The Challenge of Learning Symbolic Representations

Luca Salvatore Lorello[1,2,*], Marco Lippi[1]

[1]DISMI, University of Modena and Reggio Emilia, Italy
[2]University of Pisa, Italy

Abstract
Learning abstract representations from perceptual stimuli is a natural task for humans, but a real challenge for AI systems. In the vast majority of cases, in fact, systems that have to deal with symbol manipulation, like in reasoning or planning, do not need to also learn the symbols they operate on, but these are typically assumed to be given by a supervisor. Moreover, symbolic manipulation often implies compositional properties which are difficult to learn. In this paper, we consider the problem of learning symbolic representations that can be associated to abstract concepts, to be used in a variety of downstream tasks, and we analyze the many challenges related to this important problem, with a particular emphasis on paving the way towards composable symbolic representations learned by neural networks. We identify key properties for symbolic composable representations, such as non-ambiguity and purity, that suggest the need for different types of regularizations within the learning process, as well as new metrics for their evaluation.

Keywords
Composable symbolic representations, Concept learning, Discrete neural embeddings

1. Introduction

In their everyday activities, humans continuously operate with symbols. Reasoning skills, decision making processes, planning activities, causality modeling, all require symbol formalization and manipulation. AI systems designed to address these tasks typically do not require to also learn the symbolic representations they operate on: symbols, predicates, or concepts are usually known a priori, or given by an external source of information, or by a supervisor [1].

In this paper, we aim to bring the attention of the neuro-symbolic AI community to the problem of learning symbolic representations, a task which has been largely under-explored in the recent literature of this research area. According to Newell and Simon [2], a symbolic system maps abstract entities into symbols that are arbitrary (i.e., whose shapes have no relation with their semantic meaning) and recursively composable (i.e., to produce complex and more general representations from elementary entities). In order to guarantee both properties, a symbol must belong to a discrete set. Learning these symbolic representations from perceptual stimuli is a challenging task. In fact, there exist several works that try to learn discrete representations, like discrete probabilistic latent spaces [3], vector-quantized variational autoencoders [4],

NeSy'23: 17th International Workshop on Neural-Symbolic Learning and Reasoning, July 03–05, 2023, Certosa di Pontignano, Siena, Italy.
*Corresponding author.
luca.lorello@phd.unipi.it (L. S. Lorello); marco.lippi@unimore.it (M. Lippi)
0009-0006-3803-8268 (L. S. Lorello); 0000-0002-9663-1071 (M. Lippi)

or deep hashing methods [5]. Yet, we claim that those approaches do not consider crucial properties that symbolic representations should have, such as non-ambiguity, purity, and especially composability [2]. While the processing phase of the input stimuli is naturally handled by neural architectures, building a differentiable end-to-end approach for discrete, symbolic representations poses a number of challenges, starting from the ill-defined nature of gradient-based techniques [6]. Moreover, we conjecture that, ideally, the learning process that produces symbolic representations should be unsupervised, or at least weakly/partially supervised [7]. This observation raises additional issues that suggest to re-think the whole learning paradigm, and that also advocate the need for novel metrics to be exploited to assess the effectiveness of the learned representations.

The paper is structured as follows. After discussing related works in Section 2, in Section 3 we describe the properties that symbolic representations should have, and we discuss the challenges related to the learning problem, also focusing on the metrics that should be used to evaluate the goodness of the learned representations. In Section 4 we present a preliminary experiment to show the intrinsic difficulties of the considered task. Finally, Section 5 concludes the paper.

2. Related Works

Neural discrete representations have been successfully used for a plethora of tasks. We identified in the literature a few categories of approaches: embedding tables, probabilistic latent spaces encoding discrete probability distributions, deep hashing approaches.

Embedding tables encode a finite (and thus discrete) set of objects with a continuous representation. Vector-quantized variational autoencoders [4] exploit a quantized embedding table, effectively making the representations discrete and achieving outstanding results in image and sound generation tasks. They found that discrete representations are crucial to avoid posterior collapse in variational autoencoders. The categorical reparametrization trick (Gumbel-softmax operator) [8] and its variant, the concrete distribution (Gumbel-sigmoid operator) [9], are continuous relaxations of binary and categorical random variables. Discretization is achieved by annealing a temperature parameter to zero, and it is thus susceptible to a trade-off between the "discreteness" of the representation and the "informativeness" of its gradient.

A notable mention of neuro-symbolic methods expoiting discrete probabilistic latent spaces is Latplan [3], a variational autoencoder capable of generating a PDDL specification, which can then be solved by a traditional symbolic planner, from pairs of images (before and after) demonstrating actions. Specific architectural constraints are enforced in order to guarantee STRIPS-like semantics, so that multiple binary latent spaces can encode a state as a set of fluents which are either true or false, and an action as a set of added and removed fluents.

Deep hashing methods [5] deal with efficient information retrieval, exploiting binary latent representations which are then used to address buckets by means of Hamming distance computations. Research directions in this area mainly focus on non-differentiability issues of discrete representations and the Hamming function, as well as efficiency, in terms of number of training samples, availability of annotations and retrieval performance.

Among neuro-symbolic appraoches, methods feeding combinatorial optimizers with neural information must deal with discrete representations and how to backpropagate through ill-defined gradients. A plethora of solutions [10, 6, 11, 12] have been proposed to define a gradient which is both well-behaved and useful as a learning signal which allows to exploit information from the downstream task of combinatorial optimization. These techniques however are often method-specific and are difficult to generalize to other tasks (e.g., planning or deduction).

