Soft Tensor Product Representations for Fully Continuous, Compositional Visual Representations

Bethia Sun
UNSW, Sydney
bethia.sun@unsw.edu.au

Maurice Pagnucco
UNSW, Sydney
morri@unsw.edu.au

Yang Song
UNSW, Sydney
yang.song1@unsw.edu.au

Abstract

Since the inception of the classicalist vs. connectionist debate, it has been argued that the ability to systematically combine symbol-like entities into compositional representations is crucial for human intelligence. In connectionist systems, the field of disentanglement has emerged to address this need by producing representations with explicitly separated factors of variation (FoV). By treating the overall representation as a *string-like concatenation* of the inferred FoVs, however, disentanglement provides a fundamentally *symbolic* treatment of compositional structure, one inherently at odds with the underlying *continuity* of deep learning vector spaces. We hypothesise that this symbolic-continuous mismatch produces broadly suboptimal performance in deep learning models that learn or use such representations. To fully align compositional representations with continuous vector spaces, we extend Smolensky's Tensor Product Representation (TPR) and propose a new type of inherently *continuous* compositional representation, *Soft TPR*, along with a theoretically-principled architecture, *Soft TPR Autoencoder*, designed specifically for learning Soft TPRs. In the visual representation learning domain, our Soft TPR confers broad benefits over symbolic compositional representations: state-of-the-art disentanglement and improved representation learner convergence, along with enhanced sample efficiency and superior low-sample regime performance for downstream models, empirically affirming the value of our inherently *continuous* compositional representation learning framework.

1 Introduction

Compositional structure, capturing the property of being decomposable into a set of constituent parts, is ubiquitous in our surroundings – from the recursive application of syntax in language, to the parsing of richly complex visual scenes into their constituent parts. Given the central role such structure plays in our understanding of the world, it is highly intuitive that deep learning representations also embody compositional structure. Indeed, empirical evidence highlights the usefulness of explicitly compositional representations, showcasing a multitude of benefits, including increased interpretability [10, 12], reduced sample complexity [30, 33], increased fairness [20, 25, 41], and improved performance in out-of-distribution generalisation [33, 48, 50].

We consider the following, intuitive notion of compositional representations. A representation of compositionally-structured data is a *compositional representation* if it has a structure that faithfully reflects the compositional structure of the represented data [49]. In the visual representation learning domain, data is clearly *compositionally-structured*, as images can be decomposed into a set of constituent *factors of variation* (FoVs), e.g., $\{\text{magenta floor}, \text{orange wall}, \text{aqua object colour}, \text{oblong object shape}\}$ for the image in Figure 1.

Code is available at `https://github.com/gomb0c/soft_tpr/`

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

A widely explored representation learning framework producing explicitly compositional representations is that of *disentanglement*. We adopt the conventional [5, 25, 33, 43], intuitive definition of a *disentangled representation*, which states that a representation, $\psi(x)$, is disentangled if each of the underlying FoVs can be cleanly separated into a distinct dimension (or contiguous subset of dimensions) of $\psi(x)$, or, in other words, if each FoV has a 1-1 correspondence with a distinct *part* of the representation [43]. Framed in this way, it is apparent that disentangled representations are explicitly compositional by nature. The majority of state-of-the-art disentanglement approaches use a variational autoencoder backbone, and rely on weak supervision [13, 22, 31, 33, 35], or a penalisation of the aggregate posterior $\int q(z|x)p(x)dx$ [10, 14, 17, 24, 26, 32] to promote disentanglement. More recent approaches depart from the restrictive assumptions of a variational framework, and instead use standard autoencoding [47], or energy-function based optimisation [36], with additional inductive biases to encourage disentanglement. Despite the diversity of methods characterising existing work, we make a crucial observation which unifies them together: by enforcing the 1-1 correspondence between FoVs and distinct parts of the representation, existing approaches [10, 13, 14, 17, 22, 24, 26, 31, 32, 33, 35, 36, 47] essentially produce compositional representations corresponding to a *concatenation* of scalar-valued or vector-valued FoV tokens, as illustrated in Figure 1a. This concatenative approach enforces a rigid, *slot-based* representational structure that constraints how information can be represented and combined, and mirrors symbolic systems, where *distinct* symbols occupy *discrete* slots within a representation. We argue that this fundamentally *symbolic* approach creates a deep incompatibility with the inherent *continuity* of the vector spaces underlying deep learning for the following reasons:

1. **Misalignment with Gradient-Based Learning**: The symbolic approach's introduction of discrete representational slots for each FoV introduces non-differentiable boundaries between FoV slots, challenging the efficacy of gradient-based learning, which thrives on smooth and continuous transformations. For example, when modifying a given FoV, this symbolic structure restricts gradient propagation to the dimensions associated solely to the corresponding representational slot, inhibiting the smooth permeation of gradients across the entire vector space. Managing these discrete, slot-based boundaries thus fragments gradient flow across the vector space, producing abrupt, discontinuous transitions that may potentially complicate learning.
2. **Restrictive / Incompatible Structure**: By allocating distinct representational slots for each FoV, the symbolic approach imposes a rigid representational structure that prevents the representation from exploiting the expressivity inherent in continuous vector spaces. More concretely, this slot-based framework prevents the encoding of FoVs as flexible combinations of basis vectors that span the *entire* representational space – an approach that is not only more intuitive, but also critical for capturing rich interactions and complex dependencies among FoVs. By failing to allow for this flexibility, the symbolic approach prevents the representation from leveraging the full expressivity of its underlying vector space.

Critically, we hypothesise that the fundamental incompatibility between the *symbolic* treatment of compositional structure provided by disentanglement, and the *continuous* vector spaces of deep learning produces suboptimal behaviour in the models that learn or use these representations. This hypothesis prompts the following question: can we instead represent compositional structure in an inherently *continuous* manner? A *continuous* compositional representation would yield the representation, $\psi(x)$, by *continuously combining* the FoVs, rather than maintaining a discrete, slot-based separation, as in the symbolic approach. The continuous approach to representing compositional structure is thus, a more mathematically intuitive framework in the context of deep learning.

Pioneered by Smolensky, the Tensor Product Representation [3] is a specific representational form that encodes compositional structure in an inherently *continuous* manner. At the crux of it, TPRs are formed by *continuously blending* the FoVs together into the overall representation, in a manner analogous to superimposing multiple waves together to produce a complex waveform, as illustrated in Figure 1b. For a representation to qualify as a TPR, it must adhere to a highly specific mathematical form, which confers upon the TPR valuable theoretical properties (elaborated on in Section 3.2), but also imposes two major limitations (see B.1 for further details). First, as depicted by the stars in Figure 1c, only a discrete subset of points in the underlying representational space, $\tilde{V}$, satisfies the stringent mathematical criteria to qualify as TPRs. Consequently, to learn TPRs, representation learners must map from the data manifold onto this *discrete subset*, which constitutes a highly constrained and inherently challenging learning task. Second, the TPR specification enforces a strict, algebraic

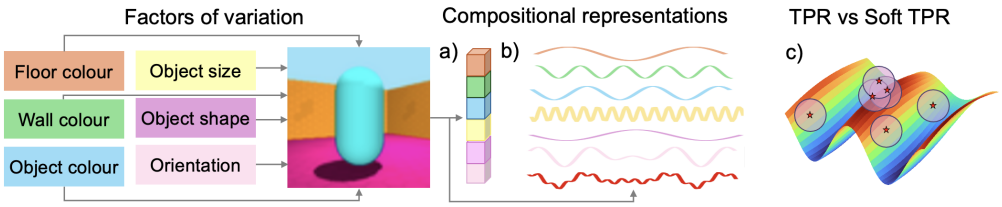

Figure 1: (a) Disentangled representations can be conceptualised as a *concatenation* of FoV tokens (coloured blocks), effectively enforcing a *string-like, symbolic* compositional structure, where each FoV is allocated to a discrete slot in the representation. We instead, consider a *continuous* representation of compositional structure, (b), where the FoVs (first 6 waves) are *continuously superimposed* together to produce the overall representation, $\psi(x)$ (in red). (c) Only a subset of points (stars) in the underlying representational space (rainbow manifold) satisfy the TPR specification. The Soft TPR relaxes this, capturing larger, *continuous regions* of the underlying representational space (the translucent circles), while preserving the TPR's key properties.

definition of compositional structure, limiting the TPR's ability to faithfully represent real-world data which is often *quasi*-compositional, only *approximately* adhering to a rigid, formal definition of compositionality. Historically, these limitations have confined TPR learning to formal domains characterised by explicit, algebraic structure – as evidenced by the near exclusive deployment of TPRs in language [19, 23, 28, 34, 52] – and, to contexts where strong supervision from highly structured downstream tasks is available to steer the representation learning process [23, 28, 38, 51]. To negate these drawbacks and extend continuous compositional representations to weakly supervised, non-formal domains, we propose *Soft TPR*, a new, inherently *continuous* compositional representation that can be thought of as a *continuous relaxation* of the traditional TPR, as illlustrated by the translucent circular regions in Figure 1c. At its core, the Soft TPR is designed to promote representational flexibility and ease of learning while simultaneously preserving the structural and mathematical integrity of the traditional TPR. We additionally introduce *Soft TPR Autoencoder*, a theoretically-principled *weakly-supervised* architecture for learning Soft TPRs, which we use to operationalise the Soft TPR framework in the visual representation learning domain.

Our main contributions are threefold: i) We propose a novel compositional representation learning framework, introducing the inherently continuous *Soft TPR* compositional form, alongside a dedicated, weakly-supervised architecture, *Soft TPR Autoencoder*, for learning this form. ii) Our framework is the first to learn *continuous compositional representations* in the non-formal domain of *vision*. iii) We empirically affirm the far-reaching benefits of enhanced vector space alignment produced by the Soft TPR framework, demonstrating that Soft TPRs achieve state-of-the-art disentanglement, accelerate representation learner convergence, and provide downstream models with enhanced sample efficiency and superior low-sample regime performance.

2 Related Work

Disentanglement: In aiming to produce explicitly compositional representations without strong supervision, our work shares the same objective as disentangled representation learning. Prior to the highly influential work of [25], which proved the impossibility of learning disentangled representations without supervision or other inductive biases, disentangled representations were learnt in a completely unsupervised fashion [8, 10, 14, 17, 24, 26, 37]. Our use of weak supervision is inspired by the work [13, 22, 31, 33, 35] relating to this highly influential impossibility result. In particular, we leverage the type of weak supervision termed 'match pairing' [35], where pairs, (x, x'), differing in values for a subset of known FoVs are presented to the model, to incentivise disentanglement. Our work, however, fundamentally diverges from all disentanglement work we are aware of, by adopting an inherently *continuous* representation of compositional structure, which contrasts with the *symbolic* representations of compositional structure characterising existing work.

TPR-based Work: Existing TPR-based approaches generate continuous representations of compositional structure by producing an element with the explicit mathematical form of a TPR. To learn this highly specific form, these approaches rely on the algebraic characterisation of compositionality present in formal domains, such as mathematics [38], or language [19, 23, 28, 34, 52] in addition to strong supervision signals from highly structured downstream tasks, such as part-of-speech tagging [23], and answering structured language [28, 51] or mathematics questions [38]. In contrast, our *Soft*

TPR eases these stringent constraints by offering a relaxed specification of inherently continuous compositional structure. This allows our approach to extend continuous representations of compositional structure to an orthogonal and less structured domain, that of visual representation learning, while also reducing reliance on annotated data by instead using weak supervision to learn this relaxed representational form.

3 Preliminaries

3.1 A Formal Framework for Compositional Representations

We adopt a generalised, non-generative version of the definition of *compositional representations* from [49]. Data $x \in X$ is *compositionally-structured* if there exists a decomposition function $\beta : X \to A_1 \times \ldots \times A_n$ decomposing x into constituent parts, i.e. $\beta(x) = \{a_1, \ldots, a_n\}$, where $a_i \in A_i$. A map $\psi : X \to V_F$ produces a *compositional representation* if $\psi(x) = C(\psi_1(a_1), \ldots, \psi_n(a_n))$ where $\psi_i : A_i \to V_i$ denote *component functions* that independently embed the parts of x into vector spaces, and $C : V_1 \times \ldots \times V_n \to V_F$ denotes a *composition function* that combines the embedded parts of x together to form the overall representation. Intuitively, this definition enforces a faithful structural correspondence between the constituency structure of the data, x (i.e., the parts, $\{a_1, \ldots, a_n\}$) and the constituency structure of the representation, $\psi(x)$ (i.e., the *embedded* parts, $\{\psi_1(a_1), \ldots, \psi_n(a_n)\}$) (assuming C is invertible).

We formalise a *symbolic* compositional representation, $\psi_s(x)$, as a compositional representation where C is a concatenation operation. Thus, $\psi_s(x) = \left(\psi_1(a_1)^T, \ldots, \psi_n(a_n)^T\right)^T$ for any *symbolic compositional representation*, $\psi_s(x)$. Clearly, the aforementioned disentanglement methods [8, 10, 13, 14, 17, 22, 24, 26, 31, 32, 33, 35, 36, 37, 47] all fit this framework. While we observe that the fundamentally *symbolic* definition of C as concatenation produces an inherent misalignment with the continuous vector spaces of deep learning, it has one notable benefit: the embedded FoVs, $\{\psi_i(a_i)\}$, can be easily recovered from the representation, $\psi_s(x)$, by simply partitioning $\psi_s(x)$.

It is highly intuitive that for any compositional representation, $\psi(x)$, to be broadly useful, the FoVs, $\{a_i\}$, should be easily recoverable from $\psi(x)$ (here we assume the ψ_i's are invertible, and so, that this property corresponds to being able to recover the representational components, $\{\psi_i(a_i)\}$ from $\psi(x)$). We thus explore whether an alternative C exists that simultaneously 1) combines the embedded FoVs into the overall representation in an inherently *continuous* manner and 2) preserves the direct recoverability of the embedded FoVs.

3.2 The TPR Framework

TPR [3] is a specific type of representation that is *compositional*, *continuous*, and under certain conditions, ensures the *direct recoverability* of the embedded parts $\{\psi_i(a_i)\}$ from the overall representation. We briefly introduce essential aspects of the framework, deferring further details and formal proofs to Appendix A. The TPR framework views compositionally-structured objects as possessing a number of (potentially infinite) roles[1], where each role is bound to a corresponding filler. It thus defines the constituent parts $\{a_i\}$ of any compositionally-structured object as a set of *role-filler bindings*. This role-filler binding formalism has predominantly been applied in the natural language domain [19, 28, 34, 52], with fillers often corresponding to words and roles to grammatical categories (e.g., the word cat as a filler, and the the category noun as a role). We translate this formalism into the domain of visual representation learning by informally equating roles as FoV *types*, and fillers as FoV *values*, e.g., $\{\mathrm{floor\ colour}, \mathrm{wall\ colour}, \mathrm{object\ colour}, \mathrm{object\ size}, \mathrm{object\ shape}, \mathrm{orientation}\}$ and $\{\mathrm{blue}, \mathrm{magenta}, \mathrm{orange}, \mathrm{green}, \mathrm{small}, \mathrm{medium}, \mathrm{large}, \mathrm{oblong}, \mathrm{cube}, \ldots\}$ respectively for the Shapes3D domain of Figure 1. The *binding* of a filler, f, from a set F of N_F fillers, to a role, r, from a set R of N_R roles, such as the filler $\mathrm{magenta}$ to the role $\mathrm{object\ colour}$ conveys a sort of filler-specific, role-modulated semantic content, and is denoted by f/r. The compositional structure of the image, x, in Figure 1 would thus correspond to the following set of role-filler bindings: $\beta(x) = \{\mathrm{magenta/floor\ colour}, \mathrm{orange/wall\ colour}, \mathrm{aqua/object\ colour}, \mathrm{large/object\ size}, \mathrm{oblong/object\ shape}\}$.

[1] We assume a finite set of N_R roles, an intuitive assumption to make in the context of visual representation learning.

To produce the TPR, the roles and fillers for each binding in x are independently embedded using role and filler embedding functions, $\xi_R : R \to V_R, \xi_F : F \to V_F$ respectively. To produce an embedding of the binding f/r, a tensor product, which we denote by $\otimes$, is then taken over the embedded role, $\xi_R(r)$, and embedded filler, $\xi_F(f)$, comprising the binding. Finally, a summation is performed over *all* embedded bindings in x to produce the overall representation. More concretely, the TPR, $\psi_{tpr}(x)$, is defined as:

$$\psi_{tpr}(x) := \sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i), \tag{1}$$

where $m : \{1, \ldots, N_R\} \to \{1, \ldots, N_F\}$ is a matching function associating each of the roles to the unique filler it binds to in the decomposition of x [2].

We now place the TPR in the formal framework of Section 3.1, by observing that the TPR defines the component functions as $\psi_i(f_{m(i)}, r_i) := \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$, and the composition function, C, as ordinary vector space addition. As the TPR does not *concatenate*, but rather, *additively superimposes* all representational components $\{\psi_i(f_{m(i)}, r_i)\}$ together to produce the overall representation, it instantiates an inherently *continuous* representation of compositional structure. Defining C as ordinary addition, however, prompts the question of whether the summed up representational components $\{\psi_i(f_{m(i)}, r_i)\}$ can be recovered from the representation, $\psi_{tpr}(x)$. Remarkably, due to the special form of the TPR, each representational component $\psi_i(f_{m(i)}, r_i)$ corresponding to an embedded role-filler binding, $\xi_F(f_{m(i)}) \otimes \xi_R(r_i)$ can be faithfully recovered from the TPR through a process referred to as *unbinding* (see A.2 for more details). More concretely, provided that all role embedding vectors $\{\xi_R(r_i)\}$ are linearly independent, the embedded filler bound to the i-th role can be *unbound* from the representation, $\psi_{tpr}(x)$, by taking a (tensor) inner product between $\psi_{tpr}(x)$ and the i-th *unbinding* vector, u_i:

$$\psi_{tpr}(x)u_i = \left(\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)\right) u_i = \xi_F(f_{m(i)}), \tag{2}$$

where u_i is a vector corresponding to the i-th column of U^T, the (left) inverse of the matrix formed by taking all (linearly independent) role embeddings as columns. By repeating the *unbinding* procedure using each of the N_R unbinding vectors, the embedded fillers bound to each role, and hence, the embeddings $(\xi_F(f_{m(i)}), \xi_R(r_i))$ comprising each binding embedding, $\xi_F(f_{m(i)}) \otimes \xi_R(r_i)$, can be recovered from the overall representation, $\psi_{tpr}(x)$. Thus, provided that the linear independence condition is satisfied, the TPR represents a *continuous* compositional representation that retains the key benefit of *symbolic* compositional representations: the direct recoverability of the representational parts, $\{\psi_i(f_{m(i)}, r_i)\}$.

