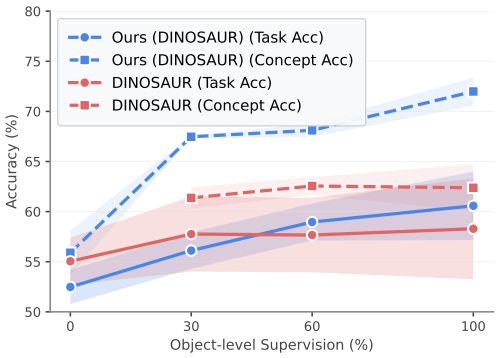

Figure 3: Concept and task accuracy on CO-COLogic under varying levels of object-level supervision.

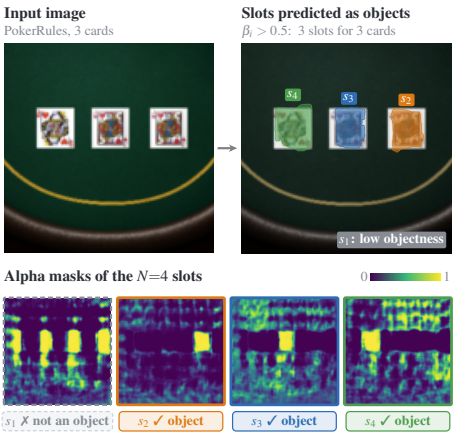

Figure 4: Alpha masks of the slots learned by DeepObjectLog on PokerRules. Slots that attend to a single card are predicted as objects ($\beta_i > 0.5$), while the slot spread over multiple regions receives low objectness.

in compositional generalization tasks, (2) it effectively generalizes to scenes with previously unseen object counts and classes, and (3) it enables robust reasoning under uncertainty without requiring explicit object-level supervision. These advances mark an important step toward more general neurosymbolic systems, contributing to the development of robust and interpretable AI, though future deployments must mitigate risks of bias and dual-use surveillance.

Limitations and future work. Our approach shares limitations with current object-centric and neurosymbolic methods. First, exact inference makes the model less scalable than purely neural approaches. While we mitigate this using [18] (Appendix C), approximate methods could be adopted to further address this limitation. Second, weakly supervised models often fall back on reasoning shortcuts [22], but our contribution is a step toward overcoming these challenges by enforcing structural constraints that discourage such behaviors.

Acknowledgements

The authors would like to express their gratitude to Luc De Raedt and Robin Manhaeve for the early discussions on the subject of this research. This research has received funding from the KU Leuven Research Fund (GA No. STG/22/021, CELSA/24/008), from the Research Foundation-Flanders FWO (GA No. 1185125N, G047124N), from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme and from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (GA No. 101142702).

References

[1] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[2] L.-W. Cai, W.-Z. Dai, Y.-X. Huang, Y. Li, S. H. Muggleton, and Y. Jiang. Abductive learning with ground knowledge base. In *IJCAI*, pages 1815–1821, 2021.

[3] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, pages 2462–2467. IJCAI-INT JOINT CONF ARTIF INTELL, 2007.

[4] L. De Smet, P. Z. Dos Martires, R. Manhaeve, G. Marra, A. Kimmig, and L. De Readt. Neural probabilistic logic programming in discrete-continuous domains. In *Uncertainty in Artificial Intelligence*, pages 529–538. PMLR, 2023.

[5] F. De Sousa Ribeiro, K. Duarte, M. Everett, G. Leontidis, and M. Shah. Object-centric learning with capsule networks: A survey. *ACM Computing Surveys*, 56(11):1–291, 2024.

[6] A. Dittadi, S. Papa, M. D. Vita, B. Schölkopf, O. Winther, and F. Locatello. Generalization and robustness implications in object-centric learning. 7 2021.

[7] G. Elsayed, A. Mahendran, S. Van Steenkiste, K. Greff, M. C. Mozer, and T. Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. *Advances in Neural Information Processing Systems*, 35:28940–28954, 2022.

[8] M. Engelcke, A. R. Kosiorek, O. P. Jones, and I. Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

[9] M. Engelcke, O. Parker Jones, and I. Posner. Genesis-v2: Inferring unordered object representations without iterative refinement. *Advances in Neural Information Processing Systems*, 34: 8085–8094, 2021.

[10] K. Fan, Z. Bai, T. Xiao, T. He, M. Horn, Y. Fu, F. Locatello, and Z. Zhang. Adaptive slot attention: Object discovery with dynamic slot number. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23062–23071, 2024.

[11] E.-H. Gao, Y.-X. Huang, W.-C. Hu, X.-H. Zhu, and W.-Z. Dai. Knowledge-enhanced historical document segmentation and recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8409–8416, 2024.

[12] K. Greff, R. L. Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. Multi-object representation learning with iterative variational inference. In *International conference on machine learning*, pages 2424–2433. PMLR, 2019.

[13] K. Greff, S. van Steenkiste, and J. Schmidhuber. On the binding problem in artificial neural networks. 12 2020.

[14] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[15] A. Kori, F. Locatello, F. D. S. Ribeiro, F. Toni, and B. Glocker. Grounded object-centric learning. In *ICLR*, 2024.

[16] A. Kori, F. Locatello, A. Santhirasekaram, F. Toni, B. Glocker, and F. De Sousa Ribeiro. Identifiable object-centric representation learning via probabilistic slot attention. *Advances in Neural Information Processing Systems*, 37:93300–93335, 2024.

[17] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

[18] J. Maene, V. Derkinderen, and P. Z. D. Martires. Klay: Accelerating neurosymbolic ai. *arXiv preprint arXiv:2410.11415*, 2024.

[19] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

[20] R. Manhaeve, S. Colamonaco, V. Derkinderen, R. Adriaensen, L. Van Praet, L. De Raedt, and G. Marra. Deeplog: A software framework for modular neurosymbolic ai. *arXiv preprint arXiv:2605.10279*, 2026.

[21] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[22] E. Marconato, S. Teso, A. Vergari, and A. Passerini. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. *Advances in Neural Information Processing Systems*, 36:72507–72539, 2023.

[23] G. Marra, S. Dumančić, R. Manhaeve, and L. De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328:104062, 2024.

[24] E. Misino, G. Marra, and E. Sansone. Vael: Bridging variational autoencoders and probabilistic logic programming. *Advances in Neural Information Processing Systems*, 35:4667–4679, 2022.

[25] M. Seitzer, M. Horn, A. Zadaianchuk, D. Zietlow, T. Xiao, C.-J. Simon-Gabriel, T. He, Z. Zhang, B. Schölkopf, T. Brox, et al. Bridging the gap to real-world object-centric learning. *arXiv preprint arXiv:2209.14860*, 2022.