There is extensive literature in philosophy of science concerned with the formal definition of desirable properties for symbols and symbolic reasoning. In the work by Taddeo and Floridi [13], for example, a "truly" symbolic system has to perform grounding while satisfying the zero semantic commitment condition (i.e., the agent must be able to develop its own semantics without external guidance). In the work by Santoro et al. [14], properties are instead defined within a meaning-by-convention framework: symbolic behavior is defined as the set of traits allowing an interpreter to operate within a socio-cultural setting. They identify six desirable properties, possessed by human behaviour but not yet by autonomous agents, and postulate that symbolic fluency can arise as emergent behaviour under meaning-by-convention training regimes, exploiting social interactions with humans, mediated by natural language, and large training sets. According to them, symbolic systems should be: receptive, constructive, embedded, malleable, meaningful and graded. While these properties are certainly intuitive, they are difficult to quantify and, therefore, they cannot be exploited to guide the learning process.

In this work, we took inspiration also from classic principles of knowledge organization [15] and knowledge representation [16, 17].

3. How to Learn Symbolic Representations

Symbolic manipulation is almost unanimously assumed to play an important role in reasoning generalization. The binding problem for subsymbolic systems is an open problem concerning the generation and manipulation of symbolic entities (objects) starting from subsymbolic stimuli [18]. Many neuro-symbolic approaches address this problem, by combining symbolic and subsymbolic components, so that manipulation subtasks (notoriously difficult to address by neural networks) are dealt by the former whereas learning and noise-sensitive subtasks (difficult to address symbolically) are solved by the latter. Our idea is to reduce this dichotomy, conferring a reasoning capability to neural networks, so that synergy with symbolic methods can be further strengthened, allowing for interleaved end-to-end neuro-symbolic computation.

3.1. Properties of symbolic representations

Toward this goal, we propose the use of symbolic representations, with the final objective of acting both as an interface between neural and symbolic components, and as computational entities within the neural component. These representations should possess the following desirable properties: (i) non-ambiguity, (ii) purity, (iii) usefulness, (iv) symbol composability, (v) manipulation composability, and (vi) generalizability with respect to different tasks.

Non-ambiguity means that different entities should not be mapped to the same representation, while **purity** is its dual: every entity should have the smallest possible number of representations (ideally only one each). A high degree of non-ambiguity increases downstream performance and a high degree of purity implies a stronger robustness to noise. If both properties are optimal, there is a 1-to-1 mapping between entities and their representations.

Usefulness is the capability of improving performance for a downstream symbolic task, with respect to another representation. According to Newell and Simon [2], a symbolic system should be able to operate with arbitrary symbols. However, this property only works under ideal assumptions of perfect non-ambiguity and purity, whereas imperfect representations may convey information more or less effectively. Moreover, an arbitrary representation does not allow to optimize for performance, preventing the use of heuristics or pruning strategies. We advocate for a case-by-case evaluation of usefulness, as tasks with a large reasoning component will arguably benefit more from a concise representation, while tasks pivoting around a strong perceptual component may benefit more from redundant representations, robust to noise. It is also important to note that usefulness is necessarily a relative metric, subject to an absolute performance: a small usefulness with respect to a large performance may be more beneficial than a great usefulness associated with low performance.

Symbol composability is the possibility of creating complex constructs (e.g., sequences, expressions, graphs, etc.) from atomic symbols and it is commonly accepted as the key component of any form of reasoning. As both perceptual information and underlying semantics are seldom "flat", it is often necessary to encode hierarchies of entities. Grounding complex structures can be addressed by a wide spectrum of strategies, ranging from grounding atoms into a fixed structure, to learning part-whole hierarchies from perceptual stimuli [18]. Since different downstream tasks require different abstractions, we advocate for a semi-quantitative evaluation of symbol composability, to determine whether a given technique is beneficial, while acknowledging the difficulty in devising a quantitative metric.

Manipulation composability is the capability of deriving a new symbol by performing a sequence of operations to an existing symbol. There are many reasons justifying the need of manipulating entitites in a learnable way. To give some examples: the initial encoding may be at the "wrong" abstraction level (thus requiring a specialization/generalization operation), there may be the need to enforce a specific property (e.g., when encoding the set of natural numbers, it can be beneficial to enforce that each number is the successor of some other number), incremental construction (like contextual embeddings are computed in a transformer), denoising a representation, and so on. Although distinct concepts, manipulation composability is strongly related with the principle of elaboration tolerance [19] in knowledge representation, as it provides a mechanism to alter (or, conversely, guarantee invariance) a representation in response to external conditioning. As manipulation composability paves the way for a recursive processing, it also provides enough expressive power to allow a certain degree of reasoning capability, thus, just like symbol composability, it is important to quantify such capability.

Other properties are more application-specific, and thus we leave their analysis to future research, among these, **generalizability** (the capability of reusing the same representations for different downstream tasks) is of particular importance, while others are compactness, monotonicity, invertibility (or de-composability), and so on.

Existing representations present some, but not all of these properties. One-hot encoding is the most pure and general representation, however it is also arbitrary (unless it is relaxed into a continuous softmax, it cannot propagate gradients) and non-compact. Manipulation composability is extremely limited, as it can only be performed in terms of permutations, and symbol composability is non-trivial. Continuous embeddings are useful and general, but they lack in purity and may be ambiguous. Manipulation composability is extremely good, if implemented by transformers, and very limited symbol composability can be performed by algebraic operations in embedding space. To the best of our knowledge, symbolic properties of both probabilistic discrete latent representations and deep hashing methods have not been studied yet, and their application outside their traditional domains (variational autoencoders and retrieval systems) is relatively unexplored. We conjecture they both can achieve good generalizability and usefulness, provided they are learned with additional regularizations, but lack either kind of composability. Finally, discrete tensors acting as interfaces between neural and symbolic components, achieve the maximum degree of composability (delegated entirely to the symbolic part), along with the potential for high levels of purity and non-ambiguity (handled by the neural one). However, they have limited generalizability and, unless it is possible to backpropagate through the symbolic component, they may have reduced (or absent) usefulness.

In appendix A, we provide examples on how these properties may be quantified in practice.