4 Methods

4.1 Soft TPR: An Extension to the TPR Framework

The TPR's highly specific representational form, $\psi_{tpr}(x) := \sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$, can only be satisfied by a discrete subset of points in the underlying representational space, $V_F \otimes V_R$. This imposes an arduous learning task on representation learners: to parameterise the highly constrained map from the data manifold to a *discrete subset* of points. This representational form additionally assumes a strict algebraic definition of compositionality that corresponds to a set of bindings, where each binding comprises a *single* role and a *single* filler, precluding the TPR from representing *quasi*-compositional objects that only *approximately* satisfy this strict, algebraic definition of compositional structure (e.g., French liaison consonants, where a weighted sum of *multiple* fillers, rather than a *single* filler, bind to a role [9]). Our primary insight is that both these drawbacks can be mitigated by *continuously relaxing* the TPR specification (see B.1). This relaxation allows for a wider variety of mappings within a 'cloud' around each TPR (represented by the translucent circular regions in Figure 1c), which eases the difficulty of representation learning. Furthermore, it relaxes the rigid role-filler based specification of compositional structure into a *softer*, less rigid notion, enabling the representation of nuanced, quasi-compositional data. We thus introduce the ***Soft TPR***, a continuously relaxed, less stringently defined variant of the explicit TPR that *simultaneously* retains the TPR's key

[2] We omit the dependence of m on x for notational clarity.

properties of 1) *continuous* compositional structure, and 2) direct recoverability of representational parts $\{\big(\xi_F(f_{m(i)}), \xi_R(r_i)\big)\}$ from the overall representation.

Consider an element z in $V_F \otimes V_R$, the vector space underlying TPRs produced by an arbitrary role embedding function $\xi_R : R \to V_R$ and an arbitrary filler embedding function $\xi_F : F \to V_F$. If z is sufficiently 'close' to some TPR, ψ_{tpr}[3], quantified by $||z - \psi_{tpr}||_F < \epsilon$, where $||A||_F$ denotes the Frobenius norm of the rank-2 tensor, A, and ϵ is some small, scalar valued quantity, then this distance metric induces the approximation $z \approx \psi_{tpr}$. By performing unbinding of the i-th filler to both sides of the approximation, we have:

$$zu_i \approx \psi_{tpr} u_i = \left(\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i) \right) u_i = \xi_F(f_{m(i)}), \tag{3}$$

$$= \xi_F(f_{m(i)}) + \epsilon_i =: \tilde{f}_i, \text{ where } \epsilon_i = zu_i - \xi_F(f_{m(i)}). \tag{4}$$

Thus, performing unbinding on an element z in $V_F \otimes V_R$ where the sufficient closeness condition holds recovers a *soft filler* embedding, $\tilde{f}_i$, that approximates the true filler embedding, $\xi_F(f_{m(i)})$, of the filler bound to role r_i for ψ_{tpr} with approximation error ϵ_i. We define such elements as *Soft TPRs*, noting that these elements both 1) softly approximate the continuous compositional structure captured by some explicit TPR, ψ_{tpr}, as z is sufficiently 'close' to ψ_{tpr} based on the chosen distance metric, and 2) approximately preserve the recoverability of the representational components $\{(\xi_F(f_{m(i)}), \xi_R(r_i))\}$ of the explicit TPR, ψ_{tpr}, they approximate, with the only difference being that *soft* filler embeddings, $\tilde{f}_i$ are returned in place of the actual filler embeddings, $\xi_F(f_{m(i)})$. Defining *Soft TPRs* in this way 1) provides a less restrictive representational specification that can be satisfied by any arbitrary element from $V_F \otimes V_R$ provided that, for some explicit TPR, ψ_{tpr}, the sufficient closeness requirement, $||z - \psi_{tpr}||_F < \epsilon$, holds, and 2) allows learned representations to embody a more flexible, *relaxed* notion of compositional structure.

4.2 Soft TPR Autoencoder: A Concrete Implementation of Learning Soft TPRs

We define our vector spaces of interest over the reals as $V_F := \mathbb{R}^{D_F}$ and $V_R := \mathbb{R}^{D_R}$ where D_F, D_R denote the dimensionality of the filler and role embedding spaces. The main insight underlying our method is that, as the *Soft TPR* is effectively any arbitrary element from a vector space[4] $\mathbb{R}^{D_F \cdot D_R}$ that is sufficiently close to some explicit TPR, any $(D_F \cdot D_R)$-dimensional vector produced by an encoder in a standard autoencoding framework can be treated as a Soft TPR *candidate*. This suggests that a simple autoencoding framework only needs to be slightly modified to produce Soft TPRs. Briefly speaking, our Soft TPR Autoencoder contains a standard encoder, E, the TPR decoder, and a standard decoder, D, where the encoder output, z, corresponds to the Soft TPR. At a high level, our framework aims to ensure two properties: 1) representational form, and 2) representational content. Representational form requires that the encoder output, z, has the desired Soft TPR *form* (i.e. that $||z - \psi_{tpr}||_F < \epsilon$ for some TPR, ψ_{tpr}). However, having a Soft TPR form alone is insufficient; the representation produced by the autoencoder must also reflect the *true* role-filler content of the data to be a good representation, as required by the aim of *representational content*. These 2 properties are (mostly) respectively achieved using the unsupervised and weakly supervised components of our method.

Representational Form: to encourage the autoencoder to produce elements that are Soft TPRs, we penalise the Euclidean distance $||z - \psi^*_{tpr}||_2$ between the encoder output, z, and the explicit TPR, ψ^*_{tpr}, that z best approximates. To obtain ψ^*_{tpr}, needed to penalise the above distance, we derive an explicit analytical form for ψ^*_{tpr} and construct elements satisfying this analytical form using a role embedding matrix, M_{ξ_R} containing N_R D_R-dimensional role embedding vectors, $\{\xi_R(r_i)\}$, and a filler embedding matrix, M_{ξ_F} containing N_F D_F-dimensional filler embedding vectors, $\{\xi_F(f_i)\}$. To define an explicit analytical expression for ψ^*_{tpr}, we are guided by the intuition that the TPR that z best approximates should have an explicit dependency on the *soft filler bindings* of z, and so, elect to

[3]We occasionally omit the dependence on x for notational clarity.

[4]Due to isomorphism of vector spaces $\mathbb{R}^{D_F} \otimes \mathbb{R}^{D_R} \cong \mathbb{R}^{D_F \cdot D_R}$, we henceforth use vectors from $\mathbb{R}^{D_F \cdot D_R}$ in place of rank-2 tensors from $\mathbb{R}^{D_F} \otimes \mathbb{R}^{D_R}$, and the Euclidean norm instead of the Frobenius norm to align the Soft TPR framework more seamlessly with the autoencoding framework.

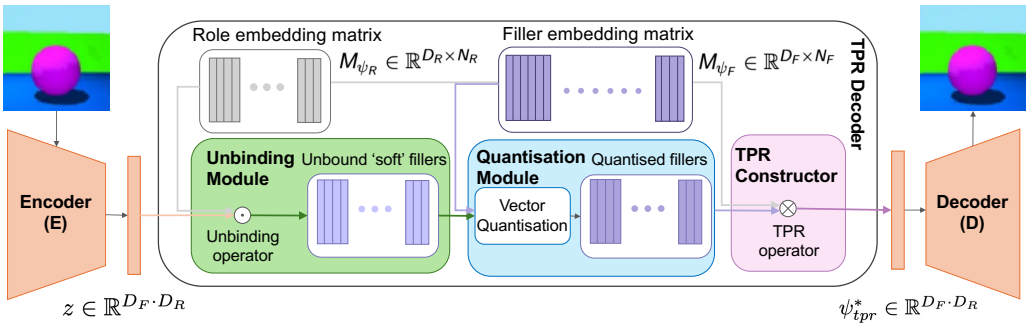

Figure 2: Diagram illustrating the Soft TPR Autoencoder. We encourage the encoder E's output, z, to have the form of a Soft TPR by penalising its distance with the greedily defined, explicit TPR, ψ^*_{tpr} of Equation 5 that z best approximates. ψ^*_{tpr} is recovered using a 3 step process performed by our TPR decoder (center rectangle): 1) unbinding, 2) quantisation, and 3) TPR construction. The decoder, D, reconstructs the input image using ψ^*_{tpr}.

use the following, greedily optimal definition:

$$\psi^*_{tpr} := \sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i), \text{where } m(i) := \arg\min_j ||\tilde{f}_k - \xi_F(f_j)||_2, \text{and } \tilde{f}_k := z u_i. \quad (5)$$

That is, we define ψ^*_{tpr} as the TPR constructed from explicit filler embeddings $\xi_F(f_j)$ with the smallest Euclidean distance to the *soft* filler embeddings $\tilde{f}_k$ of z. To construct elements satisfying (5), we use a 3-step process carried out by the novel TPR decoder we introduce, visible in Figure 2: **1) Unbinding**: The unbinding module consists of a fixed semi-orthogonal role embedding matrix M_{ξ_R} and recovers the soft filler embeddings $\{\tilde{f}_i\}$ associated with each encoder output, z, by performing the TPR *unbinding* operation. We elaborate on our theoretically-informed reason for this choice of role embedding matrix, how the unbinding vectors are obtained, and for not backpropagating gradient to M_{ξ_R} in B.3.1. **2) Quantisation**: The quantisation module, containing a learnable filler embedding matrix M_{ξ_F}, employs the VQ-VAE vector quantisation algorithm [11] to both 1) learn the explicit filler embeddings, and 2) quantise the soft filler embeddings $\{\tilde{f}_1, \ldots, \tilde{f}_{N_R}\}$ produced by the unbinding module into the explicit filler embeddings $\{\xi_F(f_1), \ldots, \xi_F(f_{N_R})\}$ with the smallest Euclidean distances. **3) TPR Construction**: The TPR construction module recovers ψ^*_{tpr} by simply performing the TPR operation $\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$ over the fixed, semi-orthogonal role embedding vectors with the corresponding explicit filler embeddings produced by the quantisation module. To ensure the quantised filler embeddings $\{\xi_F(f_{m(i)})\}$ depend explicitly on the reconstructed image, we pass the output of the TPR decoder, ψ^*_{tpr} to the image decoder, D, for reconstruction. The overall unsupervised loss, $\mathcal{L}_u$, is thus given by the following, where s denotes the stop-gradient operator, β a hyperparameter, and $\mathcal{L}_r$ any suitable image-based reconstruction loss (we use L_2):

$$\mathcal{L}_u := \underbrace{||z - \psi^*_{tpr}||_2^2}_{\text{Soft TPR penalty}} + \underbrace{\mathcal{L}_r(x, D(\psi^*_{tpr}))}_{\text{reconstruction loss}} + \underbrace{\sum_i \frac{1}{N_R} \left(||\text{s}[\xi_F(f_{m(i)})] - \tilde{f}_i||_2^2 + \beta ||\xi_F(f_{m(i)}) - \text{s}[\tilde{f}_i]||_2^2 \right)}_{\text{VQ-VAE quantisation loss}}, \quad (6)$$

Representational Content: While the unsupervised loss $\mathcal{L}_u$ encourages the autoencoder to produce encodings z with the desired Soft TPR form, it may not ensure that the content of z accurately reflects the true role-filler semantics of the represented data. To address this, we introduce a weakly supervised loss to ensure that the explicit TPR, z^*_{tpr}, which z best approximates, reflects the ground-truth semantics of the image. We employ a match-pairing context similar to [13, 22, 33, 35], where image pairs (x, x') share the same role-filler bindings for all but one of the roles, r_i, and the identity of r_i is known, but not any of the fillers, or role-filler bindings. Our intuition is that, for the role-filler binding embeddings of z to reflect the semantics of the represented image, the Euclidean distance between the quantised fillers of x and x' bound to role r_i should be maximal, relative to the distances between the pairs of filler embeddings for all other roles $r_j, j \neq i$. To encourage this, we apply the cross entropy loss corresponding to the 3rd term in Eq 7, where Δq denotes the N_R-dimensional vector with each dimension $(\Delta q)_k$ populated by the Euclidean distance between the quantised fillers of x and x' for role r_k, and l denotes the one-hot vector of dimension N_R with the index for r_i set

to 1. Additionally, we apply a reconstruction loss using the TPRs, $\psi^s_{tpr}(x)$ and $\psi^s_{tpr}(x')$, which are constructed by swapping the quantised filler embeddings of x and x' bound to role r_i, to reconstruct x' and x respectively.

Our final loss, $\mathcal{L}$, is a weighted sum over the unsupervised and weakly supervised loss components, where λ_1 and λ_2 are hyperparameters:

$$\mathcal{L} := \mathcal{L}_u + \lambda_1 \left(\frac{1}{2}\mathcal{L}_r(x, D(\psi^s_{tpr}(x'))) + \frac{1}{2}\mathcal{L}_r(x', D(\psi^s_{tpr}(x))) \right) + \lambda_2 \mathrm{CE}(\Delta q, l), \tag{7}$$

5 Results

To assess the compositional representations produced by our Soft TPR framework, we perform evaluation along three dimensions: **1) Compositional Structure / Disentanglement**: What is the degree to which Soft TPR representations achieve *explicitly compositional structure*? **2) Representation Learner Convergence Rate**: Can representation learners learn the inherently continuous compositional structure embodied in the Soft TPR *faster than symbolic alternatives*? **3) Downstream Models**: Does the *enhanced vector space alignment* produced by the Soft TPR *facilitate benefits for downstream models* using compositional representations?

We benchmark against a suite of weakly supervised disentanglement baselines: Ada-GVAE [33], GVAE [22], ML-VAE [13], SlowVAE [39], and the GAN-based model of [35], which we henceforth refer to as 'Shu'. These models all produce symbolic compositional representations corresponding to a concatenation of *scalar-valued* FoV tokens. Like our model, Ada-GVAE, GVAE, ML-VAE, and Shu are trained with paired samples (x, x') sharing values for all but a subset of FoVs types (roles), I. ML-VAE, GVAE, and Shu assume access to I, the FoV types (roles) that differ between x and x', matching our model's level of supervision, while Ada-GVAE does not. We thus modify Ada-GVAE (method detailed in Appendix C.2.2) for more direct comparability, denoting our modification by Ada-GVAE-k. In contrast, SlowVAE is trained with pairs of samples where *all* underlying FoV values change, and also assumes that this change can be characterised as a sample from a Laplacian. We additionally benchmark against 2 baselines producing *vector-tokened* compositional representations: COMET [36], and Visual Concept Tokeniser (VCT) [47]. These *vector-tokened* models cannot be directly compared to our model as they are fully unsupervised, however, we include them for completeness. We train 5 instances of each representation learning model using 5 random seeds for 200,000 iterations across all datasets, and report results averaged over the 5 random runs.

5.1 Compositional Structure / Disentanglement

To evaluate the degree to which Soft TPRs achieve explicitly compositional structure, we quantify representational disentanglement using standard disentanglement datasets Cars3D [4], MPI3D [21], and Shapes3D [24] (see C.4.1 for further details). As can be seen in Table 1, our model achieves state-of-the-art disentanglement for all datasets, with notable DCI metric increases of 29% and 74% on the 2 more challenging datasets of Cars3D and MPI3D respectively. To rule out the possibility that the improvement in disentanglement produced by our model is due to a slight increase of 13,568 (Cars3D), 1,824 (Shapes3D), 1,600 (MPI3D) learnable parameters produced by the addition of the filler embedding matrix, M_{ξ_F}, to a standard (variational) encoder-decoder framework, we modify models with fewer parameters to have an identical number of parameters as ours. We note that the modifications are applicable only for the scalar-tokened baselines, as our model has tens of millions less parameters than COMET and VCT. In line with [40], we observe that the performance of scalar-tokened disentanglement models remains fairly consistent, or even deteriorates, when the number of learnable parameters increases, so we defer control experiment results, and our associated method to Appendix C.2.4.

Key Implication: The Soft TPR's superior level of explicit compositionality (as quantified by the disentanglement metrics) in a controlled, parameter-count environment, suggests that its fundamentally *continuous* representational form is potentially *easier for deep learning models to learn* compared to *symbolic* representational forms characterising existing disentanglement work.

5.2 Representation Learner Convergence Rate

To evaluate whether the inherently *continuous* compositional form of the Soft TPR can be learned more quickly than symbolic alternatives, we consider representations produced at 100, 1,000, 10,000, 100,000 and 200,000 iterations of training, and evaluate 1) their disentanglement, and 2) their utility, as quantified by the performance of downstream models using these representations. For downstream model performance, we consider two commonly used [30, 46, 33] tasks: the classification-based abstract visual reasoning task of [30] and a regression task involving the prediction of continuous FoV values for the disentanglement datasets. Downstream models are evaluated on a fixed, held-out test set for both tasks. While our framework does not promote faster disentanglement convergence (results in Appendix C.4.1), it interestingly, promotes *accelerated* learning of *useful* representations for *both* downstream tasks compared to baselines. The downstream performance improvements are particularly pronounced in the low iteration regime of 100 iterations of representation learner training, as demonstrated by the improvements of $10\%, 10\%$, and 31% in Table 2 and the 27% improvement in Table 3. To ensure fair comparison, we embed baseline representations of both higher, and lower dimensionality into the same space as our model. For each baseline model, we take the best result from *either* the original, or the modified model (denoted by †), and present the full suite of results, and details of our embedding method and downstream model setup in Appendix C.

Key Implication: The representation learning convergence results suggests that while our model may not learn the *explicit* compositional structure captured by disentanglement metrics more quickly than baselines (though in the limit, the greatest possible disentanglement is higher), it learns *useful information for downstream tasks more quickly* than baselines, where that useful information is encoded in the *relaxed* compositional structure of the Soft TPR.

Table 1: FactorVAE and DCI scores. Additional results in Section C.3.3

Models	Cars3D		Shapes3D		MPI3D	
	FactorVAE score	DCI score	FactorVAE score	DCI score	FactorVAE score	DCI score
Symbolic scalar-tokened compositional representations						
Slow-VAE	0.902 ± 0.035	0.509 ± 0.027	0.950 ± 0.032	0.850 ± 0.047	0.455 ± 0.083	0.355 ± 0.027
Ada-GVAE-k	0.947 ± 0.064	*0.664 ± 0.167*	*0.973 ± 0.006*	**0.963 ± 0.077**	0.496 ± 0.095	0.343 ± 0.040
GVAE	0.877 ± 0.081	0.262 ± 0.095	0.921 ± 0.075	0.842 ± 0.040	0.378 ± 0.024	0.245 ± 0.074
ML-VAE	0.870 ± 0.052	0.216 ± 0.063	0.835 ± 0.111	0.739 ± 0.115	0.390 ± 0.026	0.251 ± 0.029
Shu	0.573 ± 0.062	0.032 ± 0.014	0.265 ± 0.043	0.017 ± 0.006	0.287 ± 0.034	0.033 ± 0.008
Symbolic vector-tokened compositional representations						
VCT	*0.966 ± 0.029*	0.382 ± 0.080	0.957 ± 0.043	0.884 ± 0.013	*0.689 ± 0.035*	*0.475 ± 0.005*
COMET	0.339 ± 0.008	0.024 ± 0.026	0.168 ± 0.005	0.002 ± 0.000	0.145 ± 0.024	0.005 ± 0.001
Fully continuous compositional representations						
Ours	**0.999 ± 0.001**	**0.863 ± 0.027**	**0.984 ± 0.012**	*0.926 ± 0.028*	**0.949 ± 0.032**	**0.828 ± 0.015**

Table 2: FoV regression R^2 scores (100 iterations of representation learner training).