[26] H. Shindo, V. Pfanschilling, D. S. Dhami, and K. Kersting. α ilp: thinking visual scenes as differentiable logic programs. *Machine Learning*, 112(5):1465–1497, 2023.

[27] H. Shindo, M. Brack, G. Sudhakaran, D. S. Dhami, P. Schramowski, and K. Kersting. Deisam: Segment anything with deictic prompting. *Advances in Neural Information Processing Systems*, 37:52266–52295, 2024.

[28] G. Singh, Y.-F. Wu, and S. Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. *Advances in Neural Information Processing Systems*, 35:18181–18196, 2022.

[29] A. Skryagin, W. Stammer, D. Ochs, D. S. Dhami, and K. Kersting. Neural-probabilistic answer set programming. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pages 463–473, 2022.

[30] W. Stammer, P. Schramowski, and K. Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3619–3629, 2021.

[31] D. Steinmann, W. Stammer, A. Wüst, and K. Kersting. Object centric concept bottlenecks. *arXiv preprint arXiv:2505.24492*, 2025.

[32] S.-H. Sun. Multi-digit mnist for few-shot learning, 2019. URL `https://github.com/shaohua0116/MultiDigitMNIST`.

[33] E. van Krieken, T. Thanapalasingam, J. Tomczak, F. Van Harmelen, and A. Ten Teije. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. *Advances in Neural Information Processing Systems*, 36:24586–24609, 2023.

[34] T. Wiedemer, J. Brady, A. Panfilov, A. Juhos, M. Bethge, and W. Brendel. Provable compositional generalization for object-centric learning. *arXiv preprint arXiv:2310.05327*, 2023.

[35] Z. Wu, N. Dvornik, K. Greff, T. Kipf, and A. Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

[36] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

[37] Y. Zhang, D. W. Zhang, S. Lacoste-Julien, G. J. Burghouts, and C. G. Snoek. Unlocking slot attention by changing optimal transport costs. In *International Conference on Machine Learning*, pages 41931–41951. PMLR, 2023.

Supplementary Material

Table of Contents

A Amortized MAP objective **14**

A.1 Prior and Activation Regularization 15

B Experimental details **15**

B.1 Architectural details 15

B.2 Training Details 17

B.3 Datasets 17

C Additional Results **19**

C.1 Complementary experiments 19

C.2 Training Performance 21

C.3 Attention weights analysis 23

C.4 Alpha masks analysis 26

D Code, licenses and resources **27**

A Amortized MAP objective

The learning objective used in DeepObjectLog follows a deterministic MAP principle. For each image–label pair (x, y), we seek a single latent explanation z that receives high probability under the model. The ideal objective is therefore

$$\mathcal{J}_{\mathrm{MAP}}(\theta_p) = \sum_{(x,y)\in D} \max_z \log p_{\theta_p}(x, y, z). \tag{8}$$

Equivalently, if we introduce one latent variable z_k for each training example (x_k, y_k), this can be written as

$$\max_{\theta_p, \{z_k\}_{k=1}^K} \sum_{k=1}^{K} \log p_{\theta_p}(x_k, y_k, z_k). \tag{9}$$

We now derive the form of the score $p_{\theta_p}(x, y, z)$. Starting from the full joint model,

$$p_{\theta_p}\big(y, x, \mathbf{s}, \boldsymbol{\beta}, \mathbf{o}, \mathbf{c}, z\big), \tag{10}$$

the score of a candidate latent explanation z is obtained by marginalizing the slot variables $\mathbf{s}$, the objectness scores $\boldsymbol{\beta}$, the objectness assignments $\mathbf{o}$, and the class assignments $\mathbf{c}$:

$$p_{\theta_p}(x, y, z) = \sum_{\mathbf{o},\mathbf{c}} \int d\mathbf{s}\, d\boldsymbol{\beta}\; p_{\theta_p}\big(y, x, \mathbf{s}, \boldsymbol{\beta}, \mathbf{o}, \mathbf{c}, z\big). \tag{11}$$

In our deterministic setting, the conditional distributions over $\mathbf{s}$ and $\boldsymbol{\beta}$ are Dirac distributions centered at the outputs of the neural modules:

$$\mathbf{s} = \mathbf{s}^\star(z), \qquad \boldsymbol{\beta} = \boldsymbol{\beta}^\star(z).$$

Using the sifting property of Dirac distributions, these variables collapse out of Equation 11. The remaining discrete variables are marginalized by the probabilistic logic layer. This yields

$$p_{\theta_p}(x, y, z) = p(z)\, p_{\theta_p}\big(x \mid \mathbf{s}^\star(z), \boldsymbol{\beta}^\star(z)\big)\, p_{\theta_p}\big(y \mid \mathbf{s}^\star(z), \boldsymbol{\beta}^\star(z)\big). \tag{12}$$

The task likelihood is computed by summing over the discrete objectness and class variables:

$$p_{\theta_p}\big(y \mid \mathbf{s}, \boldsymbol{\beta}\big) = \sum_{\mathbf{o},\mathbf{c}} p_L(y \mid \mathbf{o}, \mathbf{c}) \prod_i p(o_i \mid \beta_i) p_{\theta_p}(c_i \mid s_i, \beta_i). \tag{13}$$

Thus, although the continuous part of the model is treated deterministically, the symbolic prediction remains probabilistic and marginalizes over possible objectness and class assignments.

Taking logs in Equation 12, the score of a latent explanation becomes

$$\log p_{\theta_p}(x, y, z) = \log p_{\theta_p}\big(y \mid \mathbf{s}^\star(z), \boldsymbol{\beta}^\star(z)\big) + \log p_{\theta_p}\big(x \mid \mathbf{s}^\star(z), \boldsymbol{\beta}^\star(z)\big) + \log p(z). \tag{14}$$

This is the log joint density of the observed image, the observed label, and the candidate latent explanation. For fixed x and y, maximizing this score with respect to z is equivalent to MAP inference:

$$\arg\max_z p_{\theta_p}(z \mid x, y) = \arg\max_z p_{\theta_p}(x, y, z), \tag{15}$$

because

$$p_{\theta_p}(z \mid x, y) = \frac{p_{\theta_p}(x, y, z)}{p_{\theta_p}(x, y)},$$

and the denominator does not depend on z.