3.2. The challenges of learning

Every discretization operation, from the simplest $sign(x)$, to the most complicated, is typically implemented by step functions, that are not well suited for gradient-based optimization methods, since their derivative is zero everywhere except in discontinuity points. This characteristic effectively stops **information flow**, because the chain rule would cause vanishing gradients. Existing methods mitigating this issue are not general and present specific issues. Residual connections between continuous and discrete representations are a naive and somewhat ineffective approach: since gradient only flows along the continuous path, the learning process is encouraged to discard information coming from the discrete one. A temperature parameter scaling inputs for a sigmoid or softmax non-linearity can be annealed to zero during training, providing a progressively smaller gradient. Although simple and potentially effective, this method is sensitive to initial values and annealing schedule [20]. Straight-through estimation replaces a non-differentiable non-linear activation with the identity function in the backward pass, hence "skipping" the activation during chain rule computation. This approach works well in those cases in which the non-linearity closely approximates the identity, but may provide misleading gradient updates when this assumption does not hold. Methods such as the ones exploited by differentiable combinatorial optimizers [11] are extremely powerful techniques for backpropagation, based on building gradients as finite differences between the input representation and the minimal perturbation acting as a counterfactual. Although effective and representation-centric, they require binary decision boundaries and are slow to compute.

An additional issue arising in gradient-based methods is the **choice of the loss function**. Symbolic computations are usually discrete in nature, and therefore it may be challenging to devise both a quantitative evaluation of the representation, and an optimizable continuous relaxation or upper bound of such evaluation. Assignment-based manipulations (e.g., deduction, Hungarian matching, etc.) can be reformulated within a fuzzy or probabilistic framework, allowing for a differentiable objective, with suitable thresholding operations performed at inference time, which can however present boundary instability issues. Retrieval-based approaches heavily rely on non-differentiable functions comparing different representations (such as the Hamming distance) and often require to devise a continuous upper bound [21].

High-quality training data is often expensive to annotate and the ideal condition in which supervision is provided for every concept associated to each data point is often far from reality. Symbolic representation learning should exploit intrinsic characteristics which reduce the **quantity and quality of annotations required**, ideally up to the completely unsupervised discovery of symbols and their compositions. In general, one could provide (i) complete supervision on concepts, (ii) supervision on the downstream task only, (iii) no supervision at all, (iv) weak or partial supervision in the form of relations between entities (e.g., monotonicity constraints, ontologies, equivalence classes between entities, etc.).

Knowledge plays an important role both in learning and reasoning. Being able to learn a representation by means of **knowledge injection** is not only a way of reducing the amount of data required to achieve a "good" representation, but also greatly improves usefulness. This challenge is strongly related to the way in which supervisions are provided, as well as to the problem of learning through gradient-descent. Knowledge can be injected in the form of (i) representational constraints (e.g., distances between representations should mirror paths inside a knowledge graph), (ii) constraints related to the downstream task structure (as it happens with planning [3]) , (iii) manipulation constraints (e.g., symbols should be combined according to a generative grammar). Knowledge injection could exploit techniques such as architectural constraints, semantic-based regularization [22], empirical model learning [23], conditioning from a symbolic component [24], unstructured knowledge integration [25], etc.

Regarding the **metrics** to be used to evaluate symbolic representations, we do not believe in the existence of a one-size-fits-all metric, in virtue of different downstream applications for which different subsets of desirable properties are needed. To define a suitable metric, one should consider: (i) the target property, (ii) the type of supervision required, (iii) the interaction with learning. By target property we mean the property quantified by the metric: we provide some examples in Appendix A. An important open question along this dimension is **how to quantify composability**. The second dimension addresses the availability of ground truth with respect to the four categories of quality of annotations mentioned above. For example, in a metric learning setup only the equivalence relation between inputs is known (category iv), and purity and non-ambiguity could be measured by re-identification metrics such as mAP@k, while in planning from visual stimuli where only feedback on the downstream task is available (category ii), usefulness of representations could be measured by the number of failed plans, or the average length of successful plans. Differentiable metrics may be also exploited during learning. Along this dimension we can distinguish (i) truly differentiable metrics, (ii) non-differentiable metrics, (iii) metrics with differentiable proxies (e.g., Hamming distance approximated by binary crossentropy), (iv) compliance to logic or combinatorial constraints. In the latter case it may be possible to define, for example, complex purity measures taking into account hierarchies of entities, or logic formulae against which compliance can be evaluated with techniques such as semantic-based regularization [22].

Additionally, specific challenges arise in specific scenarios. For example, in a **multi-agent setting**, agents may share information in the form of visual stimuli or directly as symbols. In the first case, an additional challenge is to ensure coherence across different representations; in the second case, the learning process should pursue consistency, so that the same symbol has the same meaning for all agents, and should also handle the case in which an agent may receive new symbols it never encountered before. In the case of **continual learning**, additional challenges are posed by handling catastrophic forgetting (not only in the sense of poor performance in forgotten tasks, but also in the sense of forgetting or overwriting the "meaning" of rarely observed representations), guaranteeing elaboration tolerance when new information is discovered, and generally requiring a stronger robustness to noise. As these require case-specific considerations, we leave their discussion to future work.

4. A Preliminary Experiment

We now present a preliminary experiment, devised to back up some of our claims, and to pave the way for further investigations. Our aim is to analyze (i) whether symbolic representations can be achieved both by means of supervised classification and metric learning, and (ii) whether learning a symbolic representation via a downstream subsymbolic task is beneficial.

4.1. Experimental Setup

We considered the traditional MNIST digit classification [26] and the associated one-digit sum tasks [27]. We employ a siamese neural network [28] to learn discrete representations that are fed into two independent neural classifiers, one for digit classification (output layer with 10 neurons) and the other for the sum (classification task with 19 possible labels). Both classifiers are implemented via a multi-layer perceptron with two hidden layers (16 and 32 neurons, respectively), using batch normalization, GELU activation function, and dropout rate $p = 0.1$. The two classifiers share no parameters. We measure performance in terms of F1-score.