Models	Cars3D	Shapes3D	MPI3D
	Symbolic scalar-tokened compositional representations		
Slow-VAE	0.233 ± 0.048	0.600 ± 0.048	*0.557 ± 0.012*
Ada-GVAE-k	0.307 ± 0.084	0.684 ± 0.059	0.519 ± 0.023
GVAE	0.319 ± 0.073	0.565 ± 0.034	0.511 ± 0.037
ML-VAE	0.317 ± 0.058	0.551 ± 0.059	0.504 ± 0.016
Shu†	0.012 ± 0.007	0.461 ± 0.075	0.299 ± 0.040
	Symbolic vector-tokened compositional representations		
VCT	0.080 ± 0.001	*0.886 ± 0.033*	0.316 ± 0.016
COMET†	*0.484 ± 0.477*	0.474 ± 0.062	0.240 ± 0.010
	Fully continuous compositional representations		
Ours	**0.531 ± 0.054**	**0.981 ± 0.003**	**0.732 ± 0.012**

Table 3: Abstract visual reasoning accuracy (100 iterations of representation learner training).

Models	Abstract visual reasoning dataset
	Symbolic scalar-tokened
Slow-VAE†	0.552 ± 0.035
Ada-GVAE-k†	*0.631 ± 0.037*
GVAE†	0.554 ± 0.031
ML-VAE†	0.550 ± 0.025
Shu	0.208 ± 0.052
	Symbolic vector-tokened
VCT	0.440 ± 0.033
COMET†	0.348 ± 0.069
	Fully continuous
Ours	**0.804 ± 0.016**

5.3 Downstream Models

To evaluate whether the enhanced alignment between compositional representations and continuous vector spaces produced by our Soft TPR benefits downstream models, we examine downstream 1) sample efficiency, and 2) raw performance in the low sample regime. We use the previously mentioned tasks of abstract visual reasoning and FoV regression. To quantify sample efficiency, in line with [25], we use a ratio-based metric obtained by dividing the performance of the downstream model when trained using a restricted number of samples (100, 250, 500, 1,000 and 10,000 samples), by its performance when trained using all samples (between 19,104-1,036,800 samples depending on the task). As illustrated in Table 4, our model has superior sample efficiencies compared to baselines,

especially in the most restrictive case where downstream models have access to *only 100* samples produced by representation learners, achieving a 93% improvement. The Soft TPR representations produced by our model additionally produce substantial raw performance increases in the low sample regime, as evidenced in Table 4, where its performance in the low-sample regimes of 100 and 200 samples constitutes a respective 138% and 168% improvement, and in Table 5, a 30% improvement.

Table 4: Downstream FoV R^2 scores (odd columns) and sample efficiencies (even columns) on the MPI3D dataset.

	100 samples	100 samples/all	250 samples	250 samples/all
Models	Symbolic scalar-tokened compositional representations			
Slow-VAE	0.127 ± 0.050	0.130 ± 0.051	0.152 ± 0.011	0.155 ± 0.011
Ada-GVAE-k	*0.206 ± 0.031*	0.270 ± 0.037	0.213 ± 0.023	0.279 ± 0.026
GVAE	0.181 ± 0.030	0.234 ± 0.035	0.217 ± 0.023	0.282 ± 0.027
ML-VAE	0.182 ± 0.013	0.236 ± 0.019	*0.222 ± 0.024*	0.288 ± 0.030
Shu	0.151 ± 0.016	*0.343 ± 0.024*	0.211 ± 0.026	*0.482 ± 0.075*
	Symbolic vector-tokened compositional representations			
VCT	0.086 ± 0.051	0.189 ± 0.107	0.119 ± 0.070	0.246 ± 0.137
COMET	-0.051 ± 0.015	0.000 ± 0.000	-0.042 ± 0.018	0.000 ± 0.000
	Fully continuous compositional representations			
Ours	**0.490 ± 0.068**	**0.556 ± 0.078**	**0.594 ± 0.056**	**0.665 ± 0.067**

Table 5: Abstract visual reasoning accuracy in the low-sample regime of 500 samples.

Models	Symbolic scalar-tokened
Slow-VAE	0.196 ± 0.028
Ada-GVAE-k	0.203 ± 0.007
GVAE	0.182 ± 0.013
ML-VAE	0.193 ± 0.012
Shu	0.200 ± 0.010
	Symbolic vector-tokened
VCT	*0.277 ± 0.039*
COMET	0.259 ± 0.016
	Fully continuous
Ours	**0.360 ± 0.033**

Table 6: Sample efficiencies for FoV regression.

Representational form	Cars3D	Shapes3D	MPI3D
TPR (100/all samples)	0.403 ± 0.029	0.428 ± 0.101	**0.569 ± 0.054**
Soft TPR (100/all samples)	**0.705 ± 0.023**	**0.464 ± 0.071**	0.555 ± 0.078
TPR (250/all samples)	0.639 ± 0.055	0.634 ± 0.123	**0.759 ± 0.089**
Soft TPR (250/all samples)	**0.889 ± 0.042**	**0.730 ± 0.038**	0.665 ± 0.067

Table 7: Effect of model properties on disentanglement performance (MPI3D dataset).

Property	DCI Score
- Weak supervision	0.225 ± 0.034
- Explicit filler dependency	0.718 ± 0.051
- Semi orthogonality	0.756 ± 0.039
Full	0.828 ± 0.015

Key Implication: The consistent performance improvement we observe in generic downstream models with no a priori knowledge of the Soft TPR's representational form, suggests the *relaxed*, inherently *continuous* representation of compositional structure embodied by our Soft TPR can be more efficiently leveraged by downstream models compared to symbolic compositional representations, benefiting both sample efficiency, and raw performance in the low sample regime.

5.4 More Ablation Studies

We additionally repeat our full suite of experiments using the explicit TPR, ψ^*_{tpr}, produced by the TPR decoder, in place of our Soft TPR, z. These experiments, a subset of which is presented in Table 6, empirically demonstrate that the Soft TPR's *continuously relaxed* specification of compositional structure confers *exclusive* benefits for both the representation learner and the downstream models not captured by the traditional TPR (see Appendix C.6.1). Note for MPI3D, the explicit TPR has a lower raw R^2 score when fully trained (0.785 ± 0.019 vs 0.882 ± 0.016), contributing to its higher sample efficiency in Table 6. We additionally examine the importance of the following properties of our model in producing explicitly compositional Soft TPR representations: 1) the presence of weak supervision, by setting $\lambda_1 = \lambda_2 = 0$ in Equation 7, 2) the explicit dependency between the quantised filler embeddings and the decoder output, by instead using the Soft TPR to reconstruct the input image, and 3) the semi-orthogonality of the role embedding matrix, M_{ξ_R} by removing this constraint in the random initialisation of M_{ξ_R}, with the results of these ablations illustrated in Table 7.

6 Conclusion

In this work, we address a longstanding issue in the connectionist approach to compositionality: the fundamental mismatch between disentangled representations and the inherently continuous nature of deep learning vector spaces. To overcome this, we introduce *Soft TPR*, a novel, inherently *continuous* compositional representational form that extends Smolensky's Tensor Product Representation, together with the *Soft TPR Autocoder*, a theoretically-principled architecture designed for learning Soft TPRs. Our *flexible*, *continuous* framework yields substantial improvements in the visual domain, enhancing compositional structure, accelerating convergence in representation learners, and boosting efficiency in downstream models. These wide-ranging empirical benefits underscore the importance of rethinking compositional representations to honour deep learning's continuous foundations. Future work will extend our continuous framework to hierarchical forms of compositionality, enabling bound fillers themselves to decompose into role-filler bindings for enhanced representational expressivity.

Acknowledgements

This research has been supported by an UNSW University Postgraduate Award given to B.S. We thank the anonymous reviewers for their valuable feedback, and B. Sphear for insightful discussions.

References

[1] Noam Chomsky. *Syntactic Structures*. The Hague: Mouton, 1957.

[2] Jerry A. Fodor. *The Language of Thought: A Theory of Mental Representation*. Cambridge, MA: Harvard University Press, 1975.

[3] Paul Smolensky. "Tensor product variable binding and the representation of symbolic structures in connectionist systems". In: *Artificial Intelligence* 46.1 (1990), pp. 159–216.

[4] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. "3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model". In: *Advances in Neural Information Processing Systems*. 2012.

[5] Yoshua Bengio. *Deep Learning of Representations: Looking Forward*. 2013. arXiv: 1305.0445 [cs.LG].

[6] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. "Neural Module Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 39–48. URL: https://api.semanticscholar.org/CorpusID:5276660.

[7] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*. 2015. arXiv: 1502.05698 [cs.AI]. URL: https://arxiv.org/abs/1502.05698.

[8] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[9] Paul Smolensky and Matthew A. Goldrick. "Gradient Symbolic Representations in Grammar: The case of French Liaison". In: 2016. URL: https://api.semanticscholar.org/CorpusID:36953611.

[10] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017.

[11] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu koray. "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[12] Tameem Adel, Zoubin Ghahramani, and Adrian Weller. "Discovering Interpretable Representations for Both Deep Generative and Discriminative Models". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 50–59.

[13] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. "Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).

[14] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. *Understanding disentangling in β-VAE*. 2018. arXiv: 1804.03599 [stat.ML].

[15] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. "Isolating Sources of Disentanglement in Variational Autoencoders". In: *Advances in Neural Information Processing Systems*. 2018.

[16] Cian Eastwood and Christopher K. I. Williams. "A Framework for the Quantitative Evaluation of Disentangled Representations". In: *International Conference on Learning Representations*. 2018.

[17] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. "Variational Inference of Disentangled Latent Concepts from Unlabeled Observations". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018.

[18] Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. "Measuring abstract reasoning in neural networks". In: *International conference on machine learning*. 2018, pp. 4477–4486.

[19] Kezhen Chen, Qiuyuan Huang, Hamid Palangi, Paul Smolensky, Kenneth D. Forbus, and Jianfeng Gao. "Natural- to formal-language generation using Tensor Product Representations". In: *CoRR* abs/1910.02339 (2019). arXiv: 1910.02339. URL: http://arxiv.org/abs/1910.02339.

[20] Elliot Creager, David Madras, Joern-Henrik Jacobsen, Marissa Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. "Flexibly Fair Representation Learning by Disentanglement". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[21] Muhammad Waleed Gondal, Manuel Wüthrich, undefinedorđe Miladinović, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. "On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019.

[22] Haruo Hosoya. "Group-based learning of disentangled representations with generalizability for novel contents". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2019, pp. 2506–2513.

[23] Qiuyuan Huang, Li Deng, Dapeng Wu, Chang Liu, and Xiaodong He. "Attentive Tensor Product Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 1344–1351.

[24] Hyunjik Kim and Andriy Mnih. *Disentangling by Factorising*. 2019. arXiv: 1802.05983 [stat.ML].

[25] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4114–4124.

[26] Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. "Disentangling Disentanglement in Variational Autoencoders". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[28] Imanol Schlag and Jürgen Schmidhuber. "Learning to Reason with Third-Order Tensor Products". In: *Advances in Neural Processing Information Systems*. 2019.

[29] Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. "Enhancing the Transformer with Explicit Relational Encoding for Math Problem Solving". In: *ArXiv* abs/1910.06611 (2019). URL: https://api.semanticscholar.org/CorpusID:204575948.

[30] Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. "Are disentangled representations helpful for abstract visual reasoning?" In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019.

[31] Junxiang Chen and Kayhan Batmanghelich. "Weakly Supervised Disentanglement by Pairwise Similarities". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (2020), pp. 3495–3502.

[32] Zheng Ding, Yifan Xu, Weijian Xu, Gaurav Parmar, Yang Yang, Max Welling, and Zhuowen Tu. "Guided Variational Autoencoder for Disentanglement Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[33] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen. "Weakly-Supervised Disentanglement Without Compromises". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6348–6359.

[34] R. Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. "Tensor Product Decomposition Networks: Uncovering Representations of Structure Learned by Neural Networks". In: *Proceedings of the Society for Computation in Linguistics 2020*. Association for Computational Linguistics, 2020, pp. 277–278. URL: https://aclanthology.org/2020.scil-1.34.

[35] Rui Shu, Yining Chen, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Weakly Supervised Disentanglement with Guarantees". In: *International Conference on Learning Representations*. 2020.

[36] Yilun Du, Shuang Li, Yash Sharma, B. Joshua Tenenbaum, and Igor Mordatch. "Unsupervised Learning of Compositional Energy Concepts". In: *Advances in Neural Information Processing Systems*. 2021.

[37] Insu Jeon, Wonkwang Lee, Myeongjang Pyeon, and Gunhee Kim. "IB-GAN: Disentangled Representation Learning with Information Bottleneck Generative Adversarial Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9 (May 2021), pp. 7926–7934. DOI: 10.1609/aaai.v35i9.16967. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16967.

[38] Yichen Jiang, Asli Celikyilmaz, Paul Smolensky, Paul Soulos, Sudha Rao, Hamid Palangi, Roland Fernandez, Caitlin Smith, Mohit Bansal, and Jianfeng Gao. "Enriching Transformers with Structured Tensor-Product Representations for Abstractive Summarization". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 4780–4793. DOI: 10.18653/v1/2021.naacl-main.381. URL: https://aclanthology.org/2021.naacl-main.381.

[39] David A. Klindt, Lukas Schott, Yash Sharma, Ivan Ustyuzhaninov, Wieland Brendel, Matthias Bethge, and Dylan Paiton. "Towards Nonlinear Disentanglement in Natural Data with Temporal Sparse Coding". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=EbIDjBynYJ8.

[40] Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. "The role of Disentanglement in Generalisation". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=qbH974jKUVy.

[41] Sungho Park, Sunhee Hwang, Dohyung Kim, and Hyeran Byun. "Learning Disentangled Representation for Fair Facial Attribute Classification via Fairness-aware Information Alignment". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021), pp. 2403–2411. DOI: 10.1609/aaai.v35i3.16341. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16341.

[42] L. Schott, J. von Kügelgen, F. Träuble, P. Gehler, C. Russell, M. Bethge, B. Schölkopf, F. Locatello, and W. Brendel. "Visual Representation Learning Does Not Generalize Strongly Within the Same Domain". In: *ICLR 2021 - Workshop on Generalization beyond the training distribution in brains and machines*. 2021.

[43] Frederik Träuble, Elliot Creager, Niki Kilbertus, Francesco Locatello, Andrea Dittadi, Anirudh Goyal, Bernhard Schölkopf, and Stefan Bauer. "On Disentangled Representations Learned from Correlated Data". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 10401–10412.

[44] Milton Montero, Jeffrey Bowers, Rui Ponte Costa, Casimir Ludwig, and Gaurav Malhotra. "Lost in Latent Space: Examining failures of disentangled models at combinatorial generalisation". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 10136–10149.

[45] Zoltán Gendler Szabó. "Compositionality". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Fall 2022. Metaphysics Research Lab, Stanford University, 2022.

[46] Tao Yang, Xuanchi Ren, Yuwang Wang, Wenjun Zeng, and Nanning Zheng. "Towards Building A Group-based Unsupervised Representation Disentanglement Framework". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=YgPqNctmyd.

[47] Tao Yang, Yuwang Wang, Yan Lu, and Nanning Zheng. "Visual Concepts Tokenization". In: *Advances in Neural Information Processing Systems*. 2022.

[48] H. Zhang, Y.-F. Zhang, W. Liu, A. Weller, B. Schölkopf, and E. Xing. "Towards Principled Disentanglement for Domain Generalization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8024–8034. URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Zhang_Towards_Principled_Disentanglement_for_Domain_Generalization_CVPR_2022_paper.pdf.

[49] Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. "Compositional Generalization from First Principles". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=LqOQ1uJmSx.

[50] Haoyang Li, Xin Wang, Zeyang Zhang, Haibo Chen, Ziwei Zhang, and Wenwu Zhu. "Disentangled Graph Self-supervised Learning for Out-of-Distribution Generalization". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=OS0szhkPmF.

[51] Taewon Park, Inchul Choi, and Minho Lee. "Attention-based Iterative Decomposition for Tensor Product Representation". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=FDb2JQZsFH.

[52] Qiuyuan Huang, Paul Smolensky, Xiaodong He, Li Deng, and Dapeng Wu. "Tensor Product Generation Networks for Deep NLP Modeling". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

Appendix

A TPR Framework

In this section, we provide additional details regarding Smolensky's TPR framework [3], as well as formal proofs of presented results. We refer interested readers to [3] for a more comprehensive dive into this fascinating framework.

A.1 Additional Details

By defining the constituent components of any *compositional object* as a set of role-filler bindings, the TPR defines the decomposition function, β, of Section 3.1 that maps from a set of compositional objects, X, to a set of parts, more explicitly as follows [3]:

$$\beta : X \to 2^{F \times R}; x \to \{(f, r) | f/r\}, \tag{8}$$

where F denotes a set of fillers, and R denotes a set of roles. Note that in contrast to the formal definition of β we use in Section 3.1, which assumes each $x \in X$ is decomposable into a set of n parts, the above decomposition allows objects to be decomposed into a *variably-sized* set of role-filler bindings, with this set corresponding to an element in the powerset of $F \times R$. For the visual representation learning domain we consider, all considered disentanglement datasets clearly have the property of being decomposable into a *fixed* size set of role-filler bindings, as all images in

these datasets contain the same number of FoV *types* and each FoV type is bound to a FoV *token*. Due to this property, we can take β as a special subcase of the generalised definition in Equation 8:

$$\beta : X \to \mathcal{A} : x \to \{(f_{m(i)}, r_i) | f_{m(i)}/r_i\}. \tag{9}$$

where $m : \{1, \ldots, N_R\} \to \{1, \ldots, N_F\}$ denotes a matching function that associates each role r_i with the filler it binds to in $\beta(x)$ (we again drop the dependence of m on x for ease of notation), and $\mathcal{A}$ denotes the set of all possible bindings produced by binding a filler to each of the N_R roles, with size $\binom{N_F+N_R-1}{N_R}$ (we assume the same filler can bind to multiple roles).

A.2 Formal Proofs

We now formally prove that the TPR with β defined in 9 has form $\psi_{tpr}(x) = C(\psi_1(a_1), \ldots, \psi_n(a_n))$ and thus corresponds to the definition of a *compositional representation* in Section 3.1.

Proof. We denote the role embedding and filler embedding functions as $\xi_R : R \to V_R, \xi_F : F \to V_F$ respectively.