However, solving this MAP problem independently for every training example would require an inner optimization loop. We therefore amortize this step with a deterministic encoder q_ϕ, replacing the per-example MAP variable by

$$z \simeq q_\phi(x).$$

The resulting amortized MAP objective is

$$\mathcal{J}_{\mathrm{AMAP}}(\theta_p, \phi) = \sum_{(x,y)\in D} \log p_{\theta_p}(x, y, q_\phi(x)). \tag{16}$$

Expanding the score gives

$$\mathcal{J}_{\mathrm{AMAP}}(\theta_p, \phi) = \sum_{(x,y)\in D} \Big[\log p_{\theta_p}\big(y \mid \mathbf{s}^\star(q_\phi(x)), \boldsymbol{\beta}^\star(q_\phi(x))\big) + \log p_{\theta_p}\big(x \mid \mathbf{s}^\star(q_\phi(x)), \boldsymbol{\beta}^\star(q_\phi(x))\big) + \log p(q_\phi(x)) \Big] \tag{17}$$

For fixed model parameters, the amortized objective is bounded above by the ideal per-example MAP objective:

$$\log p_{\theta_p}(x, y, q_\phi(x)) \leq \max_z \log p_{\theta_p}(x, y, z). \tag{18}$$

A.1 Prior and Activation Regularization

The term $\log p(q_\phi(x))$ is the prior evaluated at the amortized latent explanation. It regularizes the encoder output.

Assuming a standard isotropic Gaussian prior,

$$p(z) = \mathcal{N}(0, I),$$

we have

$$\log p(z) = -\frac{1}{2}\|z\|^2 + \text{const}.$$

Substituting $z = q_\phi(x)$ gives

$$\log p(q_\phi(x)) = -\frac{1}{2}\|q_\phi(x)\|^2 + \text{const}.$$

Therefore, maximizing the prior term is equivalent, up to an additive constant, to minimizing

$$\frac{1}{2}\|q_\phi(x)\|^2.$$

We refer to this as activation regularization because it penalizes the magnitude of the encoder output rather than the model parameters.

B Experimental details

B.1 Architectural details

All architectures and code used in this study were implemented in Python using PyTorch, with DeepLog [20] employed for the symbolic integration. We provide here the architectural details of the baseline models and our proposed method used in the experiments.

CNN The CNN baseline consists of a convolutional encoder followed by a multi-layer perceptron (MLP) classifier over the target classes. The encoder comprises a sequence of convolutional layers with kernel size 5, each followed by a ReLU activation. The first convolutional layer uses a stride of 2 to reduce spatial resolution, while the remaining layers use a stride of 1. The output of the final convolutional layer is processed by an Adaptive Average Pooling layer. The resulting feature vector is then passed to an MLP with hidden dimension 64, which outputs the class logits.

Slot Attention For Slot Attention, we adopt the architecture from the original paper. Both the slot dimension and the embedding dimension are set to 64, and the number of slot attention iterations is 2. For the final classification, we apply an MLP over the summed slot embeddings. The decoder module follows the design proposed in the original Slot Attention paper and several PyTorch implementations are available online [1].

[1] https://github.com/amazon-science/AdaSlot

MESH The MESH baseline builds on the Slot Attention architecture, with the addition of mesh-based interactions [37] [2]. Both the slot dimension and the embedding dimension are set to 64, and the number of slot attention iterations is 2. The remaining hyperparameters match those in the original MESH repository. For the final classification, as in Slot Attention, we apply an MLP over the summed slot embeddings. The decoder employed matches that used in the Slot Attention baseline.

Transformer predictor (Tf) The Transformer predictor shares the encoder of the MESH baseline, but replaces the MLP over the summed slot embeddings with a Transformer encoder that attends over all slots. Each slot is linearly projected to dimension 128, and a learnable [CLS] token is prepended to the sequence. We use no positional embeddings, so the predictor is invariant to the order of the slots and accepts any number of them. The sequence is processed by 3 pre-norm Transformer encoder layers with 4 attention heads, feed-forward dimension 256, GELU activations and dropout 0.1. The output of the [CLS] token is layer-normalized and passed to an MLP with hidden dimension 128, which outputs the class logits.

CoSA The CoSA implementation code is provided by the authors of the paper and is available online, even if not all hyperparameters of the model were explicitly provided in the released code [3]. Based on reported results, we selected the Gumbel variant with 3 slot attention iterations as the best-performing configuration and used it across all experiments. For the MM-A and Poker Rules tasks we adapted the reasoning classifier described in their work to fit the specific tasks considered in our evaluation (i.e. setting the correct amount of properties and classes).
This model required structural modifications for the CLEVR-A task due to the two input images used to calculate the sum. To adapt the model, we used the Slot Attention mechanism on each image separately and added the resulting slots to feed everything into a single classification head. As the reasoning classifier described in their paper cannot be used for this task without modification, we replaced it with a simple MLP classifier.

NeSy baseline Following the idea exposed in Misino et al. [24], in particular for the Label Classification task, the NeSy baseline uses a single CNN encoder backbone to process the input and leverages the DeepProbLog interface, mirroring our own method for fair comparison. To enable classification over multiple entities, the architecture includes two MLPs per possible object in the training set—one for estimating objectness and one for classification. All hyperparameters of the encoder align with those used in the CNN baseline. The symbolic part is interfaced through DeepLog[20], where each output of the objectness and classifiers represents a neural predicate. These predicates are then utilised in combination to construct the rules employed for the final classification of the task. An example pseudo-code for the MultiMNIST-addition task is available in Table 3.

DINOSAUR We include DINOSAUR [25] as a strong object-centric baseline designed for real-world images. We re-implemented DINOSAUR and use the variant based on a pretrained DINO backbone to obtain high-quality visual features. For the purely neural baseline, we follow the same classification strategy adopted for Slot Attention, applying an MLP over the aggregated slot representations. In our method, instead, the extracted slots are directly integrated into the neurosymbolic pipeline.

Our model Our proposed method was implemented using both the standard Slot Attention mechanism and the MESH variant. As the latter showed slightly better average performance, we report results for the MESH-based implementation. The hyperparameters follow those recommended in the MESH paper, where maintaining a low number of slot iterations (2) and increasing the number of mesh iterations (5) proved beneficial. Unlike the NeSy baseline, our method employs only two MLPs in total (one for objectness and one for classification), which are applied separately to each extracted object representation. To condition both the class classifier and the reconstruction, we follow the Zero Slot Strategy from Adaptive Slot Attention: the objectness probability is multiplied with the slot before it is passed to the classifier and decoder (See the next subsection for details.). The decoder architecture is the same as used in Slot Attention and the symbolic interface is the same as described in the NeSy baseline.
As with CoSA, this model required some structural modifications for the CLEVR-A task due to the

[2] https://github.com/davzha/MESH
[3] https://github.com/koriavinash1/CoSA

two input images. We simply adapted the model by using the Slot Attention mechanism separately on the two images. All classification results were then fed into the same logic circuit. Please note that, for this specific task, adding a classification head for the object properties is unnecessary, as the objectness head is sufficient.