Our training data consist of triplets randomly sampled from the MNIST dataset (we extract batches of 64 samples for 20,000 training steps): two images belong to the same class, and the third to a different one. In the literature, these samples are usually called anchor (a), positive (p) and negative (n). We store as training labels the value of the anchor for the single digit classification, and the sum of anchor and negative for the sum classification.[1]

As an additional experiment, we consider whether the discrete representations, learned when the downstream task is the one-digit sum, can be exploited by a decision tree to perform the same classification tasks. For this scenario, we collect 200 training batches of 64 samples each (thus 12,800 training samples), we compute the discrete representations, and implement a standard decision tree using the default parameters of the scikit-learn library [29].

After training, we measure non-ambiguity and purity of learned representations, exploiting the entropy-based method presented in appendix B for supervised evaluation. Being intrinsic properties of the representation, we measure them once, regardless of the downstream task.

We leave to future work the design and implementation of metrics to validate the other properties presented in Section 3.

[1]Single digit classification could be improved by also annotating positive and negative samples: however, to balance the number of supervisions available for both tasks, we decided not to.

4.2. Implementation details

The discrete representation bottleneck is a layer with N neurons which is discretized by means of the sign function, and which provides intermediate values for loss computation as follows:

$$\ell = W \cdot x + b \qquad h = \tanh(\ell)$$
$$d = \text{sign}(h) \qquad h' = \frac{h}{2} + \frac{1}{2}$$

where ℓ is a linear combination of the inputs, h applies to ℓ the non-linear activation function $\tanh$, d is the discretized (binary) encoding of h, and h' rescales the values of h in [0,1], in order to further exploit training with a cross-entropy loss. Embedding ℓ is subject to L2 regularization and to the traditional Euclidean triplet loss with anchor a, positive p and negative n:

$$\mathcal{L}_{triplet} = \max\left(\|\ell_a - \ell_p\|_2^2 - \|\ell_a - \ell_n\|_2^2 + \alpha, 0\right)$$

whereas a second triplet loss, modified to approximate Hamming distance, is applied to h':

$$\mathcal{L}_{Hamming} = \max\left(BCE(h'_a, h'_p) - BCE(h'_a, h'_n) + \beta, 0\right).$$

BCE is the standard binary cross-entropy loss, whereas α and β are hyper-parameters, fixed to 0.5 in every experiment. The final learning objective is given by:

$$\mathcal{L} = \lambda_m \cdot \mathcal{L}_{triplet} + \lambda_h \cdot \mathcal{L}_{Hamming} + \lambda_c \cdot (\mathcal{L}_{digits} + \mathcal{L}_{sum}) + \lambda_r \|\ell\|_2^2$$

being λ_m and λ_h the weights of the representation losses, λ_c that of the classification losses $\mathcal{L}_{digits}$ and $\mathcal{L}_{sum}$ (standard cross-entropy), and λ_r that of L2-regularization applied to ℓ.

As the sign function has an ill-behaved gradient, the gradient flow across the discrete representation layer is effectively stopped.[2] To demonstrate the importance of receiving information from a downstream task, we implemented a simple strategy via the stop gradient operator: basically, we propagate d in the forward pass, and the gradient of h during the backward pass.

Purity and non-ambiguity are computed with a representation matrix associating discrete representations (d) and true labels.

4.3. Results

Figure 1 shows the F1 scores for the tested classifiers as a function of the number of bits used for the discrete representations. The top row reports performance of the neural classifiers for the digit (a) and sum (b) classification tasks, respectively. The bottom row in (c) and (d) does the same for the decision trees trained on the learned discrete representations. Lines are grouped by those where the classifiers backpropagate information (grad, $\lambda_c = 1$, solid lines), those where they do not (nograd, $\lambda_c = 1$, dashed lines) and those where no classifier is trained and the discrete representation depends entirely on the triplet losses (nocls, $\lambda_c = 0$, dotted lines). Nonsensical combinations (e.g., nocls, $\lambda_m = \lambda_h = 0$) have not been used. We kept $\lambda_r = 1$.

Results show that raw performance is good for every model, both for the neural classifier and the decision tree, with the exception of the cases in which the Hamming loss is the only training

[2]When gradient is not back-propagated, then the learned representations are inherently arbitrary.