By definition of the TPR in Eq 1, we have that:

$$\psi_{tpr}(x) := \sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i).$$

and hence,

$$\psi_{tpr}(x) = \sum_i \psi_i(f_{m(i)}, r_i),$$

where $a_i := (f_{m(i)}, r_i) \in \beta(x)$, $\psi_i : F \times R \to V_F \otimes V_R; (f_{m(i)}, r_i) \to \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$, and C is ordinary vector space addition. Hence, almost trivially, $\psi_{tpr}(x)$ clearly has the required form to be a *compositional representation*. □

Now, we prove the recoverability of the embedded components $\{\psi_i(f_{m(i)}, r_i)\}$ from the TPR, $\psi_{tpr}(x)$, provided that the set of all role embedding vectors, $\{\xi_R(r_i)\}$, are linearly independent. Similar variants of this proof can be found in [3], [19].

Proof. Assume the set of all role embedding vectors $\{\xi_R(r_i)\}$ are linearly independent. Then, the role embedding matrix, $M_R := (\xi_R(r_1) \ldots \xi_R(r_{N_R}))$ formed by taking the role embedding vectors as columns, has a left inverse, U, such that:

$$UM_{\xi_R} = I_{N_R \times N_R}.$$

Hence, we have that $(UM_{\xi_R})_{ij} = U_{i:}M_{\xi_R:j} = I_{ij}$.

For ease of notation, let u_i denote the i-th column of U^T, and note that $\xi_R(r_j)$ clearly corresponds to $M_{\xi_R:j}$. So, $U_{i:}M_{\xi_R:j} = (U^T_{:i})^T M_{\xi_R:j} = u_i^T \xi_R(r_j) = I_{ij}$.

Hence, we have that:

$$u_i^T \xi_R(r_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise.} \end{cases}$$

Using the definition of $\psi_{tpr}(x)$, $\psi_{tpr}(x) = \sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$, we apply the (tensor) inner product of $\psi_{tpr}(x)$ with u_i:

$$\begin{aligned}\psi_{tpr}(x) &= \left(\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)\right) u_i \\ &= \left(\sum_i \xi_F(f_{m(i)}) \xi_R(r_i)^T\right) u_i \\ &= \sum_i \xi_F(f_{m(i)}) \delta_{ji} \\ &= \xi_F(f_{m(i)}).\end{aligned}$$

Thus, the embedding of the filler, $f_{m(i)}$, bound to each role, r_i, can be recovered through use of a tensor inner product with the *unbinding* vector, u_i, corresponding to the i-th column of U^T. Note that the representational components of $\psi_{tpr}(x)$, i.e., the embedded bindings, $\psi_i(f_{m(i)}, r_i)$ are fully determined by the embedding of the role, $\xi_R(r_i)$, and filler, $\xi_F(f_{m(i)})$, comprising the binding, as $\xi_F(f_{m(i)}, r_i)$ simply corresponds to their tensor product. Thus, recovering $(\xi(f_{m(i)}), \xi_R(r_i))$ for each binding in $\beta(x)$ corresponds to recovering the representational component, $\psi_i(f_{m(i)}, r_i)$. So, provided the set of role embeddings are linearly independent, and they can be obtained (e.g. through a look-up table of role embeddings), all representational components, $\psi_i(f_{m(i)}, r_i)$ can be directly recovered from the overall TPR representation, $\psi_{tpr}(x)$. □

B Soft TPR Framework

In this section, we provide additional information regarding our Soft TPR framework, more extensively detailing our theoretically-informed rationale behind certain design decisions, as well as providing all necessary model architecture information for our Soft TPR Autoencoder to replicate our results (note that code is available at `https://github.com/gomb0c/soft_tpr/`).

B.1 Shortcomings of the TPR and How Soft TPR Helps

In this subsection, we provide a more detailed explanation regarding the fundamental limitations produced by the TPR's stringent specification (Eq 1). For concreteness, we use an example. We consider the following set of roles and fillers, $R = \{\text{shape}, \text{colour}\}, F = \{\text{red}, \text{blue}, \text{square}\}$. Additionally, we define the role $\xi_R : R \to \mathbb{R}^2$ and filler $\xi_F : F \to \mathbb{R}^3$ embedding functions as follows:

$$\xi_R(\text{shape}) = \begin{bmatrix}1\\0\end{bmatrix},$$

$$\xi_R(\text{colour}) = \begin{bmatrix}1\\1\end{bmatrix},$$

$$\xi_F(\text{red}) = \begin{bmatrix}1\\2\\3\end{bmatrix},$$

$$\xi_F(\text{blue}) = \begin{bmatrix}2\\2\\3\end{bmatrix},$$

$$\xi_F(\text{square}) = \begin{bmatrix}0\\0\\1\end{bmatrix},$$

1. **Discrete Mapping**: Consider the set of possible TPRs, T, that can be produced by R, F, ξ_R, ξ_F as defined above. We have:

$$\psi_{tpr}(x_{\text{red\_square}}) = \xi_F(\text{red}) \otimes \xi_R(\text{colour}) + \xi_F(\text{square}) \otimes \xi_R(\text{shape})$$
$$= \begin{bmatrix}1\\2\\3\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} \cong \begin{bmatrix}1\\2\\3\\1\\2\\3\end{bmatrix} + \begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix}$$
$$= \begin{bmatrix}1\\2\\4\\1\\2\\3\end{bmatrix}$$

$$\psi_{tpr}(x_{\text{blue\_square}}) = \xi_F(\text{blue}) \otimes \xi_R(\text{colour}) + \xi_F(\text{square}) \otimes \xi_R(\text{shape})$$
$$= \begin{bmatrix}2\\2\\3\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} \cong \begin{bmatrix}2\\2\\3\\2\\2\\3\end{bmatrix} + \begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix}$$
$$= \begin{bmatrix}2\\2\\4\\2\\2\\3\end{bmatrix}$$

These 2 possible TPRs form a discrete, 2 element subset of the underlying representational space, $\mathbb{R}^6$, $T = \{(2\,2\,4\,2\,2\,3)^T, (1\,2\,4\,1\,2\,3)^T\}$. Relaxing the TPR specification to the ***Soft TPR*** allows any point, z, in the underlying representational space, $\mathbb{R}^6$ to qualify as a Soft TPR provided that for some explicit TPR, $\psi_{tpr} \in T$, the sufficient closeness requirement holds: $||z - \psi_{tpr}||_2 < \epsilon$. So, if we consider the set of possible Soft TPRs, T_S, we have = $T_S := \{(2\,2\,4\,2\,2\,3)^T + \alpha, (1\,2\,4\,1\,2\,3)^T + \alpha : |\alpha| < \epsilon\}$. Hence, T_S clearly has strictly more points than T, so there should be more functions parameterising the map from the observed data to T_S compared to T. Furthermore, in contrast to T, which contains discrete (singular) points scattered around in $\mathbb{R}^6$, T_S contains continuous *cloud-like* regions centered at each $\psi_{tpr} \in T$. Both these factors should make the Soft TPR representation potentially easier to learn and extract information from, compared to the TPR. This is reflected in our empirical results in Section C.6.1, where, compared to the traditional TPR, the Soft TPR demonstrates 1) greater representation learner convergence, 2) superior sample efficiency for downstream tasks, and 3) superior raw downstream performance in the low sample regime.

2. **Quasi-Compositional Structure**: The traditional TPR enforces a strict algebraic definition of compositionality (i.e. $\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$). Even explicitly algebraically-structured domains such as natural language, do not always adhere to this rigid specification of compositional structure (e.g. French liaison consonants, which consist of a weighted sum of fillers bound to a single role [9]). The Soft TPR's continuous relaxation of this constraint allows it to represent structures that only *approximately* satisfy the TPR's strict definition of compositionality. A more relevant example could be the construction of compositional representations of coloured squares where the colour is a weighted combination of fillers blue and red (i.e., varying shades of purple). In this case, even if purple has not been seen by the TPR Autoencoder previously (i.e., there is no quantised filler representing $\xi_F(\text{purple})$), it may be possible for the TPR Autoencoder to produce a Soft TPR corresponding to purple square if there is a suitable ϵ-level relaxation of any of the explicit TPRs (i.e., $\text{red square}, \text{blue square}$) that represents a purple square.

3. **Serial Construction**: Building explicit TPRs requires that the embedded fillers and roles comprising a binding (i.e. $(\xi_F(f_{m(i)}), \xi_R(r_i))$) are first *tokened* before the entire compositional representation, $\psi_{tpr}(x)$ can be produced [19, 23, 28, 34, 38, 51, 52]. This sort of sequential approach, where constituents must be tokened before the compositional representation can be formed, is a key characteristic of the symbolic representation of compositional structure [45]. In contrast, the Soft TPR allows the Encoder to produce any *arbitrary* element of $V_F \otimes V_R$ (in this case $\cong \mathbb{R}^6$), provided that the sufficient closeness requirement holds. Thus, once the Encoder is trained, it is theoretically possible for the Soft TPR Autoencoder to exploit vector space continuity to generate approximately compositional representations by mapping directly from the data to a Soft TPR, without needing to token any representational constituents.

B.2 Alternative Formulations

In contrast to the greedily optimal analytic form of ψ^*_{tpr} that we derive in Equation 5, it is possible to instead derive a globally optimal TPR that the $(D_R \cdot D_F)$-dimensional vector, z, produced by the encoder, E, best approximates with reference to the Frobenius norm metric of $||z - \psi_{tpr}||_F$. We fix the set of fillers, F, and roles, R, as well as the embedding functions, $\xi_F : F \to \mathbb{R}^{D_F}$ and $\xi_R : R \to \mathbb{R}^{D_R}$ to be arbitrary sets and embedding functions respectively. Note that with F, R, ξ_F, and ξ_R fixed, the only degree of freedom in defining the space of possible TPRs is given by defining $\mathcal{M}^*$, the set of role-filler matching functions that define the permissible role-filler binding decompositions of $x \in X$. The globally optimal TPR that z best approximates, ψ^{opt}_{tpr}, is given by:

$$\psi^{opt}_{tpr} := \underset{\psi_{tpr} \in \mathcal{T}}{\arg\min} ||z - \psi_{tpr}||_F, \tag{10}$$

where $\mathcal{T} := \{\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)\}_{m \in \mathcal{M}^*}$ denotes the set of all possible TPRs given the choices of F, R, ξ_F, ξ_R and $\mathcal{M}^*$. Again, we drop the dependency of both ψ_{tpr} and m on x for ease of notation.

If $\mathcal{M}^*$ is a subset of the entire set of matching functions, $\mathcal{M}$, i.e., if there are some constraints to what fillers can be bound to what roles, then, solving for 10 is NP-complete. If $\mathcal{M}^*$ however corresponds to the entire set of possible matching functions, i.e., $\mathcal{M}^* = \mathcal{M}$, then, 10 can be solved by simply using any method that solves the (one-to-many) assignment problem. Noting the poor worst case time complexity of $O(n^3)$ for such solutions, and guided by our intuition that there should be a more explicit dependency between ψ^*_{tpr} and the structure of z, we thus propose the greedily optimal definition of ψ^*_{tpr} in 5, which creates an explicit dependency between the unbound *soft* fillers of z, $\{\tilde{f}_i\}$, and ψ^*_{tpr}.

B.3 Soft TPR Autoencoder

Note that defining m as $m(i) := \arg\min_j ||\tilde{f}_k - \xi_F(f_j)||_2$ in Equation 5 effectively permits *any* filler, f_j, to bind to a role, r_i. This is somewhat inevitable given the weakly supervised framework we are situated in, as without knowledge of the ground-truth FoV type-token bindings, it is impossible to define $\mathcal{M}^*$ as a *subset* of all possible matching functions, $\mathcal{M}$, that produces the ground-truth set of role-filler binding decompositions, i.e. we cannot know $\left\{m \in \mathcal{M} | \{(\xi_F(f_{m(i)}), \xi_R(r_i))\} = \beta(x)\right\}_{x \in X}$. We observe, however, that the size of $\mathcal{M}$ is combinatorially large, i.e., $|\mathcal{M}| = \binom{N_F + N_R - 1}{N_R}$, and contains role-filler binding decompositions that are clearly misaligned with the ground-truth semantics of images, e.g., $\{\text{cube/object colour}, \text{purple/object shape}, \text{green/orientation}, \text{blue/wall colour}, 2\pi/\text{floor colour}\}$ for the image in Figure 2. We thus, decide to design our Soft TPR Autoencoder with 2 orthogonal and separately achieved aims: 1) *representational form*, to ensure the produced representation has the *form* of a Soft TPR, and 2) *representational content*, to ensure that m produces sensible role-filler bindings, and hence, that the Soft TPR, z, which best approximates ψ^*_{TPR} reflects the ground-truth FoV type-token bindings of the image.

B.3.1 Representational Form

We now provide additional details on how our method achieves the representational form property. Recall from Section 4.2, that we penalise the Euclidean distance between the encoder, E's output, z, and the explicit TPR ψ^*_{tpr} that z best approximates with the analytic form of Equation 5. Such a penalisation should ideally encourage the Soft TPR Autoencoder to produce encodings of inputted images that have the form of a Soft TPR, as all of E's outputs are directly penalised to satisfy the sufficient closeness property of $||z - \psi_{tpr}||_2$[5] for the TPR, $\psi_{tpr} = \psi^*_{tpr}$. To recover ψ^*_{tpr}, we use our TPR decoder. As specified in Section 4.2, the TPR decoder consists of 3 main modules, 1) *Unbinding*, where the soft fillers of z, $\{\tilde{f}_i\}$ are unbound from z, 2) *Quantisation*, where the explicit TPR filler embeddings $\{\xi_F(f_{m(i)})\}$ with the closest Euclidean distances to each soft filler of z are obtained, and 3) *TPR Construction*, where the required operations are applied to the embedding vectors of the role-filler bindings produced from previous modules, to construct the explicit TPR, $\sum_i \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$ with the analytic form of 5. We now provide additional details on these 3 modules.

1) Unbinding: While FoV *types* (roles) can be bound to the same FoV *value* (fillers), e.g. $\mathrm{object\ colour}/\mathrm{magenta}, \mathrm{floor\ colour}/\mathrm{magenta}$, each FoV *type* (role) clearly represents an independent visual concept type, and so, it is reasonable to use linearly independent embedding vectors for the FoV types. Thus, within our considered domain of visual representation learning, we can reasonably satisfy the linear independence requirement that ensures recoverability of the embedded representational components $(\xi_F(f_{m(i)}), \xi_R(r_i))$ from the TPR representation. Note that any randomly initialised role embedding matrix $M_{\xi_R} \in \mathbb{R}^{D_R \times N_R}$ is likely (though not guaranteed) to consist of N_R linearly independent role embedding vectors provided $D_R >> N_R$, and furthermore, that we can encourage that the linear independence property to be preserved during all stages of training by adding orthogonality regularisation, $||M^T_{\xi_R} M_{\xi_R} - I_{N_R \times N_R}||_F$. There is one pertinent problem, however, if we simply let the role embedding matrix, M_{ξ_R}, consist of any set of arbitrary, linearly independent role embedding vectors, $\{\xi_R(r_i)\}$. That is, for any choices of N_R, D_R, where these values are large, it is computationally expensive to obtain U, the (left) inverse of M_{ξ_R} required to produce the unbinding vectors, $\{u_i\}$, needed for unbinding. We thus, decide to leverage the properties of (left-invertible) semi-orthogonal matrices, $A \in \mathbb{R}^{d \times n}$, for which $A^T A = I_{n \times n}$. In this case, $U^T = M_{\xi_R}$, and so, the i-th unbinding vector, u_i simply corresponds to the i-the role embedding vector, $\xi_R(r_i)$. Thus, we can simply unbind the soft filler embedding from z 'bound' to role r_i by taking the (tensor) inner product between z and $\xi_R(r_i)$, the i-th column of M_{ξ_R}.

To accomplish this in practice, our unbinding module randomly initialises the role embedding matrix, M_{ξ_R} as a semi-orthogonal matrix, through use of $\mathrm{torch.nn.utils.parameterizations.orthogonal}$, and fixes M_{ξ_R} during all stages of training (i.e., gradient is *not* backpropagated to M_{ξ_R}), to ensure that the semi-orthogonal property is preserved during all stages of training. Fixing M_{ξ_R} may prompt concerns that that role embedding vectors do not capture semantically informative content. However, for the visual representation learning domain we consider, it is intuitive that role embeddings themselves do not need to convey much semantic content, as the majority of an image's semantic content can be conveyed through filler embeddings and the *bindings* of fillers to roles. More concretely, filler embeddings, being learnable, can convey semantic information by being close (in some norm) to one another, e.g., the filler embeddings for $\{\mathrm{green}, \mathrm{magenta}, \mathrm{blue}\}$ may be close in Euclidean distance to one another, and thus convey the semantic information that they correspond to *values* for the same (colour) role. Additionally, as the decoder, D, receives information from all embedded bindings in the form of the TPR, ψ^*_{tpr}, to perform image reconstruction, D should intuitively learn through the reconstruction loss term of Equation 6, the semantics associated with a role. For example, given the embedding for binding $(\mathrm{oblong/object\ \ shape})$, D should learn from the reconstruction loss, the mapping between the randomly initialised role embedding vector for object colour and the semantics of the corresponding FoV type (i.e., that this role changes the *colour* of the *object* in the image).

Now, we proceed to formally prove the aforementioned semi-orthogonality properties.

We first prove that for a (left-invertible) semi-orthogonal matrix, $A \in \mathbb{R}^{d \times n}$, $A^T A = I_{n \times n}$.

[5]We again assume we are working with vector spaces over the reals, and use the vector space isomorphism property of $\mathbb{R}^{D_F} \otimes \mathbb{R}^{D_R} \cong \mathbb{R}^{D_F \cdot D_R}$ to align the TPR framework more seamlessly with the autoencoding framework.

Proof. Let A denote a (left-invertible) semi-orthogonal matrix of dimension $\mathbb{R}^{d\times n}$. Writing A out using its columns, $\{a_1, \ldots, a_n\}$, we have:

$$A^T A = \begin{pmatrix} a_1^T \\ \vdots \\ a_n^T \end{pmatrix} (a_1 \quad \cdots \quad a_n)$$

As the columns $\{a_1, \ldots, a_n\}$ are orthonormal, we have that: $a_i \cdot a_j = \delta_{ij}$, where $\cdot$ denotes the dot product, and

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise,} \end{cases}$$

from which the result clearly follows. □

For completeness, we also prove that u_i corresponds to the i-th role embedding vector, though this follows rather trivially from the properties of semi-orthogonal matrices, and the definition of u_i.