Soft masking Our implementation of soft masking draws inspiration from the Zero Slot strategy introduced in Adaptive Slot Attention [10]. In their formulation, each slot representation s_i is modulated by its associated objectness score (our β_i):

$$\tilde{s_i} = s_i \cdot \beta_i$$

However, unlike their approach, we do not apply a Gumbel-Softmax to discretize the objectness scores.

On the number of slots N. In our architecture, the hyperparameter N serves as an upper bound on the number of representable objects. During training, we set N equal to the maximum number of objects present in the training scenes (e.g., $N = 3$ for MultiMNIST-Addition). For extrapolation to 4 or 5 digits, we increase N at inference time to provide the necessary capacity. Interestingly, while initializing with a very high N expands the combinatorial search space, our end-to-end logical signal mitigates this: if $N = 9$ but the task is a sum of 9, the logic program naturally prunes impossible configurations (e.g., two 9s). Furthermore, the perceptual module and symbolic prior collaborate to suppress spurious slots via the objectness variable. For example, a "0" in MultiMNIST does not influence the sum, yet our model correctly identifies it with high objectness and the correct class. The perceptual reconstruction ensures the "0" is recognized as an entity, while the logic ensures the other digits align with the required sum, proving that extrapolation is driven by genuine scene decomposition rather than slot over-provisioning. This is supported by our interpolation experiments (Table 5), where the model dynamically adapts to an unseen number of objects within a fixed N. Finally, to rigorously evaluate this synergy, our concept accuracy relies on an exact match (subset accuracy) metric: the model must correctly predict the objectness and class for *every* slot in the image, otherwise the entire prediction scores zero.

Experiment with object-level supervision. In the experimental setting designed to evaluate varying levels of object-level supervision (i.e., the COCOLogic benchmark), we adapt the training objective for both our proposed method and the DINOSAUR baseline by utilizing a combined loss function. Specifically, the total loss is formulated as the sum of the primary task-level loss, the reconstruction loss and an auxiliary object-level loss. Because the object-centric slots are inherently unordered and permutation-invariant, following common practices, we compute the object-level loss by first applying the Hungarian algorithm to establish an optimal bipartite matching between the predicted slots and the available ground-truth object labels. Once the optimal assignment is determined, a standard classification loss (e.g., cross-entropy) is applied to the matched slots. This symmetric formulation ensures that both our method and the baseline can effectively leverage partial explicit object annotations while continuing to rely on the global task loss for the overarching reasoning objective.

B.2 Training Details

We run all our experiments on a Nvidia L40S 48GB GPU card with AMD EPYC 9334 CPU and 256GB RAM.

We train all models using the AdamW optimizer with a learning rate of 1e-4 and weight decay of 1e-4. Neural baselines are trained for 100 epochs with a batch size of 64. Neurosymbolic models are trained longer, depending on the dataset: 400 epochs with batch size 64 for MultiMNIST and PokerRules, and 200 epochs with batch size 32 for CLEVR-Addition (the same batch size is also used for CoSA on CLEVR-Addition). In all cases, the best model is selected based on validation performance. All experiments are run with multiple seeds, and we report mean and standard deviation across runs in the tables.

B.3 Datasets

MultiMNIST. We build upon an existing MultiMNIST generator [32], modifying it to better suit the requirements of our experimental setup. In particular, we adapted the generation process to support distant supervision, where the model is trained only on the sum of the digits present in each

image. The training set includes 30,000 images, and the validation and test set both include 3,000 images. Digit appearances are balanced to ensure uniform distribution across the dataset and we make sure that all possible output sums (generable from possible compositions) are seen during training. All images are normalized to the range $[-1, 1]$. While the models are trained only on the total sum of the digits, we retain the individual digit annotations to support the evaluation of concept-level accuracy.

Table 3: Example of predicates and background knowledge for MultiMNIST. This is a simplification of the version employed in the experiments, wherein a maximum of two digits are considered. It is important to note the role of $isobj_tmp$ (objectness) in this process, as it facilitates the selection of elements to be added, thereby determining the final result, and those to be excluded.

$$
\begin{array}{l}
\% \;\; Neural \;\; predicates \\
classifier_slot0(X, N) :: digit_tmp(X, 0, N) : -between(0, 9, N). \\
classifier_slot1(X, N) :: digit_tmp(X, 1, N) : -between(0, 9, N). \\
isobject_slot0(X) :: isobj_tmp(X, 0). \\
isobject_slot1(X) :: isobj_tmp(X, 1). \\
\\
\% \;\; Conditioned \;\; digit \;\; definition \\
digit(X, ID, Y) : - \; isobj_tmp(X, ID), \; digit_tmp(X, ID, Y). \\
\\
\% \;\; Rules \\
addit(X, ID0, SumIn, SumOut) : - \; isobj_tmp(X, ID0), digit(X, ID0, C), \\
\qquad\qquad\qquad\qquad\qquad\qquad SumOut \; is \; SumIn + C. \\
addit(X, ID0, ID1, SumIn, SumOut) : - \; isobj_tmp(X, ID1), \; digit(X, ID1, C), \\
\qquad\qquad\qquad\qquad\qquad\qquad Y \; is \; SumIn + C, \; addit(X, ID0, Y, SumOut). \\
addit(X, ID0, SumIn, SumIn) : - \; not(isobj_tmp(X, ID0)). \\
addit(X, ID0, ID1, SumIn, SumOut) : - \; not(isobj_tmp(X, ID1)), \\
\qquad\qquad\qquad\qquad\qquad\qquad addit(X, ID0, SumIn, SumOut). \\
\\
addition(X, Z) : -addit(X, 0, 1, 0, Z).
\end{array}
$$

$$
\begin{array}{l}
\% \;\; Query \\
? - \; addition(input, Z)).
\end{array}
$$

PokerRules. The PokerRules dataset is synthetically generated to support relational reasoning from visual input, where the label depends on the combination of card types present in the scene. The training set includes 20,000 images, with 2,000 images each for validation and testing. Each image contains between 1 and 4 playing cards (5 for the extrapolation setting) positioned along a horizontal line with slight random perturbations to their location. The cards are drawn from a fixed suit (hearts) and include the ranks: ten, jack, queen, king, and ace. Images are normalized to the range $[-1, 1]$. The training set includes five classes: nothing (1–4 cards), pair (2–4 cards), two pairs, poker and straight (4 cards). The appearance of each class is balanced across training samples. To evaluate the model's ability to generalize, we introduce three test sets. In the first, we add the unseen class *three of a kind*, an out-of-distribution composition where three identical cards appear together (with 3 or 4 cards in total). In the second, we test *poker-5*, where a standard poker hand is accompanied by an additional unrelated card. This introduces extrapolation to a higher number of objects (five), while keeping the underlying class structure familiar. Finally, in the third test set we evaluate *full house*, a completely novel hand type composed of three cards of one rank and two of another.