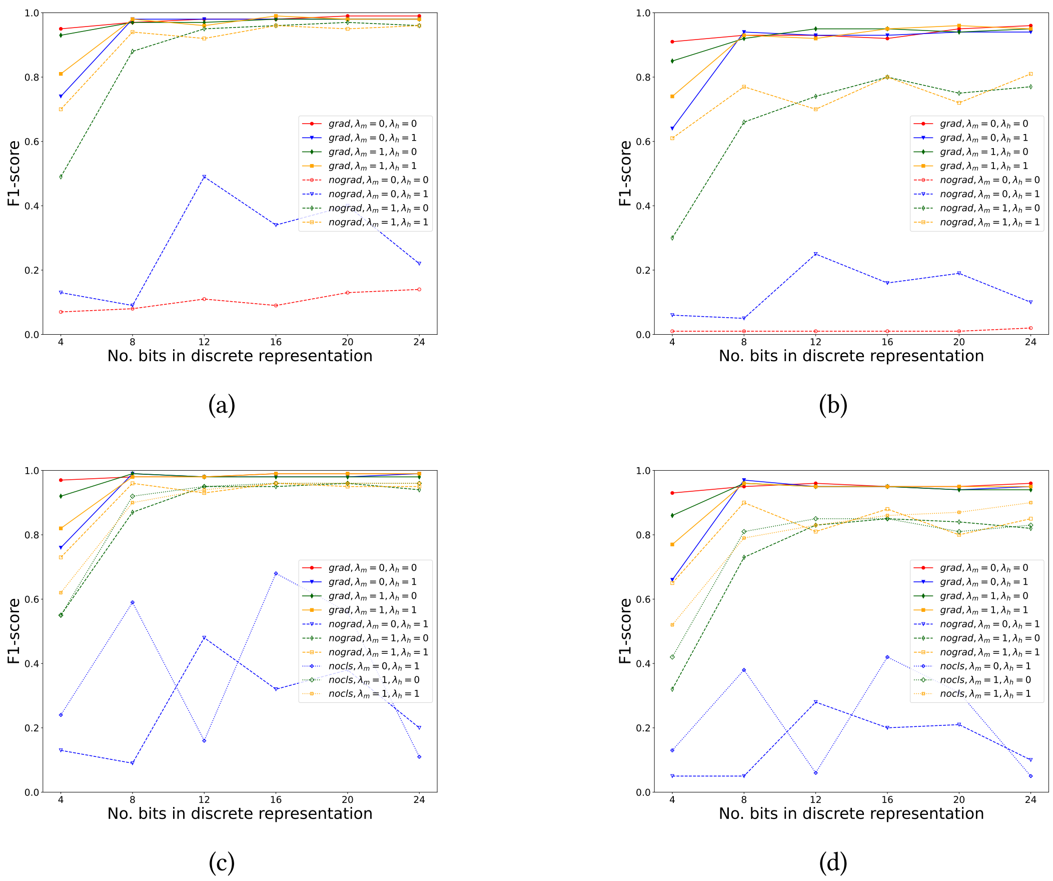
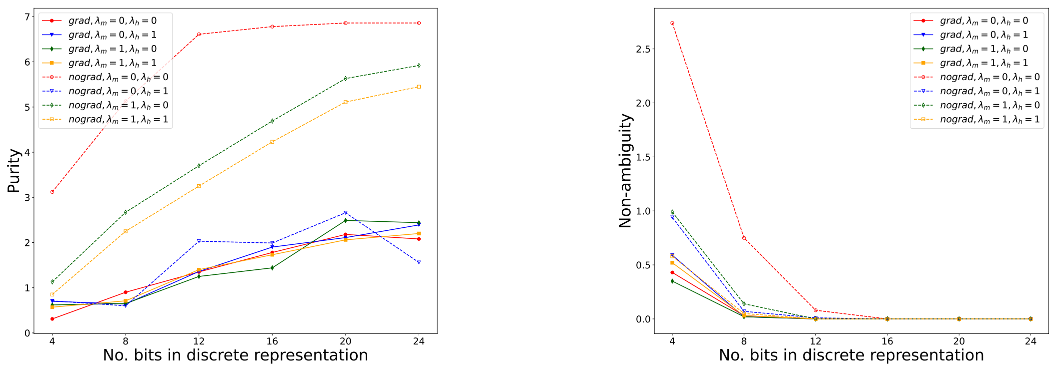


Figure 1: F1 score (y-axis) on digit and sum classification as a function of the number of bits used for the discrete representation (x-axis): (a) digit classification with MLP; (b) sum classification with MLP; (c) digit classification with decision trees; (d) sum classification with decision trees.

signal (nograd, $\lambda_m = 0$, $\lambda_h = 1$). The Hamming triplet loss alone is not able to converge to useful discrete representations (blue dashed and dotted curves in Figures 1c and 1d): this is a counter intuitive result, however we posit that the multiple steps involved in the computation (bits approximated by tanh, Hamming distance approximated by binary crossentropy on tanh rescaled in [0,1]) may be a poor proxy of the Hamming distance. Unsurprisingly, the classification objective alone (grad, $\lambda_m = \lambda_h = 0$) provides a stronger representation than the metric objective alone (nocls, $\lambda_m = 1$, $\lambda_h = 0$). Table 4 in Appendix C reports the same results in tabular form.

Figure 2 shows purity (a) and non-ambiguity (b) as a function of the number of bits in the discrete representation (lower values are better). Table 6 in Appendix C represents the same data in numerical form. Not surprisingly, we observe that purity degrades (larger values) with a larger number of bits, as without any form of regularization the learning process tends to use more symbols than necessary. When gradients from the classifier are allowed to flow (grad), purity degradation is more graceful and it is relatively insensitive to other parameters. When gradient is stopped (nograd), purity degrades more sharply. Interestingly, the Hamming



(a) (b)

Figure 2: Purity (a) and non-ambiguity (b) of learned symbols (lower values are better).

objective alone (dashed blue line) seems to have a regularizing effect on purity, whereby the other cases quickly diverge (the red dashed line represents a random baseline, as no learning objective was enforced on the discrete representation). As for non-ambiguity, it improves in all cases as the number of bits increases. While ultimately converging to zero, each curve has a different decreasing rate: as expected, when gradient flows, convergence is faster (grad, solid lines) and the random baseline (dashed red line) is the slowest.

5. Conclusions

We considered the task of learning symbolic representation, a problem that has been under-explored in the neuro-symbolic AI community. We defined the key properties of discrete symbolic representations and identified challenges associated to the learning problem. As the downstream use of symbolic representations can be applied to an ample and varied set of symbolic tasks, we identified a strong need also for appropriate evaluation metrics.

Multiple interesting research directions can be identified for the future. In particular, we aim to focus on two promising topics: the use of Peano arithmetics to represent the properties of symbolic representations by modeling composability through recursion, and the formalization of the learning problem within a continual learning setting, which also seems a natural way to enable the composability of different representations.

References

[1] G. Ciravegna, P. Barbiero, F. Giannini, M. Gori, P. Lió, M. Maggini, S. Melacci, Logic explained networks, Artificial Intelligence 314 (2023) 103822.

[2] A. Newell, H. A. Simon, et al., Human problem solving, volume 104, Prentice-hall Englewood Cliffs, NJ, 1972.

[3] M. Asai, A. Fukunaga, Classical planning in deep latent space: Bridging the subsymbolic-

symbolic boundary, in: Proceedings of the aaai conference on artificial intelligence, volume 32, 2018.