Proof. Let M_{ξ_R} be a (left-invertible) semi-orthogonal matrix. Then, from the properties of semi-orthogonal matrices proved earlier, we have that $M_{\xi_R}^T M_{\xi_R} = I$, and so, $U^T = \left(M_{\xi_R}^T\right)^T = M_{\xi_R}$. Hence, the i-th column of U^T, which corresponds to the unbinding vector, u_i, by definition, is simply $(M_{\xi_R})_{:i} = \xi_R(r_i)$, the i-th role embedding vector. □

2) Quantisation

The quantisation module relies on the VQ-VAE vector quantisation algorithm [11] to learn the filler embedding vectors of the filler embedding matrix, M_{ξ_F}, and to quantise the soft filler embeddings $\{\tilde{f}_i\}$ into the explicit filler embeddings $\{\xi_F(f_i)\}$ with the closest Euclidean distances. Briefly speaking, to perform vector quantisation, the VQ-VAE algorithm simply quantises each soft filler $\tilde{f}_i$ into the embedding vector from M_{ξ_F} with the smallest Euclidean distance [11]. This clearly corresponds with the definition of m in Equation 5 (i.e. for each soft filler embedding, $\tilde{f}_i$, m matches an explicit filler embedding with the smallest Euclidean distance to $\tilde{f}_i$ to produce ψ^*_{tpr}). As this quantisation operation corresponds to an argmax and is thus non-differentiable, the VQ-VAE algorithm uses a simple L_2 loss, the *codebook loss*, to move the embedding vectors $\xi_R(r_i)$ towards the soft filler embeddings, as captured by the first term of the VQ-VAE quantisation loss term in Equation 6. To prevent the embedding space from growing arbitrarily, a *commitment loss*, corresponding to the final term of the VQ-VAE quantisation loss in 6 is added, to ensure the encoder commits to an embedding.

We refer interested readers to [11] for more details on how the quantisation algorithm works.

3) TPR Construction

The TPR construction module contains no learnable parameters, and simply deterministically applies the operations required to produce a TPR from the quantised filler embeddings, $\{\xi_F(f_{m(i)})\}$, with the corresponding role embeddings, $\{\xi_R(r_i)\}$, from the role embedding matrix, M_{ξ_R}.

B.3.2 Representational Content

In this section, we provide an explicit form for the 'swapped' TPRs, $\psi^s_{tpr}(x)$ and $\psi^s_{tpr}(x')$, used in our weakly supervised reconstruction loss corresponding to the second term of 7.

Given a pair of images, (x, x') where x and x' share the same role-filler bindings for all roles but r_i, and the identity of r_i, but not any of the role-filler bindings are known, we construct the 'swapped' TPRs for x and x' as follows:

$$\psi^s_{tpr}(x) := \sum_{j\neq i} \xi_F(f_{m(j)}) \otimes \xi_R(r_j) + \xi_F(f_{m'(i)}) \otimes \xi_R(r_i) \tag{11}$$

$$\psi^s_{tpr}(x') := \sum_{j\neq i} \xi_F(f_{m'(j)}) \otimes \xi_R(r_j) + \xi_F(f_{m(i)}) \otimes \xi_R(r_i), \tag{12}$$

where m and m' denote the matching functions for x and x' respectively. That is, we construct 'swapped' TPRs by simply swapping the quantised filler embedding associated with role r_i in constructing new TPR representations for x and x'. Note that in contrast to a similar operation that might be applied to *scalar-tokened* or *vector-tokened* compositional representations, swapping the quantised filler between the representations for x and x' in this case produces a *global* (not local) effect on the resulting representation.

B.3.3 Model Architecture

We now provide concrete details of our model architecture. Our model consists of a standard Conv-based encoder, E, (Table 8) our TPR decoder, (Table 9) and a standard Conv-transpose based decoder, D (Table 10). The only learnable component of the TPR decoder corresponds to the filler embedding matrix, M_{ξ_F}, and so, our Soft TPR Autoencoder only adds an additional $D_F \cdot N_F$ parameters to a standard (variational) autoencoding framework consisting of the encoder and decoder, where D_F and N_F correspond to the dimensionality of the filler embedding space, and number of filler embedding vectors respectively.

For all our experiments, we use the same general architecture for the encoder, E, the TPR decoder, and the decoder, D.

Table 8: Encoder (E) architecture.

Encoder
Input $64 \times 64 \times c$
Conv 32, 4×4, stride $= 2$, padding $= 1$
BatchNorm 32
ReLU 32
Conv 64, 4×4, stride $= 2$, padding $= 1$
BatchNorm 64
ReLU 64
Conv 256, 4×4, stride $= 2$, padding $= 1$
BatchNorm 256
ReLU 256
Conv 512, 4×4, stride $= 2$, padding $= 1$
BatchNorm 512
ReLU 512
Flatten $512 \times 4 \times 4$
Linear 1024
ReLU 1024
Linear 512
ReLU 512
Linear 512
ReLU 512
Linear $D_F \cdot D_R$

Table 9: TPR Decoder architecture.

TPR Decoder
Role embedding matrix (*fixed*) $D_R \times N_R$
Filler embedding matrix $D_F \times N_F$

Table 10: Decoder (D) architecture.

Decoder
Input $D_F \cdot D_R$
ConvTranspose 512, 4×4, stride $= 2$, padding $= 1$
BatchNorm 512
ReLU 512
ConvTranspose 256, 4×4, stride $= 2$, padding $= 1$
BatchNorm 256
ReLU 256
ConvTranspose 64, 4×4, stride $= 2$, padding $= 1$
BatchNorm 64
ReLU 64
ConvTranspose 32, 4×4, stride $= 2$, padding $= 1$
BatchNorm 32
ReLU 32
ConvTranspose 3, 4×4, stride $= 2$, padding $= 1$

B.4 Model Hyperparameters and Hyperparameter Tuning

The tunable hyperparameters of the Soft TPR Autoencoder are the following following: 1) architectural hyperparameters N_R, N_F, D_R, and D_F corresponding to the number of role (respectively filler) embedding vectors and the dimensionalities of their respective embedding spaces, and 2) loss function hyperparameters β (Equation 6), λ_1 (Equation 7) and λ_2 (Equation 7).

In line with VQ-VAE [11], we set β, the coefficient for the VQ-VAE *commitment loss* to 0.5. As N_R corresponds to the number of FoV *types*, to ensure fair comparisons with scalar-tokened generative baselines, and COMET, which all assume that the number of FoVs equals 10, (VCT assumes 20), we fix N_R to be 10. We tune remaining hyperparameters by running hyperparameter optimisation using the open-source hyperparameter sweep framework of Weights and Biases (WandB). We set the search method as Bayesian search, and the optimisation criterion to be the fully unsupervised MSE reconstruction loss corresponding to the second term in Equation 6. During hyperparameter optimisation, we train all models for between 50,000-100,000 iterations. The obtained hyperparameter values for the Soft TPR Autoencoder are listed in Table 11. See Section C.6.2 for ablation experiments demonstrating our model's robustness to hyperparameter configurations.

Table 11: Hyperparameter values.

Hyperparameter	Cars3D	Shapes3D	MPI3D
	Architectural hyperparameters		
D_R	12	16	12
N_R *(fixed)*	10	10	10
D_F	128	32	32
N_F	106	57	50
	Loss function hyperparameters		
λ_1	0.00024	0.00091	0.0000
λ_2	0.02200	0.00228	1.16050
β *(fixed)*	0.5	0.5	0.5

In practice, we find that the Soft TPR penalty of Equation 6 is negligible, most likely because the *codebook loss* of VQ-VAE, which pushes the quantised fillers $\{\xi_F(f_{m(i)})\}$ and the soft filler embeddings $\{\tilde{f}_i\}$ of z together, is already sufficient to push z to ψ^*_{tpr}. To prevent redundantly calculating this term, we remove it from the overall loss in the implementation of our model.

Our model is implemented in Pytorch [27] and trained using the Adam optimiser on the loss corresponding to 7. We use a learning rate of $1\mathrm{e}{-4}$, and the default setting of $(\beta_1, \beta_2) = (0.9, 0.999)$ across all instances of model training.

C Results

In this section, we provide additional details about our experiments, and present additional results.

C.1 Datasets

For disentanglement, we consider the standard disentanglement datasets: Cars3D [4], Shapes3D [24], and MPI3D [21], where for MPI3D, we consider the 'real' variant of the dataset, (i.e., not the 'realistic', 'toy', or 'complex' variants of the dataset). These datasets contain 3, 6, and 7 ground-truth FoVs respectively, and are procedurally generated by taking the Cartesian product of all possible FoV values for each FoV type. We provide metadata in Tables 12, 13 and 14 and denote all continuous-valued FoV types by the symbol ‡. As the colours in the Shapes3D dataset correspond to 10 linearly spaced values with a natural ordering (e.g. red, with value 0, is perceptually more similar to orange, with value 1, compared to aqua, with value 5), we treat all FoV colour types in the Shapes3D dataset as continuous-valued variables, in line with [42].

Table 12: Cars3D dataset

FoV types	FoV values
Car type	199 types
Rotation‡	24 angles
Camera elevation‡	4 types

Table 13: Shapes3D dataset

FoV types	FoV values
Floor colour‡	10 colours
Wall colour‡	10 colours
Object colour‡	10 colours
Object size‡	8 sizes
Object type	4 shapes
Azimuth‡	15 angles

Table 14: MPI3D dataset

FoV types	FoV values
Object colour	6 colours
Object shape	6 shapes
Object size‡	2 sizes
Camera height	3 heights
Background colour	3 colours
Horizontal axis‡	40 values
Vertical axis‡	40 values

For the downstream task of regression, we simply use the above disentanglement datasets, regressing on each dataset's continuous-valued FoVs, as indicated by the ‡ symbol in Tables 12, 13 and 14. For the downstream task of abstract visual reasoning [30], we use the dataset from [30]. Each sample from this dataset consists of a Raven's Progressive Matrix (RPM) style question (samples of the dataset can be found in [30]). Each RPM-style question has a set of 8 *context* panels, and 6 *answer* panels, where each panel corresponds to an image from the Shapes3D dataset. For the model to select the correct answer out of a selection of 6 possible answer panels, it must correctly infer the abstract visual relation that each row of the context panels share.

C.2 Baseline Implementations and Experimental Settings

C.2.1 Experiment Compute Resources

For the generative weakly supervised, scalar-valued baselines (i.e., AdaGVAE-k, GVAE, MLVAE, SlowVAE, the Shu model), and our model, we perform model training on a single Nvidia RTX4090 GPU. We also perform all associated experiments for these models (i.e., downstream model training, downstream model evaluation, disentanglement evaluation, etc.) on this single Nvidia RTX4090 GPU. Our model takes approximately 1.5 hours to fully train (i.e., run 200,000 iterations) on the Cars3D and Shapes dataset, and approximately 4.0 hours to fully train on the MPI3D dataset.

For the vector-valued baselines of COMET and VCT, we perform model training, and all associated experiments on a single Nvidia V100 GPU.

C.2.2 Disentanglement Models

For Ada-GVAE, GVAE, MLVAE and SlowVAE, use the Pytorch-based, open-source implementation of [42], which was verified by authors to reproduce official reported results for each model. For COMET, and VCT, we use official open-source implementations published by the authors of the corresponding models [36, 47]. For the GAN-based model of Shu, we convert the official Tensorflow-based implementation of [35] to Pytorch and verify that our implementation reproduces official results. For all baseline models, we use suggested hyperparameters where dataset-specific hyperparameters are given, otherwise, we perform hyperparameter tuning using an identical WandB hyperparameter sweep setup as our model.

As mentioned in the first paragraph of Section 5, for the weakly supervised baselines of Ada-GVAE, GVAE, MLVAE, SlowVAE and Shu, all models with exception to Ada-GVAE assume access to I, the FoV that differ between each pair in an observed sample (x, x'), and thus have identical levels of weak supervision as our model. Ada-GVAE, however, does not assume access to I, so we make the following modification to make Ada-GVAE more comparable with our model. Ada-GVAE adaptively estimates I using a method that relies on the estimation of $k := |I|$, i.e., the *number* of FoVs that have changed in each observed sample. Thus, we simply amend Ada-GVAE by providing it access to the ground-truth value for k (note that if we instead provide Ada-GVAE with knowledge of the FoVs comprising I itself, it becomes identical to GVAE). We empirically verify that our modification, which we denote by Ada-GVAE-k, produces superior or in worst case, comparable results to the original Ada-GVAE model. We train all models, with exception to SlowVAE, which assumes all FoV values in observed pairs (x, x') change, with $k = 1$ (i.e., only 1 FoV changes between x and x'). This ensures all models are trained in an identical setting as our framework.

For the selected hyperparameter configuration associated with each model, including our own, we obtain 5 trained models using 5 random seeds, and collate results over these 5 seeds. All models are trained for 200,000 iterations on all datasets.

C.2.3 Downstream Models

For the task of FoV regression, in line with [42], we use a simple, generic MLP model, with the architecture listed in Table 15. For each disentanglement dataset, the MLP regression model receives representations of images produced by a representation learner (i.e., a disentanglement model), and is trained to predict the corresponding ground-truth FoV values in a supervised fashion. We evaluate regression performance by computing the R^2 score on a held out, randomly selected test set of 1,000 samples.

For each of the 5 instances of a representation learning model produced by the 5 random seeds, we obtain 2 MLPs, resulting in a total of 10 MLP models for each representation learner. We obtain each MLP by uniformly sampling the number of output nodes in the first, second, and fifth layers of the MLP, as indicated in 15. All reported results are averaged over these 10 MLP models. Note that all MLP regression models have *no* a priori knowledge of the representational form (i.e. scalar-tokened symbolic compositional representation, vector-tokened symbolic compositional representation, fully continuous compositional representation) that they will receive during training as we do not provide any inductive biases to the MLP models or optimise them in any representation-learner specific way (apart from through using the supervised, MSE-based training loss).

For each MLP model, we use the Adam optimiser on the MSE loss between predicted and ground-truth FoV values, with a learning rate of $1\mathrm{e}{-4}$, and the default setting of $(\beta_1, \beta_2) = (0.9, 0.999)$.

Table 15: MLP architecture

MLP
Linear d_1, $d_1 \in [256, 512]$
ReLU
Linear d_2, $d_2 \in [256, 512]$
ReLU
Linear $\dim(z)$
ReLU
Linear $\dim(z)$
ReLU
Linear d_3, $d_3 \in [128, 256]$
ReLU
Linear k

For the abstract visual reasoning task, in line with [30], we use the Wild Relation Network (WReN) model [18] on representations obtained by all representation learners to predict the ground-truth answer for each RPM matrix. For each of the 5 instances of a model produced by the 5 random seeds, we randomly sample 2 possible configurations of the WReN model, producing 10 WReN models for each representation learning model. In line with [30], for the edge MLP, g, in the WReN model, we uniformly sample either 256 or 512 hidden units. Similarly, we uniformly sample either 128 or 256 hidden units for the graph MLP, f. We however, fix the number of hidden layers in g to 2, and f in 1,

representing the smallest possible number of hidden layers, to constrain the capacity of the WReN model. We refer readers to [18] for further details on the WReN architecture. All WReN models are trained using Adam optimisation on the BCE loss between the predicted logits and ground-truth labels, with a learning rate of $1\text{e}{-5}$ and the default setting of $(\beta_1, \beta_2) = (0.9, 0.999)$.

C.2.4 Experimental Controls

Recall that our main hypothesis is that neural networks can both learn explicit compositional structure, and leverage it more easily when that structure is instantiated in a fully continuous way, e.g., when embodied by our Soft TPRs. To ensure that the experimental results indeed provide empiricial support for this hypothesis, we apply a series of controls to rule out the contribution of any confounding variables to the empirical results. We detail these controls below.

Disentanglement Controls

To ensure that any performance boosts on the disentanglement metrics (shown in Tables 16 and 17) are not attributable to the slight increase in learnable parameters of our model (note that our model only adds one learnable component, i.e., the filler embedding matrix, M_{ξ_F} on top of a standard (variational) autoencoding framework, with this corresponding to 13,568, 1,824, and 1,600 additional parameters for the Cars3D, Shapes3D, and MPI3D datasets respectively), we perform a control to fix the number of learnable parameters in baseline models with fewer learnable parameters compared to our model. The baseline models with fewer learnable parameters than our model are SlowVAE, Ada-GVAE-k, GVAE, ML-VAE, and the Shu model. VCT and COMET are substantially more parameter hungry (10+ million) than our model, so we do not perform the controls on these models. We increase the number of parameters by increasing the number of filters in the convolution (and transpose-convolution) layers of the generative baseline models, and/or increasing the number of convolution (and transpose-convolution) layers until the modified model has at least as many parameters as our model, repeating this process for each of the disentanglement datasets. We denote these parameter-controlled models in Tables 16 and 17 with the symbol *. In line with [40] and as shown in Tables 16 and 17, we do not observe any increase in disentanglement performance conferred by increasing the number of learnable parameters in these generative, scalar-tokened baselines, so we apply representation learning convergence experiments, and downstream model experiments using the original models, and not their parameter-controlled variants.

Downstream Task Controls

To ensure that any performance boosts on the downstream tasks of FoV regression and abstract visual reasoning are not attributable to our representation's increased dimensionality, we post-process representations produced by *all* models (including models producing higher-dimensional representations) to match the dimensionality of our representation. Note that all generative, weakly supervised models producing scalar-tokened representations produce representations with a dimension of 10 for all datasets, which contrasts with the representational dimensions of 1536 (Cars3D), 512 (Shapes3D), and 320 (MPI3D) of our Soft TPRs, highlighting the necessity of such a control.

To perform this control, we apply separate methods for the scalar-tokened and vector-tokened models. For scalar-tokened models, we multiply each dimension k of the latent vector, l_k with an random embedding vector $e_k \in \mathbb{R}^d$ from an fixed, randomly initialised, semi-orthogonal embedding matrix $E \in \mathbb{R}^{d \times 10}$, and subsequently concatenate all multiplied random embedding vectors together to form the dimensionality-controlled representation, $(l_1 e_k, \ldots, l_{10} e_{10}) \in \mathbb{R}^{10d}$. d is a dataset specific integer that ensures that the size of the dimensionality-controlled representation is at least as small as the dimensionality of the Soft TPR representation for the given dataset, i.e., $d := \lceil \dim(z) \rceil / 10$, where z is the Soft TPR representation produced by a given dataset. Note that we choose a semi-orthogonal embedding matrix with the intuition that this ensures the maximal distinguishability of each continguous subset of dimensions corresponding to $l_i e_i$.

For COMET and VCT, which both produce vector-tokened representations, we consider alternative methods of performing dimensionality-control. COMET's representations have a dimensionality of 640 for all datasets, and so, the model produces lower-dimensional representations than ours for Shapes3D and Cars3D, but not MPI3D. VCT, on the other hand, produces representations with dimension 5120, and so, produces higher-dimensional representations than ours for all datasets. For any vector-tokened representation with *higher* dimensionality than our representation, we apply PCA-based postprocessing to the model representation, reducing the dimensionality of each vector-tokened

value in the representation to the required dimensionality, d, where $d := \lceil \dim(z) \rceil / \dim(z_{baseline})$, before concatenating all PCA-reduced vectors together to produce the modified representation. For any vector-tokened representation produced by COMET with lower dimensionality than ours, we apply a simple matrix multiplication between the representation, and a randomly initialised matrix of required dimensionality to embed the representation into the same-dimensional space as ours.

We denote the results produced by all such postprocessing by † in relevant tables, and indicate such postprocessing in the captions of relevant plots.