CLEVR. Starting from the CLEVR dataset [14], we construct a new dataset tailored for the CLEVR-Addition task. Images are paired randomly, and each pair is assigned as label the sum of the number of objects contained in the two images. All images are normalized to the range $[-1, 1]$. For training and validation, we use the CLEVR training split, restricting to images with at most six objects. This results in 26,132 images, with the last 10,000 reserved for validation. All test sets are derived from the CLEVR validation split to prevent data leakage. We construct three test sets:

- In-distribution: pairs of images containing at most six objects, following the same protocol as for training.
- Extrapolation (same class space): pairs including images with seven objects, but only those combinations whose total sum remains within the training range (i.e., ≤ 12). This ensures neural baselines can still attempt the task.
- Extrapolation + OOD classes: pairs combining images with six and seven objects, producing sums beyond the training range and thus introducing new, unseen classes.

COCOLogic We use the COCOLogic dataset introduced in [31]. To ensure consistency with the original setup, we generate the dataset using the scripts provided by the authors and verify that the resulting class distributions match those reported in the paper. COCOLogic consists of images from COCO grouped into ten classes defined by logical rules over object presence, absence, and counts (e.g., combinations or exclusions of categories such as *person*, *car*, or *dog*). Each image satisfies exactly one rule, resulting in mutually exclusive and semantically meaningful classes. This structure makes the task inherently compositional and requires reasoning over object-level information rather than relying on global visual cues.



Figure 5: Example images with their corresponding classes from the COCOLogic dataset.

C Additional Results

C.1 Complementary experiments

This section presents complementary experiments on the MultiMNIST, PokerRules and CLEVR datasets.

Comparison with foundation models. To better understand how foundation models compare in our setting, we conducted an experiment using two famous Vision-Language Models: "LLaVA-1.5-7B-hf" and "Qwen2.5-VL-7B-Instruct", on our MultiMNIST-Addition task including only out-of-distribution test set for fair comparison with our model. The results of the experiment are available in Table 4. The VLMs were prompted with natural language queries describing the task and the possible solutions (i.e. how many digits can be in the image). Despite their general capabilities, we observe that they struggled to reliably solve the task.

Table 4: Accuracy results of our model and two Vision-Language Models on the MultiMNIST-addition task.

Model	OOD Compositions	Extrap. (4 digits)	Extrap. (5 digits)
Llava-1.5-7b-hf	27.43	2.73	1.87
Qwen2.5-VL-7B-Instruct	59.93	28.67	14.37
Ours	**90.00**	**69.73**	**44.06**

All experiments were conducted under controlled and fixed settings (e.g., decoding temperature, and evaluation protocols) to ensure full reproducibility of results.

The prompts used for the experiments for each of the three test sets are reported below.

- **OOD compositions**: *"What is the value of the sum of the digits in this picture? An image with only a black background is just a 0. Reply only with the sum value. Note that the sum values can be between 0 and 27 and the image contains between 0 and 3 digits."*
- **Extrapolation (4 digits)**: *"What is the value of the sum of the digits in this picture? An image with only a black background is just a 0. Reply only with the sum value. Note that the sum values can be between 0 and 36 and the image contains between 0 and 4 digits."*
- **Extrapolation (5 digits)**: *"What is the value of the sum of the digits in this picture? An image with only a black background is just a 0. Reply only with the sum value. Note that the sum values can be between 0 and 45 and the image contains between 0 and 5 digits."*

Interpolation. In Table 5, we report results from the interpolation setting on MultiMNIST, where models are trained on images containing 0, 1, or 3 digits and tested on samples containing exactly 2 digits. As with other setups, our model demonstrates superior generalization compared to all baselines. Note that, unlike the extrapolation setting, the NeSy baseline is applicable here and provides out-of-distribution predictions that are better than the neural baselines. Additionally, the interpolation setup is particularly challenging, as the model is exposed to fewer digit configurations overall (only three instead of four, as in the standard setting), which increases the difficulty of identifying structural patterns. This suggests that providing a broader range of digit counts during training (e.g., including 4 or 5 digits) could further enhance interpolation performance.

Table 5: Accuracy results on the MultiMNIST-addition interpolation setting.

Model	Test (0,1,or 3 digits)	Interpolation (2 digits)
CNN	76.50±0.70	24.90±0.14
SA	98.40±0.70	10.05±3.74
MESH	**99.05**±0.07	14.60±5.09
CoSA	87.23±5.85	22.53±20.16
NeSy	94.35±0.49	33.25±7.42
Ours	89.80±6.34	**62.30**±19.44

Supplementary results for task generalization. Table 6 presents results in an extrapolation setting involving both seen and unseen classes. The rightmost columns focus on unseen classes, where the model is tested on images containing 4 or 5 digits with sum values beyond those encountered during training. This scenario highlights the adaptability of our approach: none of the baselines can be applied here, as they lack either structural or task generalization capabilities. Our model, instead, handles both simultaneously.

Table 6: Accuracy results on the MultiMNIST-addition extrapolation settings. The right columns show the case where both the class and the number of digits are out of distribution.

	Extrapolation		OOD class extrapolation	
	4 digits	5 digits	4 digits	5 digits
CNN	22.16±0.90	13.16±0.96	-	-
SA	36.23±4.44	9.76±6.11	-	-
MESH	37.50±1.15	12.00±0.51	-	-
CoSA	52.20±14.01	25.06±12.82	-	-
Ours	**69.73**±10.74	**44.06**±3.85	**70.50**±9.46	**46.66**±5.14

Concept accuracy. In Table 7, we provide additional metrics for the PokerRules task, including concept accuracy and card-count prediction accuracy on both the in-distribution test set and the test set with an out-of-distribution class. Unlike MultiMNIST, this task lacks an inductive bias strong enough to help the model identify individual cards using only distant supervision. This is a property of the program rather than of the perceptual module. The PokerRules rules only use *same class* and *different slots*, which are invariant under any permutation of the five ranks, so the task label determines the partition of the cards into equality classes but not which rank each class carries. Consistently, adding object-level supervision on a random 1% of the training set breaks this symmetry and raises concept accuracy to 99.6%. Still, our model achieves strong accuracy in estimating the number of cards per image. We also report a best-matching concept accuracy, computed by mapping the predicted card classes to the most plausible configuration of symbols. The results reflect the increased complexity of this task, yet our model still outperforms the neurosymbolic baseline.

We also note that we tested our model's concept accuracy in the CLEVR-Addition task, achieving an excellent score of 96.28%±0.55. Here, the model, starting from the addition value, was able to understand the number of objects present in a single image.