[4] A. Van Den Oord, O. Vinyals, et al., Neural discrete representation learning, Advances in neural information processing systems 30 (2017).

[5] X. Luo, H. Wang, D. Wu, C. Chen, M. Deng, J. Huang, X.-S. Hua, A survey on deep hashing methods, ACM Transactions on Knowledge Discovery from Data (TKDD) (2020).

[6] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, M. Rolinek, Differentiation of blackbox combinatorial solvers, in: International Conference on Learning Representations, 2020.

[7] W. Stammer, M. Memmel, P. Schramowski, K. Kersting, Interactive disentanglement: Learning concepts by interacting with their prototype representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10317–10328.

[8] E. Jang, S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, arXiv preprint arXiv:1611.01144 (2016).

[9] C. J. Maddison, A. Mnih, Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, arXiv preprint arXiv:1611.00712 (2016).

[10] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, J. Z. Kolter, Differentiable convex optimization layers, Advances in neural information processing systems 32 (2019).

[11] M. Fredrikson, K. Lu, S. Vijayakumar, S. Jha, V. Ganesh, Z. Wang, Learning modulo theories, arXiv preprint arXiv:2301.11435 (2023).

[12] P.-W. Wang, P. Donti, B. Wilder, Z. Kolter, Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver, in: International Conference on Machine Learning, PMLR, 2019, pp. 6545–6554.

[13] M. Taddeo, L. Floridi, Solving the symbol grounding problem: a critical review of fifteen years of research, Journal of Experimental & Theoretical Artificial Intelligence 17 (2005) 419–445.

[14] A. Santoro, A. Lampinen, K. Mathewson, T. Lillicrap, D. Raposo, Symbolic behaviour in artificial intelligence, arXiv preprint arXiv:2102.03406 (2021).

[15] B. Hjørland, Nine principles of knowledge organization (1994).

[16] R. Davis, H. Shrobe, P. Szolovits, What is a knowledge representation?, AI magazine 14 (1993) 17–17.

[17] J. McCarthy, From here to human-level ai, Artificial Intelligence 171 (2007) 1174–1182.

[18] K. Greff, S. Van Steenkiste, J. Schmidhuber, On the binding problem in artificial neural networks, arXiv preprint arXiv:2012.05208 (2020).

[19] J. McCarthy, Elaboration tolerance, in: Common sense, volume 98, 1998, p. 2.

[20] Ł. Kaiser, S. Bengio, Discrete autoencoders for sequence models, arXiv preprint arXiv:1801.09797 (2018).

[21] M. Norouzi, D. J. Fleet, R. R. Salakhutdinov, Hamming distance metric learning, Advances in neural information processing systems 25 (2012).

[22] M. Diligenti, M. Gori, C. Sacca, Semantic-based regularization for learning and inference, Artificial Intelligence 244 (2017) 143–165.

[23] M. Lombardi, M. Milano, A. Bartolini, Empirical decision model learning, Artificial Intelligence 244 (2017) 343–367.

[24] E. Misino, G. Marra, E. Sansone, Vael: Bridging variational autoencoders and probabilistic

logic programming, arXiv preprint arXiv:2202.04178 (2022).
[25] F. Ruggeri, Towards unstructured knowledge integration in natural language processing (2022).
[26] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.
[27] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, Deepproblog: Neural probabilistic logic programming, advances in neural information processing systems 31 (2018).
[28] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a" siamese" time delay neural network, Advances in neural information processing systems 6 (1993).
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[30] A. New, M. Baker, E. Nguyen, G. Vallabha, Lifelong learning metrics, arXiv preprint arXiv:2201.08278 (2022).

A. How Can Properties Be Measured?

The definitions we provided in Section 3.1 are general enough to be applicable to a broad variety of tasks. We hereby instantiate those definitions to some exemplar metrics, applicable in specific contexts. We will mainly focus on the supervised setting in which labels are available for each of the entities to be encoded, and we will try to generalize where possible. It is important to note how more interesting cases, such as the unsupervised setting, present challenges which make quantification difficult. In Section 3.2 we discussed a possible taxonomy of metrics, to better navigate such complex landscape.

In the supervised case, discrete representations of discrete entities allow to build a **representation matrix**, where rows are associated with entities and columns with representations. Each entry $\mathcal{R}_{ij}$ counts the times in which representation j is used to encode entity i. In general this matrix is rectangular, and it can be seen as a generalization of the confusion matrix, where predicted labels are replaced by their representations.

Purity and **non-ambiguity** can be defined in terms of the representation matrix, and the ideal condition would correspond to a matrix where every row and every column have exactly one non-zero entry (such as Table 3). **Purity** can be naively computed on the representation matrix as a pseudo-recall, but a more robust metric would be the column-wise average of the entropies computed for each row (after a row-wise normalization of counts). Likewise, **non-ambiguity** can be computed as a pseudo-precision, or as the row-wise average of the entropies for each column (normalized column-wise). For both metrics, zero is the optimal value, and larger positive values are progressively worse. Appendix B contains an example on how to compute purity and non-ambiguity from a representation matrix.

Usefulness is strongly coupled with the nature of the downstream task, thus there are no general metrics. Moreover, it follows from the definition that a suitable metric should compare two different representations. Since downstream tasks can be monitored by means of performance metrics (even in the case of weakly or unsupervised symbols), we propose to quantify usefulness as a direct comparison of the two performance scores. This comparison can be instantiated as a difference, ratio, log-ratio, or any other suitable function.