C.3 Disentanglement

In reporting the disentanglement metric results for baseline models, we use published results where applicable, i.e., we use the results published in [47] for VCT and COMET and the published results for SlowVAE [39]. For GVAE, MLVAE, and Ada-GVAE-K, we evaluate disentanglement using the Pytorch-based implementation of disentanglement metrics, [42], which corresponds to a Pytorch-based implementation of the official disentanglement lib of [25]. We also use this implementation to evaluate the disentanglement of representations produced by all models.

C.3.1 Disentanglement Metrics

In line with [47], we consider the following 4 disentanglement metrics: the FactorVAE score [24], the DCI Disentanglement score [16] (we refer to DCI Disentanglement as DCI), the BetaVAE score [10], and the MIG score [15]. We provide a brief overview of all 4 disentanglement metrics, and refer interested readers to the papers these metrics are introduced in for further details, as well as the Appendix of [25] for more details on how the metrics are implemented in [42] and [25].

FactorVAE Metric: A randomly selected FoV of the dataset is fixed, and a mini-batch of observations is subsequently randomly sampled. The representation learner produces representations for the samples. Disentanglement is quantified using the accuracy of a majority vote classifier that predicts the index of the ground-truth fixed FoV based on the index of the representation vector with the smallest variance.

DCI Metric: A Gradient Boosted Tree (GBT) is trained to predict the ground-truth FoV values from representations produced by a representation learner. The predictive importance of the dimensions of a representation is obtained using the model's feature importances. For each sample, a score is computed that corresponds to one minus the entropy of the probability that a dimension of the learned representation is useful for FoV prediction, weighted by the relative entropy of the corresponding dimension. An average of these scores over the mini-batch of samples is taken to produce the final score.

BetaVAE Metric: This metric quantifies disentanglement by predicting the index of a fixed FoV from representations produced by a representation learner, similar to the FactorVAE metric. In contrast to the FactorVAE metric, however, the BetaVAE metric uses a linear classifier on difference vectors to predict the index of the fixed FoV. Each difference vector is produced by taking the difference between representations produced for a pair of samples (x, x') with one underlying fixed FoV.

MIG Metric: For each FoV, the MIG metric computes the mutual information between each dimension in the representation, and the corresponding FoV. The score is obtained by computing the average, normalised difference between the highest and second highest mutual information of each FoV with the dimensions of the representation.

C.3.2 Evaluating the Disentanglement of Soft TPRs

Since disentanglement metrics are typically computed under the assumption that the representation corresponds to a concatenation of *scalar-valued* FoV tokens, we now detail how we compute the above disentanglement metrics on the Soft TPR, a *continuous* compositional representation. Similar amendments have been made in COMET and VCT, so that the disentanglement of their representations, corresponding to a concatenation of *vector-valued* FoV tokens, can be computed.

FactorVAE metric: To compute the FactorVAE score of our Soft TPR, we produce a N_R-dimensional vector, v, for each Soft TPR, where N_R corresponds to the number of roles, i.e. the FoV *types*. We produce v by simply populating each dimension, v_i, with the index of the quantised filler that role r_i

is bound to, i.e. we set v_i to $m(i)$. We use the resulting v's to compute the variances required by the FactorVAE metric, noting that if the FoV type corresponding to role i is fixed, and this is reflected in our representation, dimension i of the v vectors produced in a mini-batch should clearly have the smallest variance, as all fillers, $f_{m(i)}$, that role r_i binds to, will have the same identity across the mini-batch.

DCI metric: As the DCI relies on computing the ground-truth FoV values from N_R-dimensional representations produced by the representation learner, we again, follow a similar procedure as the above, in converting our $(D_F \cdot D_R)$-dimensional Soft TPR representation to a N_R-dimensional representation. That is, we simply consider a N_R-dimensional vector, v, where each dimension, v_i, is populated by the index of the quantised filler, $m(i)$ that role r_i is bound to, and use this vector to compute the corresponding DCI result.

BetaVAE metric: For the BetaVAE metric, as each sample used to train the linear classifier consists of a N_R-dimensional difference vector obtained by computing the difference between the N_R-dimensional scalar-tokened compositional representations, we obtain the following difference vector, d, for our Soft TPR representations. For each role $i \in \{1, \ldots, N_R\}$, we obtain the corresponding quantised filler embeddings for each sample x, x' in the pair, i.e., we obtain $\xi_F(f_{m(i)}), \xi_F(f_{m'(i)})$. For each pair of quantised filler embeddings, we obtain a scalar distance measure corresponding to the cosine similarity between the the pair. The i-th dimension of d is populated using this value. We use difference vectors obtained in this way to compute the BetaVAE metric.

MIG metric: For the MIG metric, which relies on a discretisation of the values in each dimension i of the N_R-dimensional scalar-tokened compositional representations to compute the discrete mutual information, we apply the same postprocessing technique as in the FactorVAE, and DCI metric, to evaluate MIG on our Soft TPRs. That is, we produce the same N_R-dimensional vector, v, noting that this choice of postprocessing also performs the discretisation required by the MIG computation.

C.3.3 Full Results

We now present unabridged results for all considered disentanglement metrics, denoting the learnable parameter control modification of each relevant baseline with the symbol *. As can be seen in Tables 16 and 17, our Soft TPR representations are explicitly more compositional (as quantified by the disentanglement metric scores) compared to all considered baselines, with especially notable performance increases on the more challenging datasets of Cars3D and MPI3D.

Table 16: FactorVAE and DCI scores

Models	Cars3D		Shapes3D		MPI3D	
	FactorVAE score	DCI	FactorVAE score	DCI	FactorVAE score	DCI
Symbolic scalar-tokened compositional representations						
Slow-VAE	0.902 ± 0.035	0.509 ± 0.027	0.950 ± 0.032	0.850 ± 0.047	0.455 ± 0.083	0.355 ± 0.027
Slow-VAE*	0.872 ± 0.038	0.518 ± 0.039	0.961 ± 0.028	0.867 ± 0.028	0.421 ± 0.091	0.317 ± 0.039
Ada-GVAE-k	0.947 ± 0.064	*0.664 ± 0.167*	*0.973 ± 0.006*	**0.963 ± 0.077**	0.496 ± 0.095	0.343 ± 0.040
Ada-GVAE-k*	0.931 ± 0.051	0.641 ± 0.179	0.932 ± 0.012	0.923 ± 0.108	0.451 ± 0.129	0.319 ± 0.056
GVAE	0.877 ± 0.081	0.262 ± 0.095	0.921 ± 0.075	0.842 ± 0.040	0.378 ± 0.024	0.245 ± 0.074
GVAE*	0.841 ± 0.123	0.219 ± 0.012	0.881 ± 0.129	0.810 ± 0.102	0.341 ± 0.067	0.216 ± 0.109
ML-VAE	0.870 ± 0.052	0.216 ± 0.063	0.835 ± 0.111	0.739 ± 0.115	0.390 ± 0.026	0.251 ± 0.029
ML-VAE*	0.881 ± 0.041	0.220 ± 0.051	0.823 ± 0.091	0.714 ± 0.091	0.401 ± 0.019	0.240 ± 0.043
Shu	0.573 ± 0.062	0.032 ± 0.014	0.265 ± 0.043	0.017 ± 0.006	0.287 ± 0.034	0.033 ± 0.008
Shu*	0.551 ± 0.062	0.019 ± 0.021	0.297 ± 0.031	0.020 ± 0.006	0.219 ± 0.057	0.029 ± 0.010
Symbolic vector-tokened compositional representations						
VCT	*0.966 ± 0.029*	0.382 ± 0.080	0.957 ± 0.043	0.884 ± 0.013	*0.689 ± 0.035*	*0.475 ± 0.005*
COMET	0.339 ± 0.008	0.024 ± 0.026	0.168 ± 0.005	0.002 ± 0.000	0.145 ± 0.024	0.005 ± 0.001
Fully continuous compositional representations						
Ours	**0.999 ± 0.001**	**0.863 ± 0.027**	**0.984 ± 0.012**	*0.926 ± 0.028*	**0.949 ± 0.032**	**0.828 ± 0.015**

C.4 Representation Learning Convergence

We additionally examine representation learning convergence by evaluating the representations produced at 100, 1,000, 10,000, 100,000, and 200,000 iterations of model training, where the latter stage of 200,000 iterations corresponds to fully trained models. To quantify representation learning

Table 17: BetaVAE and MIG scores

Models	Cars3D		Shapes3D		MPI3D	
	BetaVAE score	MIG	BetaVAE score	MIG	BetaVAE score	MIG
Symbolic scalar-tokened compositional representations						
Slow-VAE	**1.000 ± 0.000** (=)	0.104 ± 0.018	**1.000 ±0.000** (=)	*0.615 ± 0.045*	0.666 ± 0.069	*0.329 ± 0.026*
Slow-VAE*	**1.000 ± 0.000** (=)	0.071 ± 0.013	**1.000 ±0.000** (=)	**0.655 ± 0.067**	0.629 ± 0.079	0.287 ± 0.045
Ada-GVAE-k	**1.000 ± 0.000** (=)	**0.395 ± 0.095**	**1.000 ± 0.000** (=)	0.556 ± 0.064	0.750 ± 0.053	0.213 ± 0.064
Ada-GVAE-k*	**1.000 ± 0.000** (=)	0.321 ± 0.102	**1.000 ± 0.000** (=)	0.498 ± 0.073	0.783 ± 0.061	0.241 ± 0.079
GVAE	**1.000 ± 0.000** (=)	0.096 ± 0.036	0.998 ± 0.004	0.251 ±0.072	0.704 ± 0.072	0.145 ± 0.074
GVAE*	**1.000 ± 0.000** (=)	0.100 ± 0.052	**1.000 ± 0.000**	0.203 ±0.089	0.681 ± 0.061	0.109 ± 0.087
MLVAE	**1.000 ± 0.000** (=)	0.088 ± 0.020	0.976 ± 0.038	0.354 ± 0.165	0.703 ± 0.039	0.142 ± 0.062
MLVAE*	**1.000 ± 0.000** (=)	0.050 ± 0.058	0.921 ± 0.052	0.298 ± 0.091	0.689 ± 0.041	0.096 ± 0.071
Shu	0.912 ± 0.022	0.025 ± 0.012	0.498 ± 0.064	0.005 ± 0.003	0.353 ± 0.040	0.013 ± 0.007
Shu*	0.923 ± 0.031	0.015 ± 0.009	0.512 ± 0.071	0.006 ± 0.002	0.327 ± 0.059	0.009 ± 0.011
Symbolic vector-tokened compositional representations						
VCT	**1.000 ± 0.000** (=)	0.117 ± 0.045	0.999 ± 0.0004	0.525 ± 0.028	*0.844 ± 0.038*	0.227 ± 0.048
COMET	0.343 ± 0.006	0.000 ± 0.000	0.166 ± 0.004	0.0002 ± 0.000	0.144 ± 0.005	0.000 ± 0.0001
Fully continuous compositional representations						
Ours	**1.000 ± 0.000** (=)	*0.348 ± 0.0124*	**1.000 ± 0.000** (=)	0.471 ± 0.088	**1.000 ± 0.000**	**0.620 ± 0.067**

convergence, we evaluate both 1) the explicit compositionality of representations produced at these stages of training (as quantified by disentanglement metric performance), and 2) the usefulness of these representations for the downstream tasks of FoV regression and abstract visual reasoning.

As mentioned in Section 5.1, the representation learning convergence of our model as measured by disentanglement performance is comparable with baselines, however, our model consistently converges faster than baselines in producing representations that can be effectively leveraged for both downstream tasks, as indicated by higher downstream model performance across the majority of representation learner training stages. We now present the full suite of unabridged results, first presenting disentanglement results, and subsequently downstream results.

In all line plots, we plot the mean, and indicate the standard deviation by the shaded regions. We use the same legend for all plots, where 0 (grey) denotes SlowVAE, 1 (orange) denotes AdaGVAE-k, 2 (green) denotes GVAE, 3 (red) denotes MLVAE, 4 (purple) denotes Shu, 5 (pink) denotes VCT, 6 (brown) denotes COMET, and 7 (blue) denotes our model, Soft TPR Autoencoder.

C.4.1 Disentanglement

We first present line plots of representation learner convergence for each of the four considered disentanglement metrics (i.e., the FactorVAE, DCI, BetaVAE and MIG scores) for all three disentanglement datasets (Cars3D, Shapes3D, MPI3D). A series of tables containing the values associated with these line plots is presented following the plots. As the disentanglement results produced by our learnable parameter controls for the models of Ada-GVAE-k, GVAE, ML-VAE and Shu, do not achieve superior disentanglement results compared to the original models, we only present disentanglement convergence results for the original variants of all baseline models.