Table 7: Concept accuracy results for PokerRules

	Task accuracy		Concept accuracy		Objects number accuracy	
	Test	OOD class	Test	OOD class	Test	OOD class
NeSy	80.23±2.11	0.46±0.05	18.63±1.37	0.13±0.05	80.20±5.82	76.53±24.88
Ours	**97.90**±1.17	**72.23**±16.72	**32.00**±5.88	**29.26**±13.95	**81.60**±2.78	**84.13**±6.60

Table 8: Results for CLEVR-A dataset

CLEVR-A	Test	Extrapolation: 7 objects	
			OOD Class
MESH	**96.97**±0.35	0.50±0.30	-
CoSA	88.79±3.95	3.44±1.69	-
Ours	93.12±0.56	**59.81**±12.45	**28.57**±3.71

Ablation study. In Table 9, we present an ablation study on the MultiMNIST-Addition task to assess the contribution of key components in our architecture. Each variant highlights the role of architectural choices in supporting accurate object-level reasoning and task performance.

Table 9: Ablation results on the MultiMNIST-Addition task. Specifically, we evaluate performance under the following modifications: (1) replacing the object-centric encoder MESH with the standard Slot Attention mechanism; (2) using a single classifier that treats non-objectness as an additional class (common approach on object-centric learning), instead of our two-head design with separate objectness and class classifiers; (3) removing the conditioning of the class classifier on the objectness score; (4) excluding the final decoding phase and corresponding reconstruction loss during training. Regarding this last point, a recent work [22] has shown that incorporating perceptual cues through reconstruction can help align perception with symbolic reasoning, discouraging reliance on irrelevant features and preventing to fall in reasoning shortcuts.

	Task Accuracy		Concept Accuracy	
	Test	OOD comps	Test	OOD comps
(1) Original SA	76.91±3.25	71.70±3.53	30.70±5.70	5.71±0.14
(2) One classifier only	93.49±0.70	64.06±19.09	15.05±9.19	20.49±14.14
(3) No class conditioning	92.51±0.28	81.84±3.46	66.77±14.96	41.16±12.10
(4) No reconstruction	**95.75**±1.76	89.55±2.19	64.64±2.33	52.20±11.59
Ours	94.26±2.00	**90.00**±3.01	**85.16**±2.45	**65.46**±5.70

C.2 Training Performance

The processing speed in terms of iteration/sec for our model and the baselines is tabulated in Table 10. It is to be noted that speed might differ slightly with respect to the considered system and the background processes.
Table 11 reports the total training time for all models in the main experiments, measured with a single seed. The project required additional computing power when testing different variations of our model and hyperparameter tuning. To mitigate the computational overhead of probabilistic reasoning, our implementation utilizes the KLay optimization framework from [18].

Table 10: Training speed (iterations per second, it/s) of the evaluated models across three experimental settings.

Methods (↓), Batch size (→)	**MM-A 64**	**PokerRules 64**	**CLEVR-A 32**
CNN	24.60	6.97	N.A.
SA	14.06	3.72	N.A.
MESH	11.06	3.12	1.98
CoSA	6.76	3.65	2.15
NeSy	12.43	3.23	N.A.
Ours	7.66	2.1	1.75

Table 11: Total training time per epoch (hours:minutes) for the evaluated models across the three experimental settings.

Methods (↓),	**MM-A**	**PokerRules**	**CLEVR-A**
CNN	∼0h 31min	∼1h 14min	N.A.
SA	∼0h 55min	∼2h 20min	N.A.
MESH	∼1h 10min	∼2h 47min	∼6h 11min
CoSA	∼1h 55min	∼2h 22min	∼5h 03min
NeSy	∼4h 11min	∼10h 46min	N.A.
Ours	∼6h 47min	∼16h 33min	∼9h 20min

Table 12: Total training time of the evaluated models in the COCOLogic setting.

Methods (↓), Supervision (%) (→)	**0**	**30**	**60**	**100**
DINOSAUR	∼5h 27min	∼6h 51min	∼11h 03min	∼15h 40min
Ours	∼6h 44min	∼8h 00min	∼14h 33min	∼19h 47min

In Table 13, we evaluated the scalability of our method against that of the MESH baseline in a MMA setup during training as the number of objects increases. Our model experiences a slowdown due to the complexity of the problem but we consider this as a genuine limitation, but also as the source of most of our advantages. It is exactly the marginalization over objectness and class assignments that lets a single program operate over an unknown number of objects, resolves object assignments without set matching, and provides the task-level signal that drives out-of-distribution generalization. The cost is also not intrinsic to the formulation as the symbolic layer enters the objective only through the marginal of Eq. 3, so approximate inference methods can be substituted without touching perception, decoder, objective or rules. To demonstrate this, we replace exact symbolic inference with a neural approximation inspired by A-NeSI [33]. A-NeSI learns a neural prediction model that approximates the weighted model-counting operation, allowing gradients to be propagated without explicitly performing exact inference. In our setting, we exploit the finite symbolic state space to generate all possible combinations of objectness and class assignments, evaluate the program on each configuration, and use these examples to train an MLP to approximate the corresponding reasoning function. This experiment provides a proof of concept that the inference bottleneck can be mitigated independently of the formulation itself, with the resulting approximation proving more efficient than the baseline in some settings.

Table 13: Number of inference iterations per second for the MM-A task (batch size 64) as the number of objects increases. We write timeout (TO) when the number of iterations per second drops below 1.

Methods (↓), Num. of objects (→)	**1**	**2**	**3**	**4**	**5**	**6**	**10**	**20**
MESH	24.04	16.61	12.27	9.88	6.70	5.68	3.37	1.04
Ours	16.46	13.24	7.69	4.06	2.74	1.70	TO	TO
Ours w A-NeSI	20.54	18.00	14.34	12.67	10.78	9.21	4.34	1.19

C.3 Attention weights analysis

We present the attention weights produced by our neurosymbolic model and by MESH on discriminative tasks trained with distant supervision. These weights are obtained from the last iteration of the slot attention module and visualized as spatial maps over the input. The comparison includes in-distribution, out-of-distribution, and extrapolation settings. Our model's attention masks clearly show its ability to separate and individually attend to the objects in the scene. In contrast, the purely neural MESH model appears less capable of capturing the underlying compositional structure of the image, often focusing on the overall configuration rather than the individual objects relevant to solving the task.

Analyzing the correspondence between attention weights and the predicted objectness and classification values for each slot, we observe that slots attending to multiple regions simultaneously tend to receive low objectness scores, whereas slots with sharp, localized attention typically exhibit high objectness confidence.