Symbol and **manipulation composability** are challenging to quantify, however it is still important to define an order relation between methods. To give a motivating example for this claim, consider the case of two systems, one of which combines symbols by concatenating their representations, and the other constructing a causal graph where nodes are grounded with the representations: these two approaches clearly possess a different degree of symbol composability. Likewise, for manipulation composability, the repeated application of the same multi-layer perceptron block a fixed number of times, transformer-like blocks, and the unbounded iteration of a recurrent architecture where gates are conditioned by external inputs, have different expressivity. We do not have a practical solution, however we advocate for the need of a (preferably formal) hierarchy of methods, which reflects the intuitive comparisons made above. We postulate that a Chomsky-like hierarchy, where classes are defined by constraints on the allowed manipulations, may serve the purpose.

Generalizability can be evaluated for multi-task settings or when it is important to assess robustness with respect to domain shifts. We suggest three possible approaches for quantifying **generalizability**: a naive direct comparison (similar to usefulness), the use of multi-task benchmarks (possibly with an aggregated final score), or the use of continual learning experimental setups [30]. Since we are measuring the generalizability of a representation, instead of the entire architecture, the encoding process should be kept fixed across every task.

B. Measuring Purity and Non-Ambiguity

In this section we show how to compute purity and non-ambiguity using the entropy-based metrics proposed in Appendix A. For this example we perform computations with base 2 logarithms and assume a numpy-like indexing (rows-first, indexes go from 0 to $size - 1$).

Let us assume a task where supervision is given for five symbols (e.g., classes), A, B, C, D, E associated with six possible representations, j, k, l, m, n, o (note that we abstract from the specific encoding used, i.e., j could be associated with a string, a sequence of bits or an ordinal number, the only consideration here is that j is different than k, and so on). After training, each symbol is mapped to the representations with a number of occurrences indicated by Table 1. Note how there is symbol imbalance (A is represented 11 times, while C only twice).

Also note how for symbols D and E, there is only one possible representation (if considered alone their representation would be perfectly pure), and how representation j is associated only to symbol B (so it is a perfectly non-ambiguous representation). The purity and non-ambiguity metrics proposed try to quantify such behavior **globally**.

Let us now compute purity. We first normalize the counts by row, which produces the matrix P on the left in Table 2. We then compute the entropy of each row as follows:

$$H_i = \sum_{j=0}^{5-1} -P_{ij} \cdot \log_2(P_{ij}).$$

Table 1
Example of representation matrix. Rows indicate symbols, columns indicate representations.

	j	k	l	m	n	o	Total
A	0	1	0	3	5	2	11
B	1	0	1	0	0	1	3
C	0	0	0	1	0	1	2
D	0	7	0	0	0	0	7
E	0	0	9	0	0	0	9
Total	1	8	10	4	5	4	

Table 2
Example of computation of purity and non-ambiguity. On the left table values are normalized by rows and entropy is computed for each (last column), then they are averaged for purity. On the right, values are normalized by columns and entropies (last row) are averaged to produce non-ambiguity.

0	0.09	0	0.27	0.45	0.18	1.79
0.33	0	0.33	0	0	0.33	1.58
0	0	0	0.5	0	0.5	1
0	1	0	0	0	0	0
0	0	1	0	0	0	0
						0.87

0	0.125	0	0.75	1	0.5	
1	0	0.1	0	0	0.25	
0	0	0	0.25	0	0.25	
0	0.875	0	0	0	0	
0	0	0.9	0	0	0	
0	0.54	0.47	0.81	0	1.5	**0.40**

Finally, we average the entropies, by summing and dividing by 5.

For non-ambiguity, the process is the same, but rows and columns are switched (we call Q the matrix normalized by columns). Entropies are computed column-wise as follows:

$$H_j = \sum_{i=0}^{5-1} -Q_{ij} \cdot \log_2(Q_{ij})$$

In this case, averaging is performed row-wise, so we divide by 6.

Table 3 is a different example, where an encoding perfectly matches symbols to representations (this can be the case of a one-hot encoding output by a perfect classifier, for example). Note how, in this case, normalizing row-wise and column-wise will both produce a permutation of the identity matrix, and thus how entropy is zero, no matter from which direction it is computed.

An important point to keep in mind when using a representation matrix is sparsity, for two reasons. The first one is storage or computation efficiency, as large representation sets grow exponentially with the degrees of freedom of representations (e.g., 32-bit binary representations for 10 classes have a 10×2^{32} representation matrix, which cannot be stored densely). Since the validation and test sets of a machine learning setup are typically assumed to be finite in size, it may be more efficient to store occurrences as associations (e.g., dictionaries mapping entities and representations, which will only grow linearly with the dataset and representation sizes), but this may result in slower computations for denser representations. The other consideration is related to the normalization term in the final average of non-ambiguity computations, as it can be computed as the total number of representations, or as the number of representations with non-zero occurrence. The former choice can lead to an underestimation of non-ambiguity in the case of a sparse representation space (where many entries in the representation matrix

Table 3
Example of a representation matrix for a perfectly pure and non-ambiguous encoding (eg. one-hot of true labels) of entities A–E to symbols j–n. Note how this is a square matrix corresponding to a permuted confusion matrix with perfect accuracy.

	j	k	l	m	n	Total
A	0	10	0	0	0	10
B	0	0	3	0	0	3
C	0	0	0	0	2	2
D	7	0	0	0	0	7
E	0	0	0	9	0	9
Total	7	10	3	9	2	

are zero).[3] It can be argued, however, that many symbolic representations, especially those rooted in logic, work better with concise (and thus dense) representations, and that the latter choice, of averaging with respect to non-zero entries, would break the symmetry with respect to purity computation.

C. Additional Details on Experiments

In Tables 4 and 5 we report the performance, in terms of F1-score, achieved by neural classifiers and, respectively, decision trees, on the digit (D) and sum (S) classification tasks, with the considered combinations of regularization hyper-parameters in the loss function. These are the same data represented in graphical form in Figure 1. Similarly, Table 6 reports the values of purity and non-ambiguity as a function of the number of bits used for the representations.

[3]Note that the same does not hold for purity, since it is averaged across entities.

Table 4
F1 score of the neural classifiers for digits (D) and sums (S).