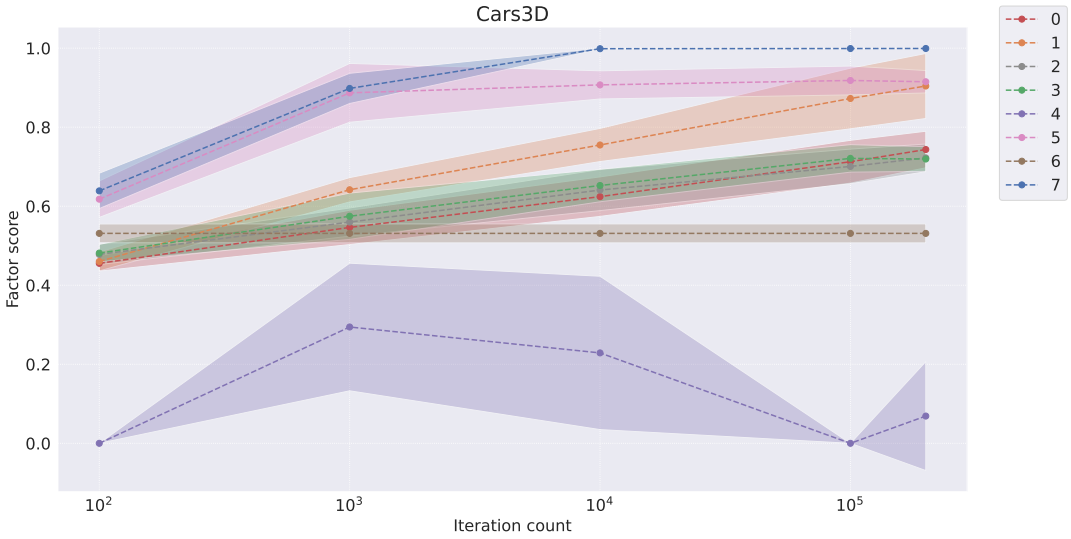

Figure 3: Factor score convergence on the Cars3D dataset

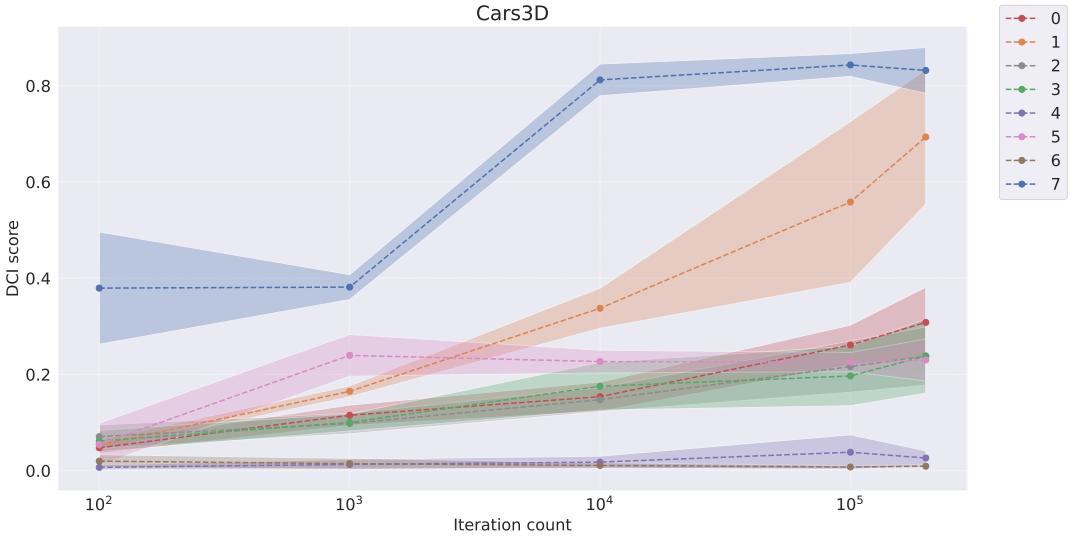

Figure 4: DCI score convergence on the Cars3D dataset

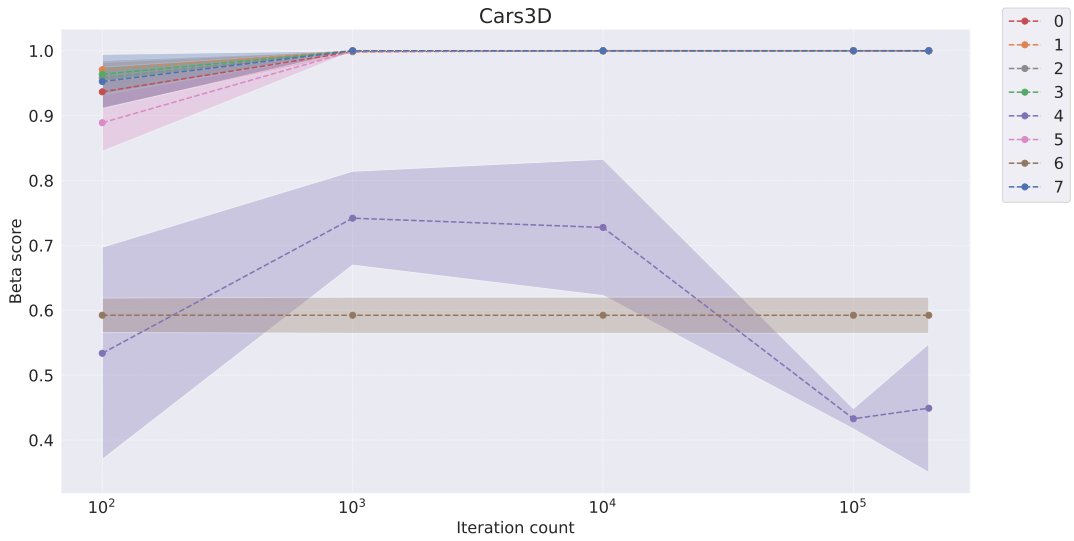

Figure 5: BetaVAE score convergence on the Cars3D dataset

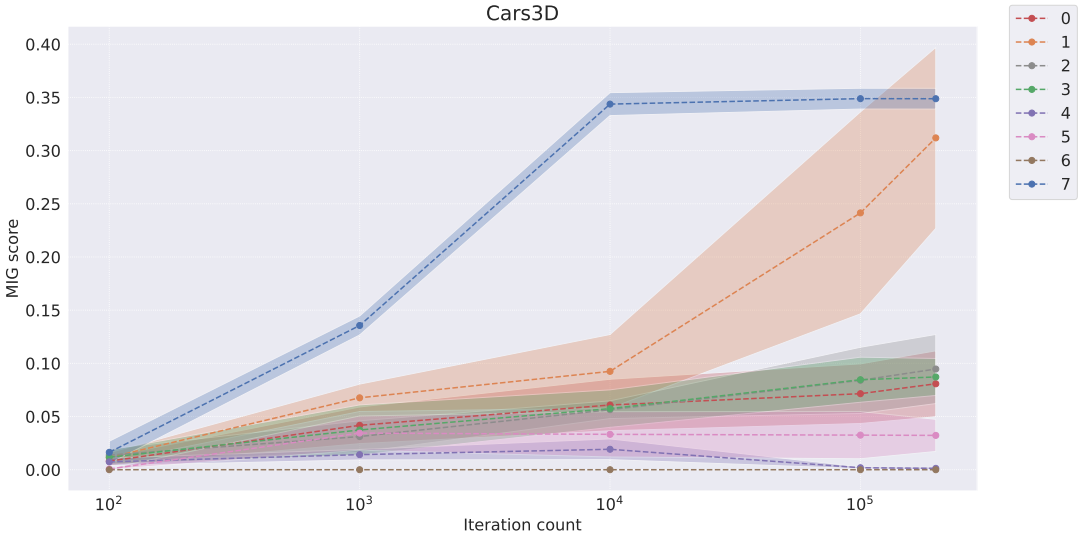

Figure 6: MIG score convergence on the Cars3D dataset

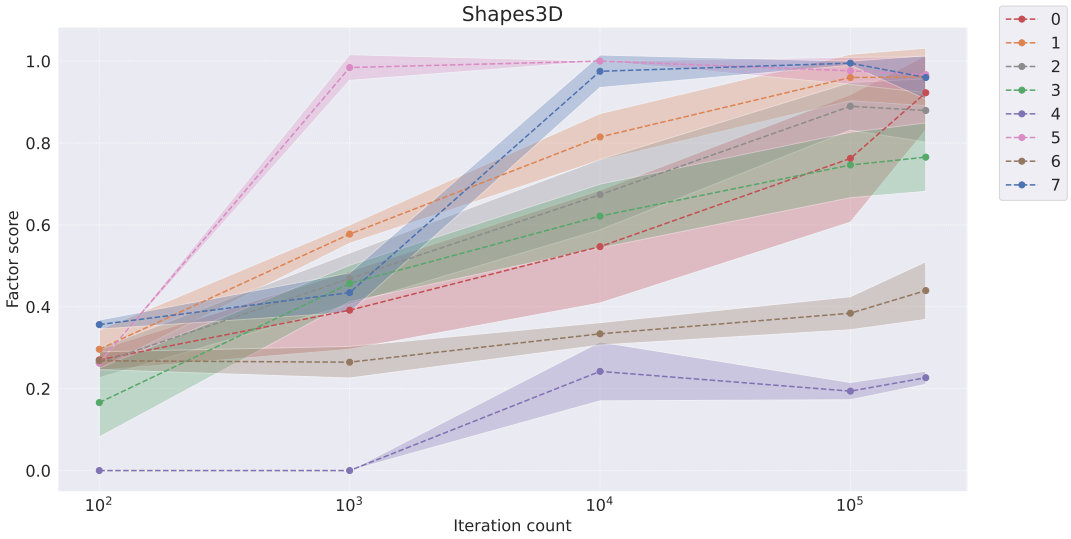

Figure 7: Factor score convergence on the Shapes3D dataset

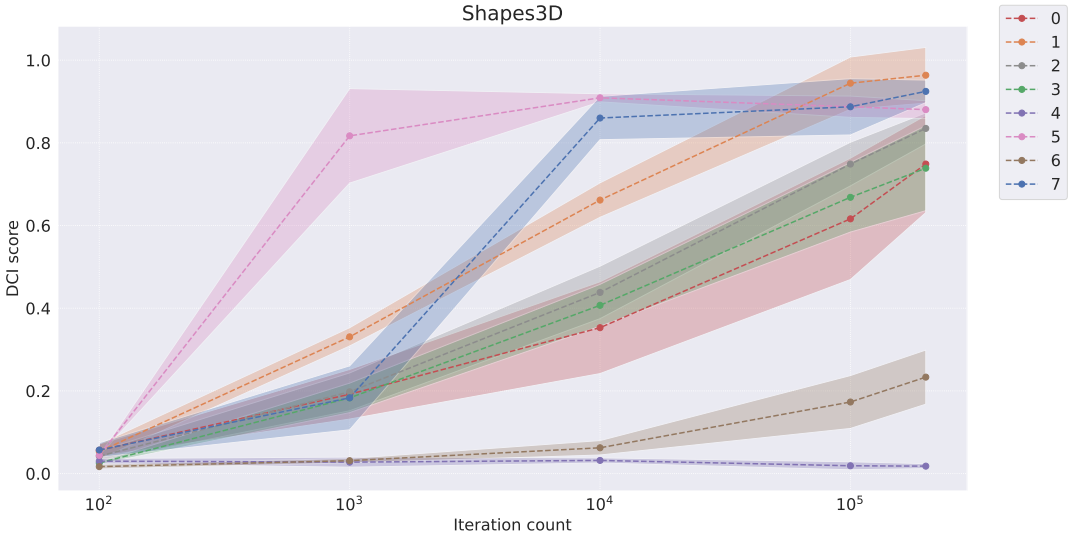

Figure 8: DCI score convergence on the Shapes3D dataset

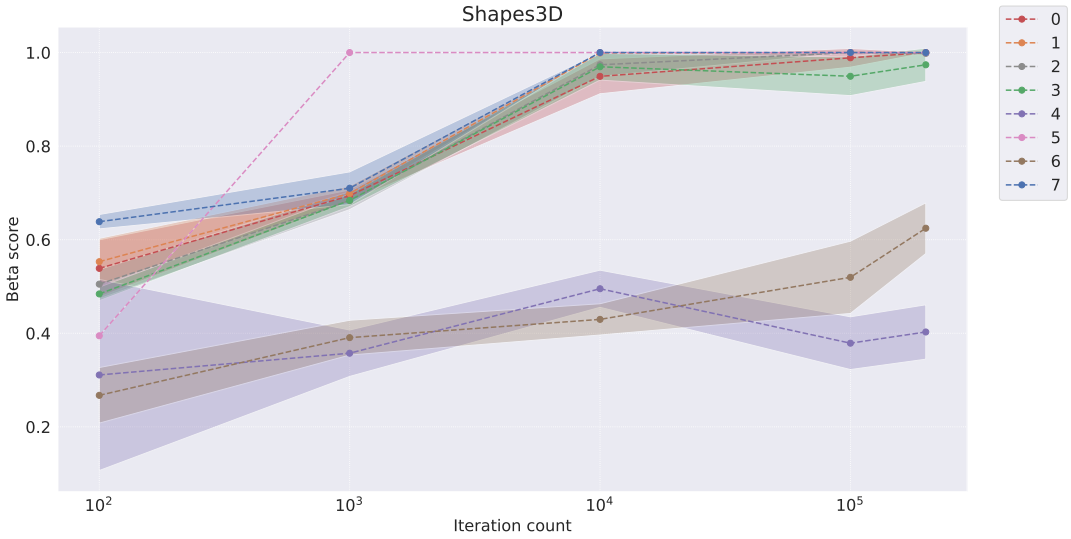

Figure 9: BetaVAE score convergence on the Shapes3D dataset

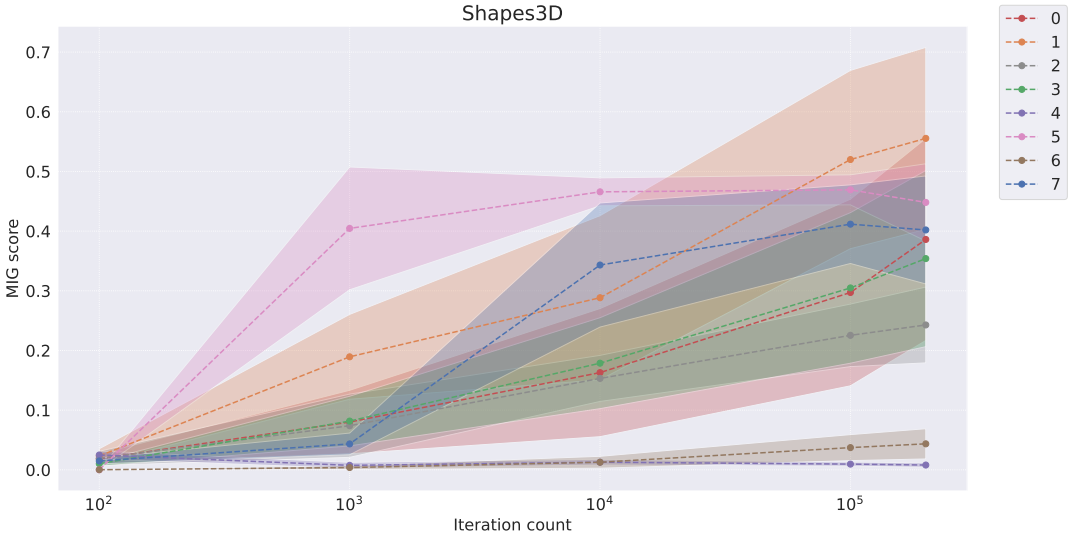

Figure 10: MIG score convergence on the Shapes3D dataset

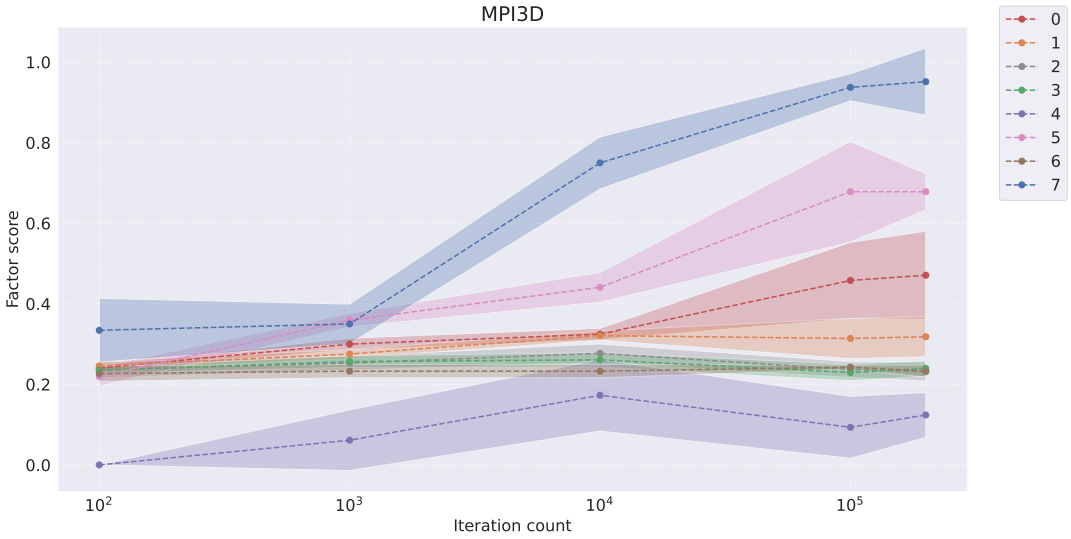

Figure 11: Factor score convergence on the MPI3D dataset

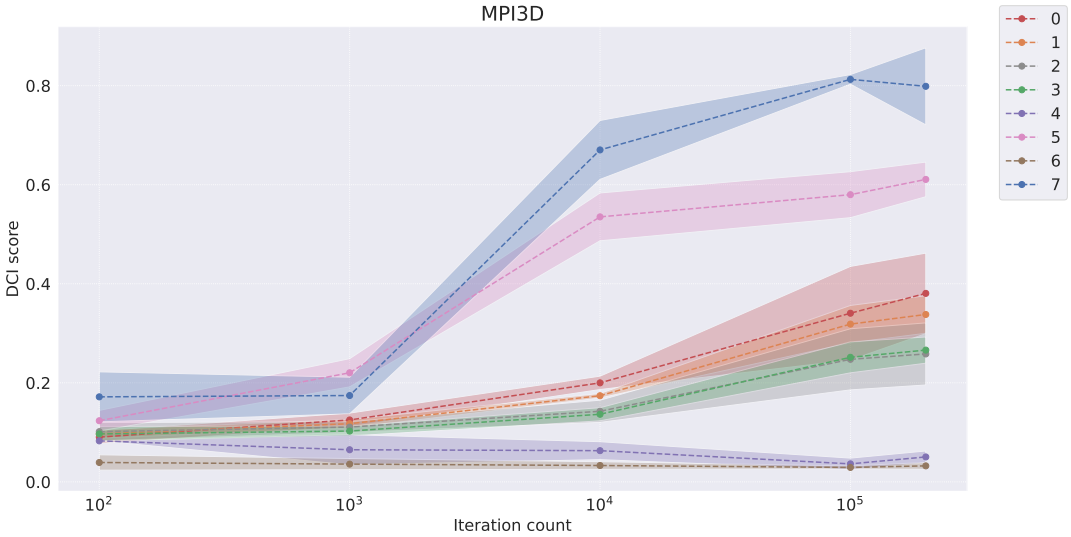

Figure 12: DCI score convergence on the MPI3D dataset

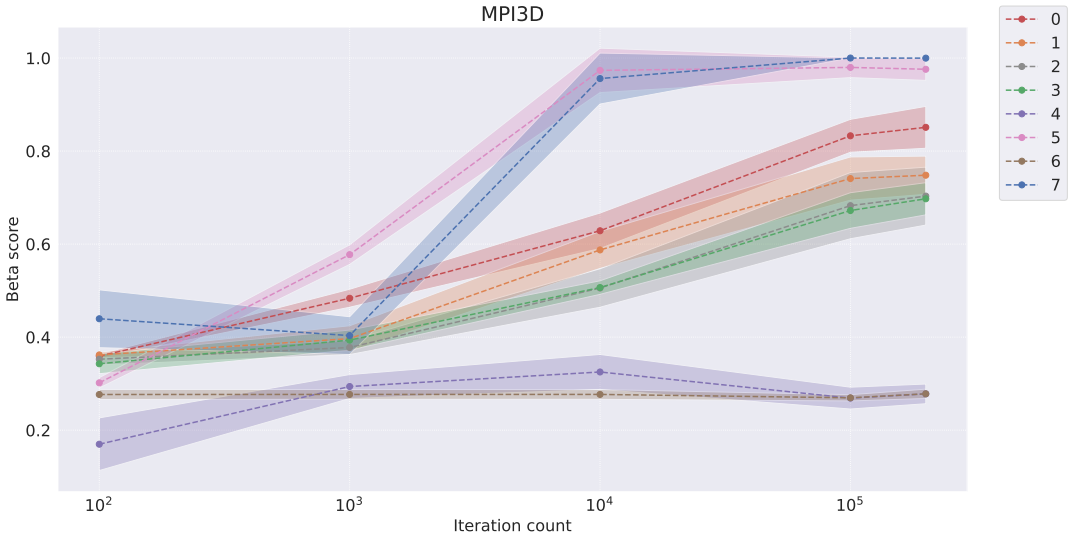

Figure 13: BetaVAE score convergence on the MPI3D dataset

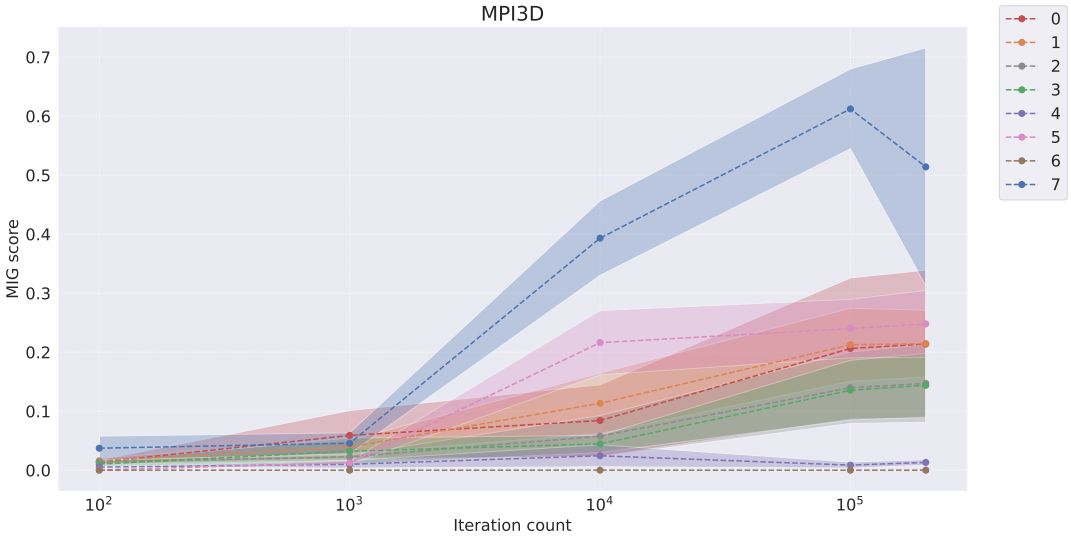

Figure 14: MIG score convergence on the MPI3D dataset

Table 18: Representation learner convergence on the Cars3D dataset (Factor score)

Models	Factor score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.455 ± 0.020	0.546 ± 0.043	0.624 ± 0.050	0.713 ± 0.054	0.743 ± 0.046
Ada-GVAE-k	0.460 ± 0.024	0.642 ± 0.031	0.755 ± 0.042	0.873 ± 0.077	0.904 ± 0.082
GVAE	0.478 ± 0.026	0.560 ± 0.036	0.641 ± 0.052	0.701 ± 0.044	0.722 ± 0.035
MLVAE	0.481 ± 0.022	0.575 ± 0.059	0.652 ± 0.041	0.721 ± 0.035	0.719 ± 0.031
Shu	0.000 ± 0.000	0.294 ± 0.162	0.229 ± 0.194	0.000 ± 0.000	0.069 ± 0.138
Symbolic vector-tokened compositional representations					
VCT	*0.618 ± 0.046*	0.887 ± 0.074	*0.907 ± 0.036*	*0.918 ± 0.037*	*0.915 ± 0.028*
COMET	0.531 ± 0.024	0.531 ± 0.024	0.531 ± 0.024	0.531 ± 0.024	0.531 ± 0.024
Fully continuous compositional representations					
Ours	**0.639 ± 0.045**	**0.898 ± 0.038**	**0.999 ± 0.001**	**0.999 ± 0.001**	**0.999 ± 0.001**

Table 19: Representation learner convergence on the Cars3D dataset (DCI score)