To support these claims we also provide quantitative results in table 14. While our datasets do not naturally include ground-truth segmentation masks for standard metrics like FG-ARI, we provide a quantitative evaluation of the masks using Count Mean Absolute Error. This metric measures $|n_active_slots - n_gt_objects|$, where a slot is considered "active" if more than 1% of its pixels exceed a threshold of 0.1. Our model outperforms the neural baseline across all three datasets.

Table 14: Count Mean Absolute Error for the number of active slots in the three datasets. The comparison involves our method and the baseline MESH. The lower the better.

	MM-A	PokerRules	CLEVR-A
MESH	0.998 ± 0.011	1.156± 0.050	2.057± 0.037
Ours	**0.733**± 0.170	**0.498**± 0.072	**1.287**± 0.188

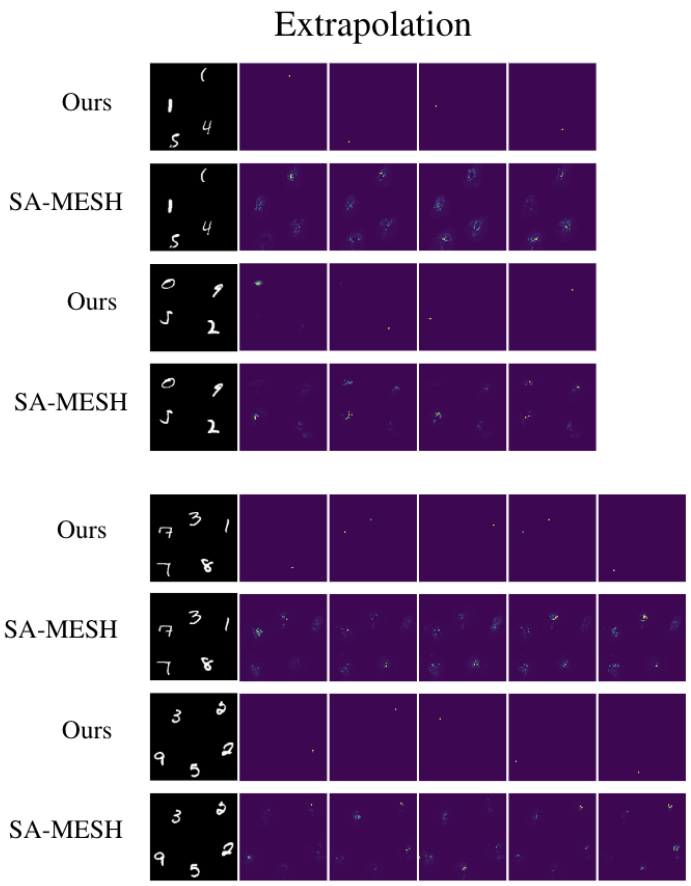

Figure 6: Attention weights of the Slot Attention mechanism for the MultiMNIST dataset.

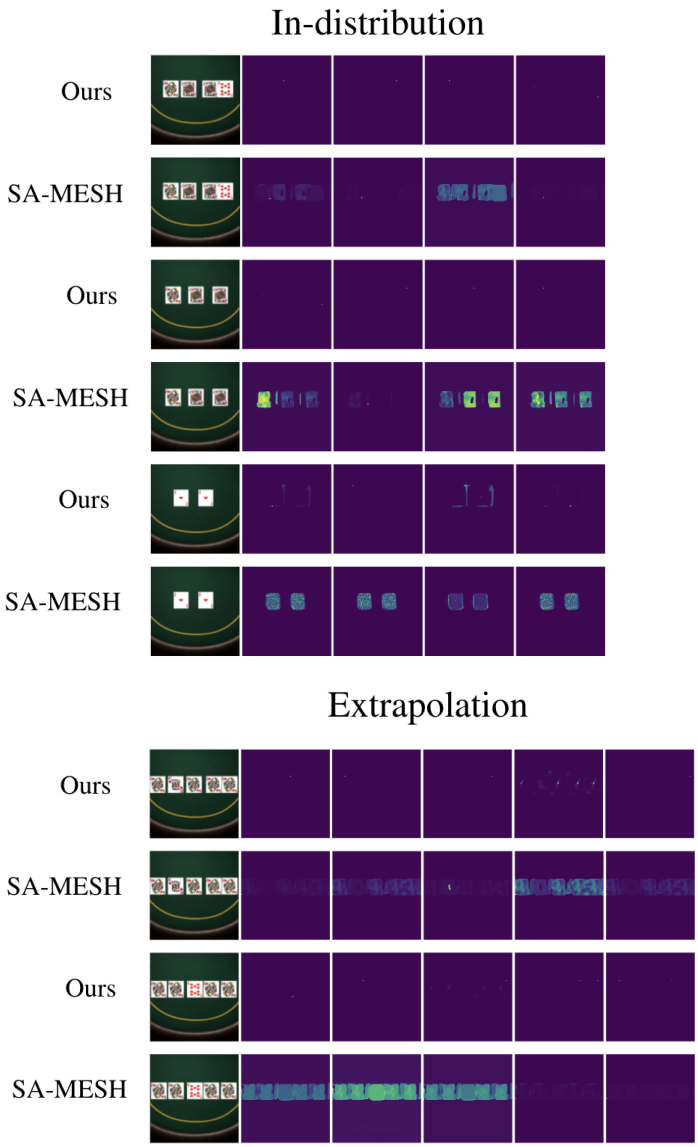

Figure 7: Attention weights of the Slot Attention mechanism for the PokerRules dataset. As the images are larger than those in MultiMNIST, it is harder to see where the attention weights are highest. However, the model is exhibiting the same behaviour as in previous experiments.