Latent dim		4	8	12	16	20	24
grad							
$\lambda_m = 0$	D	0.95	0.97	0.98	0.98	0.99	0.99
$\lambda_h = 0$	S	0.91	0.93	0.93	0.92	0.95	0.96
$\lambda_m = 0$	D	0.74	0.98	0.98	0.98	0.98	0.98
$\lambda_h = 1$	S	0.64	0.94	0.93	0.93	0.94	0.94
$\lambda_m = 1$	D	0.93	0.97	0.97	0.98	0.98	0.98
$\lambda_h = 0$	S	0.85	0.92	0.95	0.95	0.94	0.95
$\lambda_m = 1$	D	0.81	0.98	0.96	0.99	0.98	0.98
$\lambda_h = 1$	S	0.74	0.93	0.92	0.95	0.96	0.95
nograd							
$\lambda_m = 0$	D	0.07	0.08	0.11	0.09	0.13	0.14
$\lambda_h = 0$	S	0.01	0.01	0.01	0.01	0.01	0.02
$\lambda_m = 0$	D	0.13	0.09	0.49	0.34	0.40	0.22
$\lambda_h = 1$	S	0.06	0.05	0.25	0.16	0.19	0.10
$\lambda_m = 1$	D	0.49	0.88	0.95	0.96	0.97	0.96
$\lambda_h = 0$	S	0.30	0.66	0.74	0.80	0.75	0.77
$\lambda_m = 1$	D	0.70	0.94	0.92	0.96	0.95	0.96
$\lambda_h = 1$	S	0.61	0.77	0.70	0.80	0.72	0.81

Table 5
F1 score of the decision tree classifiers for digits (D) and sums (S).

Latent dim		4	8	12	16	20	24
grad							
$\lambda_m = 0$	D	0.97	0.98	0.98	0.99	0.99	0.99
$\lambda_h = 0$	S	0.93	0.95	0.96	0.95	0.95	0.96
$\lambda_m = 0$	D	0.76	0.99	0.98	0.98	0.98	0.99
$\lambda_h = 1$	S	0.66	0.97	0.95	0.95	0.94	0.95
$\lambda_m = 1$	D	0.92	0.99	0.98	0.98	0.98	0.98
$\lambda_h = 0$	S	0.86	0.96	0.95	0.95	0.94	0.94
$\lambda_m = 1$	D	0.82	0.98	0.98	0.99	0.99	0.99
$\lambda_h = 1$	S	0.77	0.96	0.95	0.95	0.95	0.95
nograd							
$\lambda_m = 0$	D	0.09	0.10	0.10	0.12	0.11	0.10
$\lambda_h = 0$	S	0.04	0.05	0.05	0.06	0.07	0.06
$\lambda_m = 0$	D	0.13	0.09	0.48	0.32	0.38	0.20
$\lambda_h = 1$	S	0.05	0.05	0.28	0.20	0.21	0.10
$\lambda_m = 1$	D	0.55	0.87	0.95	0.95	0.96	0.94
$\lambda_h = 0$	S	0.32	0.73	0.83	0.85	0.84	0.82
$\lambda_m = 1$	D	0.73	0.96	0.93	0.96	0.95	0.95
$\lambda_h = 1$	S	0.65	0.90	0.81	0.88	0.80	0.85
nocls							
$\lambda_m = 0$	D	0.05	0.09	0.11	0.12	0.12	0.12
$\lambda_h = 0$	S	0.05	0.06	0.06	0.06	0.06	0.06
$\lambda_m = 0$	D	0.24	0.59	0.16	0.68	0.56	0.11
$\lambda_h = 1$	S	0.13	0.38	0.06	0.42	0.31	0.05
$\lambda_m = 1$	D	0.55	0.92	0.95	0.96	0.96	0.96
$\lambda_h = 0$	S	0.42	0.81	0.85	0.85	0.81	0.83
$\lambda_m = 1$	D	0.62	0.90	0.94	0.96	0.96	0.96
$\lambda_h = 1$	S	0.52	0.79	0.83	0.86	0.87	0.90

Table 6
Purity (P) and non-ambiguity (N) of discrete representations for the neural classifiers. Lower values are better. The row nograd $\lambda_m = 0, \lambda_h = 0$ corresponds to random mapping and can thus be considered a baseline.

Latent dim		4	8	12	16	20	24
grad							
$\lambda_m = 0$	P	0.31	0.90	1.35	1.78	2.18	2.08
$\lambda_h = 0$	N	0.43	0.03	0.00	0.00	0.00	0.00
$\lambda_m = 0$	P	0.70	0.64	1.36	1.90	2.11	2.39
$\lambda_h = 1$	N	0.59	0.02	0.00	0.00	0.00	0.00
$\lambda_m = 1$	P	0.62	0.65	1.25	1.44	2.49	2.44
$\lambda_h = 0$	N	0.35	0.02	0.00	0.00	0.00	0.00
$\lambda_m = 1$	P	0.57	0.71	1.40	1.73	2.06	2.20
$\lambda_h = 1$	N	0.52	0.03	0.00	0.00	0.00	0.00
nograd							
$\lambda_m = 0$	P	3.12	5.13	6.61	6.78	6.86	6.86
$\lambda_h = 0$	N	2.74	0.75	0.08	0.00	0.00	0.00
$\lambda_m = 0$	P	0.71	0.60	2.03	1.99	2.66	1.56
$\lambda_h = 1$	N	0.94	0.07	0.01	0.00	0.00	0.00
$\lambda_m = 1$	P	1.13	2.67	3.70	4.69	5.63	5.92
$\lambda_h = 0$	N	0.99	0.14	0.00	0.00	0.00	0.00
$\lambda_m = 1$	P	0.85	2.25	3.25	4.23	5.11	5.45
$\lambda_h = 1$	N	0.58	0.05	0.00	0.00	0.00	0.00