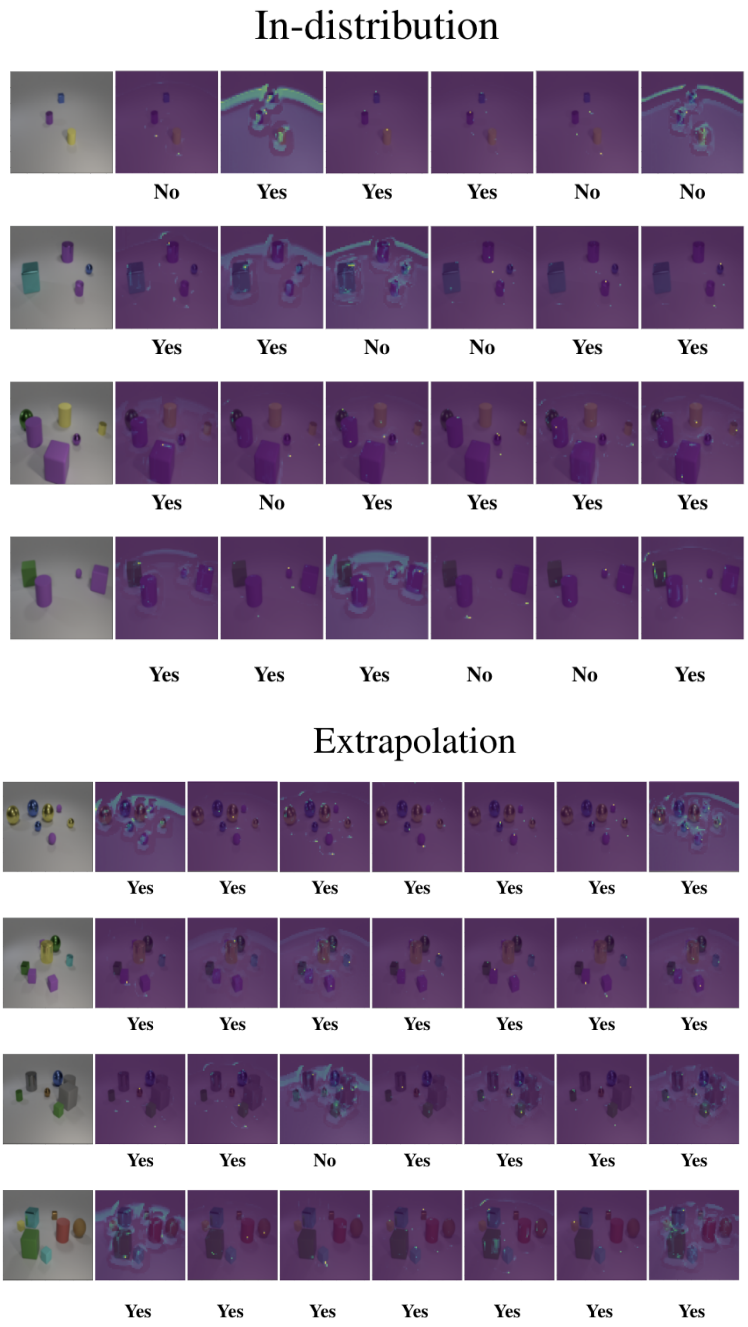

Figure 8: Attention weights of the Slot Attention mechanism trained with our method for the CLEVR dataset. As the images are larger than those in MultiMNIST, it is harder to see where the attention weights are highest. However, the model is exhibiting the same behaviour as in previous experiments. For each slot, the associated objectness score is shown, with slots above 0.5 marked as yes.

C.4 Alpha masks analysis

In this section, we present examples of the alpha masks produced by our model during the reconstruction phase. These masks are primarily useful for reconstruction, while the attention weights are instead used to compute and refine the slot representations. Notably, the attention weights often focus on small regions within objects, whereas the alpha masks tend to approximate the full segmentation

of the objects [37]. For each alpha mask, we also report the corresponding objectness predicted by the model for that slot. The associated score is directly used in the reconstruction phase.

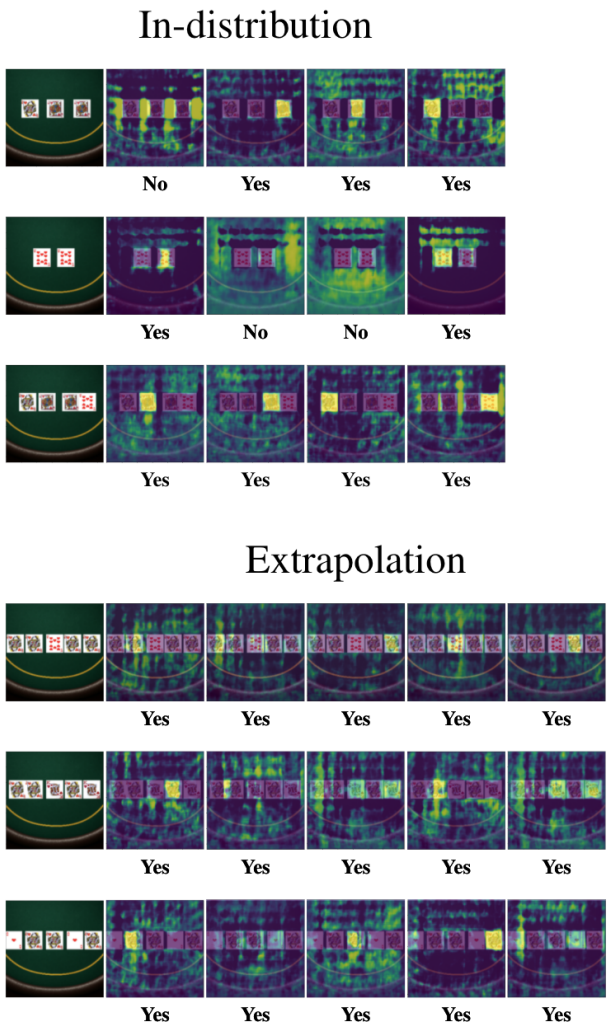

Figure 9: Alpha masks produced by the Slot Attention mechanism trained with our method on the PokerRules dataset. For each slot, the associated objectness score is shown, with slots above 0.5 marked as *yes*.

D Code, licenses and resources

Our code is publicly available under the Apache license Version 2.0 at the following link: https://github.com/StefanoColamonaco/Neurosymbolic-OCL

All the data we used to build our datasets are freely available on the web with licenses:

- MNIST - CC BY-SA 3.0 DEED,
- Cards Image Dataset - CC0: Public Domain.
- CLEVR - CC BY 4.0
- COCO - CC BY 4.0

The original datasets used as a basis for constructing our synthetic benchmarks are not redistributed. We will make the generation code of our datasets available for reproducible experiments.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 structure of our model (described in Section 3), and the experiments closely follow the claimed contributions in both abstract and introduction.

   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: This is done in our conclusion (Section 7).

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 formulae in the paper are correctly referenced and, where necessary, further explained in Appendix A.

   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: Details for the reproducibility of the experiments are provided in Appendix B.

   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: A link to the code is available in Appendix D.

   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: Training and test details are available in Section 5 and Appendix B.

   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 provide the standard-deviations over multiple seeds in all our tables.

   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 reported the details of the computer resources exploited in the experiments in Appendix B.

   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: Our research does not raise any ethical issue.

   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: Our conclusion clearly states the broader impact of our work with reference to a more robust and interpretable AI.

    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: The paper poses no such risks.

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: We mentioned and respected all the licences of data in Appendix D.

    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: [Yes]

    Justification: Documentation for the code and dataset generators is provided in a README file within the corresponding repository. See the answer to 'Open access to data and code'.

    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: The 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: The 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 core method development in this research does not involve LLMs as any important, original, or non-standard components. However, some VLMs are used for comparison in one of the experiments.

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.