Models	DCI score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.047 ± 0.012	0.115 ± 0.021	0.154 ± 0.031	0.261 ± 0.041	0.308 ± 0.073
Ada-GVAE-k	0.054 ± 0.011	0.165 ± 0.012	*0.338 ± 0.042*	*0.558 ± 0.167*	*0.694 ± 0.140*
GVAE	0.071 ± 0.024	0.098 ± 0.021	0.147 ± 0.025	0.216 ± 0.053	0.238 ± 0.061
MLVAE	*0.062 ± 0.022*	0.100 ± 0.018	0.175 ± 0.050	0.197 ± 0.062	0.238 ± 0.077
Shu	0.007 ± 0.005	0.013 ± 0.011	0.018 ± 0.012	0.038 ± 0.036	0.026 ± 0.015
Symbolic vector-tokened compositional representations					
VCT	0.055 ± 0.044	*0.240 ± 0.043*	0.227 ± 0.024	0.226 ± 0.020	0.230 ± 0.044
COMET	0.020 ± 0.013	0.015 ± 0.011	0.011 ± 0.006	0.008 ± 0.003	0.009 ± 0.003
Fully continuous compositional representations					
Ours	**0.379 ± 0.117**	**0.381 ± 0.026**	**0.812 ± 0.033**	**0.843 ± 0.024**	**0.832 ± 0.048**

Table 20: Representation learner convergence on the Cars3D dataset (BetaVAE score)

Models	BetaVAE score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.937 ± 0.025	0.999 ± 0.002 (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
Ada-GVAE-k	**0.971 ± 0.011**	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
GVAE	*0.958 ± 0.027*	*0.999 ± 0.001* (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
MLVAE	0.964 ± 0.011	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
Shu	0.534 ± 0.164	0.742 ± 0.072	*0.728 ± 0.105*	0.433 ± 0.016	0.449 ± 0.099
Symbolic vector-tokened compositional representations					
VCT	0.889 ± 0.044	**1.000 ± 0.001** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
COMET	0.592 ± 0.027	0.592 ± 0.028	0.592 ± 0.028	*0.592 ± 0.028*	*0.592 ± 0.028*
Fully continuous compositional representations					
Ours	0.953 ± 0.042	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000**

Table 21: Representation learner convergence on the Cars3D dataset (MIG score)

Models	MIG score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.008 ± 0.004	0.042 ± 0.017	0.061 ± 0.024	0.071 ± 0.028	0.081 ± 0.031
Ada-GVAE-k	*0.011 ± 0.005*	*0.068 ± 0.013*	*0.092 ± 0.035*	*0.241 ± 0.095*	*0.312 ± 0.085*
GVAE	0.014 ± 0.006	0.031 ± 0.015	0.056 ± 0.008	0.084 ± 0.031	0.095 ± 0.032
MLVAE	0.012 ± 0.007	0.037 ± 0.024	0.057 ± 0.018	0.085 ± 0.021	0.087 ± 0.017
Shu	0.007 ± 0.004	0.014 ± 0.005	0.019 ± 0.010	0.002 ± 0.001	0.001 ± 0.001
Symbolic vector-tokened compositional representations					
VCT	0.001 ± 0.001	0.034 ± 0.016	0.033 ± 0.021	0.033 ± 0.022	0.032 ± 0.015
COMET	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Fully continuous compositional representations					
Ours	**0.016 ± 0.010**	**0.136 ± 0.009**	**0.344 ± 0.011**	**0.349 ± 0.010**	**0.349 ± 0.010**

Table 22: Representation learner convergence on the Shapes3D dataset (Factor score)

Models	Factor score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.270 ± 0.025	0.392 ± 0.095	0.547 ± 0.138	0.762 ± 0.156	0.923 ± 0.091
Ada-GVAE-k	*0.296 ± 0.047*	*0.577 ± 0.023*	*0.815 ± 0.057*	0.960 ± 0.057	*0.961 ± 0.070*
GVAE	0.263 ± 0.036	0.469 ± 0.063	0.674 ± 0.086	0.890 ± 0.058	0.879 ± 0.077
MLVAE	0.166 ± 0.084	0.457 ± 0.045	0.621 ± 0.078	0.746 ± 0.080	0.765 ± 0.084
Shu	0.000 ± 0.000	0.000 ± 0.000	0.242 ± 0.072	0.194 ± 0.021	0.227 ± 0.016
Symbolic vector-tokened compositional representations					
VCT	0.263 ± 0.000	**0.984 ± 0.031**	**1.000 ± 0.000**	*0.976 ± 0.033*	**0.967 ± 0.044**
COMET	0.268 ± 0.022	0.265 ± 0.039	0.334 ± 0.028	0.384 ± 0.040	0.440 ± 0.071
Fully continuous compositional representations					
Ours	**0.356 ± 0.011**	0.435 ± 0.048	*0.975 ± 0.040*	**0.995 ± 0.005**	0.960 ± 0.053

Table 23: Representation learner convergence on the Shapes3D dataset (DCI score)

Models	DCI score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	*0.056 ± 0.015*	0.192 ± 0.060	0.353 ± 0.111	0.616 ± 0.147	0.749 ± 0.118
Ada-GVAE-k	0.053 ± 0.018	*0.331 ± 0.022*	0.662 ± 0.042	**0.944 ± 0.063**	**0.964 ± 0.067**
GVAE	0.042 ± 0.013	0.198 ± 0.046	0.438 ± 0.064	0.749 ± 0.053	0.835 ± 0.038
MLVAE	0.026 ± 0.003	0.183 ± 0.036	0.407 ± 0.051	0.668 ± 0.084	0.739 ± 0.103
Shu	0.030 ± 0.007	0.028 ± 0.012	0.032 ± 0.006	0.019 ± 0.010	0.018 ± 0.006
Symbolic vector-tokened compositional representations					
VCT	0.044 ± 0.000	**0.817 ± 0.114**	**0.909 ± 0.010**	*0.887 ± 0.026*	0.880 ± 0.022
COMET	0.017 ± 0.006	0.031 ± 0.005	0.062 ± 0.017	0.173 ± 0.064	0.233 ± 0.066
Fully continuous compositional representations					
Ours	**0.057 ± 0.018**	0.183 ± 0.077	*0.860 ± 0.052*	0.887 ± 0.068	*0.924 ± 0.027*

Table 24: Representation learner convergence on the Shapes3D dataset (BetaVAE score)

Models	BetaVAE score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.539 ± 0.061	0.694 ± 0.012	0.949 ± 0.037	*0.989 ± 0.020*	**1.000 ± 0.000** (=)
Ada-GVAE-k	0.553 ± 0.050	0.699 ± 0.016	*0.999 ± 0.001*	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
GVAE	0.505 ± 0.031	0.683 ± 0.019	0.974 ± 0.025	**1.000 ± 0.000** (=)	*0.998 ± 0.004*
MLVAE	0.484 ± 0.015	0.684 ± 0.013	0.970 ± 0.029	0.949 ± 0.041	0.974 ± 0.036
Shu	0.311 ± 0.204	0.357 ± 0.050	0.495 ± 0.040	0.379 ± 0.056	0.403 ± 0.058
Symbolic vector-tokened compositional representations					
VCT	**0.889 ± 0.044**	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
COMET	0.395 ± 0.000	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)
Fully continuous compositional representations					
Ours	*0.638 ± 0.015*	*0.710 ± 0.035*	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)	**1.000 ± 0.000** (=)

Table 25: Representation learner convergence on the Shapes3D dataset (MIG score)

Models	MIG score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.018 ± 0.009	0.080 ± 0.053	0.163 ± 0.107	0.297 ± 0.156	0.386 ± 0.169
Ada-GVAE-k	*0.023 ± 0.012*	*0.189 ± 0.071*	0.288 ± 0.138	**0.520 ± 0.149**	**0.555 ± 0.152**
GVAE	0.019 ± 0.012	0.074 ± 0.053	0.153 ± 0.039	0.225 ± 0.052	0.243 ± 0.064
MLVAE	0.010 ± 0.004	0.082 ± 0.042	0.179 ± 0.076	0.305 ± 0.126	0.354 ± 0.148
Shu	**0.025 ± 0.007**	0.008 ± 0.005	0.013 ± 0.005	0.010 ± 0.003	0.008 ± 0.004
Symbolic vector-tokened compositional representations					
VCT	0.000 ± 0.000	**0.405 ± 0.103**	**0.466 ± 0.023**	*0.469 ± 0.025*	*0.448 ± 0.065*
COMET	0.000 ± 0.000	0.004 ± 0.003	0.013 ± 0.010	0.037 ± 0.022	0.044 ± 0.025
Fully continuous compositional representations					
Ours	0.015 ± 0.005	0.043 ± 0.018	*0.343 ± 0.104*	0.412 ± 0.066	0.402 ± 0.091

Table 26: Representation learner convergence on the MPI3D dataset (Factor score)

Models	Factor score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.240 ± 0.008	0.300 ± 0.015	0.325 ± 0.014	0.458 ± 0.095	0.471 ± 0.109
Ada-GVAE-k	*0.246 ± 0.012*	0.275 ± 0.011	0.321 ± 0.011	0.314 ± 0.050	0.318 ± 0.050
GVAE	0.236 ± 0.020	0.253 ± 0.017	0.277 ± 0.023	0.237 ± 0.020	0.232 ± 0.023
MLVAE	0.235 ± 0.008	0.257 ± 0.014	0.261 ± 0.018	0.229 ± 0.021	0.240 ± 0.019
Shu	0.000 ± 0.000	0.061 ± 0.075	0.173 ± 0.089	0.093 ± 0.077	0.124 ± 0.056
Symbolic vector-tokened compositional representations					
VCT	0.220 ± 0.024	**0.360 ± 0.017**	*0.441 ± 0.037*	*0.678 ± 0.125*	*0.678 ± 0.045*
COMET	0.227 ± 0.019	0.233 ± 0.016	0.233 ± 0.016	0.243 ± 0.005	0.233 ± 0.015
Fully continuous compositional representations					
Ours	**0.334 ± 0.079**	*0.350 ± 0.049*	**0.750 ± 0.065**	**0.937 ± 0.034**	**0.951 ± 0.083**

Table 27: Representation learner convergence on the MPI3D dataset (DCI score)

Models	DCI score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.090 ± 0.013	0.125 ± 0.014	0.200 ± 0.014	0.341 ± 0.095	0.381 ± 0.081
Ada-GVAE-k	0.097 ± 0.008	0.117 ± 0.009	0.174 ± 0.005	0.319 ± 0.038	0.338 ± 0.038
GVAE	0.101 ± 0.019	0.111 ± 0.011	0.142 ± 0.022	0.247 ± 0.062	0.258 ± 0.063
MLVAE	0.097 ± 0.015	0.103 ± 0.010	0.137 ± 0.013	0.252 ± 0.031	0.266 ± 0.026
Shu	0.083 ± 0.000	0.065 ± 0.030	0.063 ± 0.018	0.036 ± 0.012	0.050 ± 0.012
Symbolic vector-tokened compositional representations					
VCT	*0.123 ± 0.021*	**0.220 ± 0.029**	*0.535 ± 0.048*	*0.580 ± 0.046*	*0.611 ± 0.035*
COMET	0.039 ± 0.016	0.036 ± 0.011	0.033 ± 0.008	0.029 ± 0.003	0.032 ± 0.007
Fully continuous compositional representations					
Ours	**0.172 ± 0.051**	*0.174 ± 0.037*	**0.670 ± 0.060**	**0.813 ± 0.010**	**0.799 ± 0.078**

Table 28: Representation learner convergence on the MPI3D dataset (BetaVAE score)

Models	BetaVAE score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.359 ± 0.004	0.484 ± 0.019	0.629 ± 0.038	0.833 ± 0.035	0.851 ± 0.045
Ada-GVAE-k	*0.362 ± 0.006*	0.397 ± 0.028	0.587 ± 0.040	0.741 ± 0.047	0.748 ± 0.041
GVAE	0.352 ± 0.012	0.378 ± 0.015	0.506 ± 0.041	0.683 ± 0.071	0.703 ± 0.062
MLVAE	0.343 ± 0.021	0.394 ± 0.022	0.507 ± 0.015	0.672 ± 0.038	0.697 ± 0.035
Shu	0.170 ± 0.057	0.294 ± 0.026	0.325 ± 0.038	0.269 ± 0.023	0.278 ± 0.021
Symbolic vector-tokened compositional representations					
VCT	**0.889 ± 0.044**	**1.000 ± 0.001**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**
COMET	0.277 ± 0.011	0.277 ± 0.011	0.277 ± 0.011	0.270 ± 0.007	0.278 ± 0.010
Fully continuous compositional representations					
Ours	0.302 ± 0.011	*0.577 ± 0.021*	*0.973 ± 0.048*	*0.980 ± 0.022*	*0.976 ± 0.024*

Table 29: Representation learner convergence on the MPI3D dataset (MIG score)

Models	MIG score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.014 ± 0.004	**0.059 ± 0.042**	0.084 ± 0.061	0.206 ± 0.120	0.214 ± 0.125
Ada-GVAE-k	*0.016 ± 0.005*	0.039 ± 0.011	0.113 ± 0.051	0.213 ± 0.062	0.214 ± 0.057
GVAE	0.014 ± 0.004	0.022 ± 0.003	0.058 ± 0.035	0.140 ± 0.061	0.147 ± 0.066
MLVAE	0.010 ± 0.004	0.032 ± 0.022	0.044 ± 0.015	0.136 ± 0.051	0.144 ± 0.054
Shu	0.005 ± 0.000	0.010 ± 0.006	0.024 ± 0.019	0.009 ± 0.005	0.013 ± 0.005
Symbolic vector-tokened compositional representations					
VCT	0.000 ± 0.000	0.013 ± 0.003	*0.216 ± 0.055*	*0.240 ± 0.050*	*0.248 ± 0.057*
COMET	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Fully continuous compositional representations					
Ours	**0.037 ± 0.020**	*0.046 ± 0.018*	**0.393 ± 0.063**	**0.612 ± 0.068**	**0.514 ± 0.202**

C.4.2 Downstream Performance

We additionally evaluate the representation learning convergence by examining the usefulness of representations produced at different stages of training (i.e., 100, 250, 500, 1,000, 10,000, 100,000

and 200,000 iterations of training). To quantify 'usefulness', we consider performance of downstream models on the tasks of FoV regression, and abstract visual reasoning when trained using representations produced by each stage of training. For each task, we present line plots, and additionally tables with values corresponding to each of the plots. We use the same legend as in the previous section, where 0 (grey) denotes SlowVAE, 1 (orange) denotes AdaGVAE-k, 2 (green) denotes GVAE, 3 (red) denotes MLVAE, 4 (purple) denotes Shu, 5 (pink) denotes VCT, 6 (brown) denotes COMET, and 7 (blue) denotes our model, Soft TPR Autoencoder. We additionally provide all results for the dimensionality-control setting, where the dimensionality of representations produced by all representation learners is held constant following the approach detailed in C.2.4, denoting this clearly in plot captions, and by the symbol † in the tables.

FoV Regression

As clearly visible in the tables and plots, generic regression models are able to more effectively use representations produced by our Soft TPR Autoencoder produced across almost all stages of training for all disentanglement datasets. Improvements are most notable in the low-iteration regime of 10^2 iterations, and across most stages of training for the more challenging task of FoV regression on the MPI3D dataset (Figures 19 and 20).

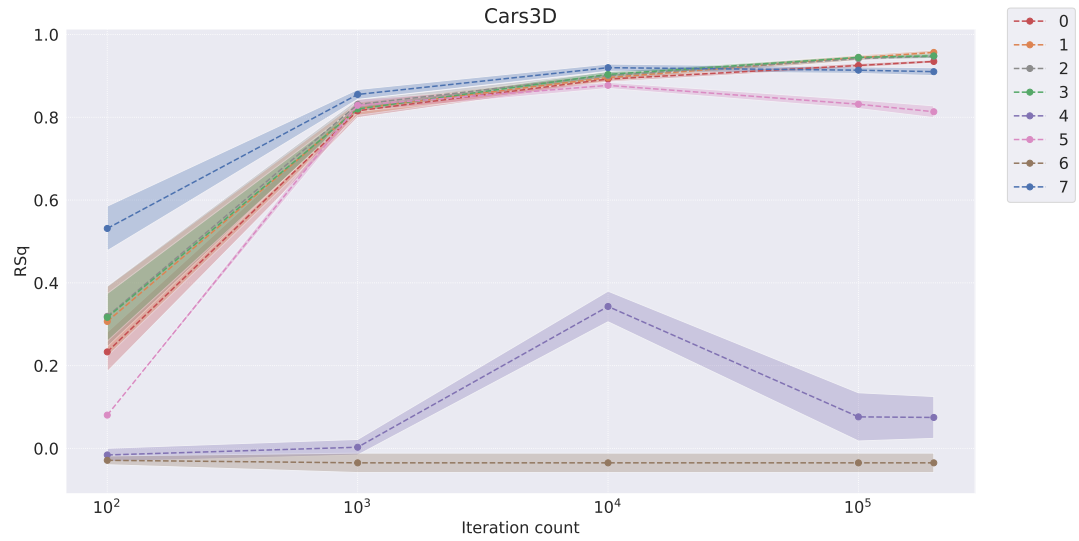

Figure 15: Convergence of representation learners as measured by FoV regression on the Cars3D dataset (original setting)

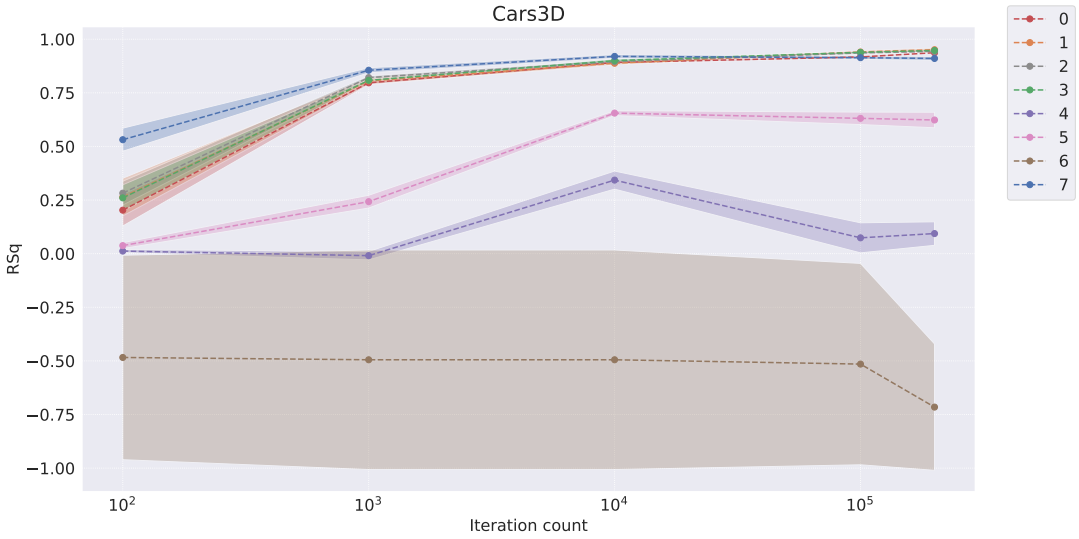

Figure 16: Convergence of representation learners as measured by FoV regression on the Cars3D dataset (dimensionality-controlled setting)

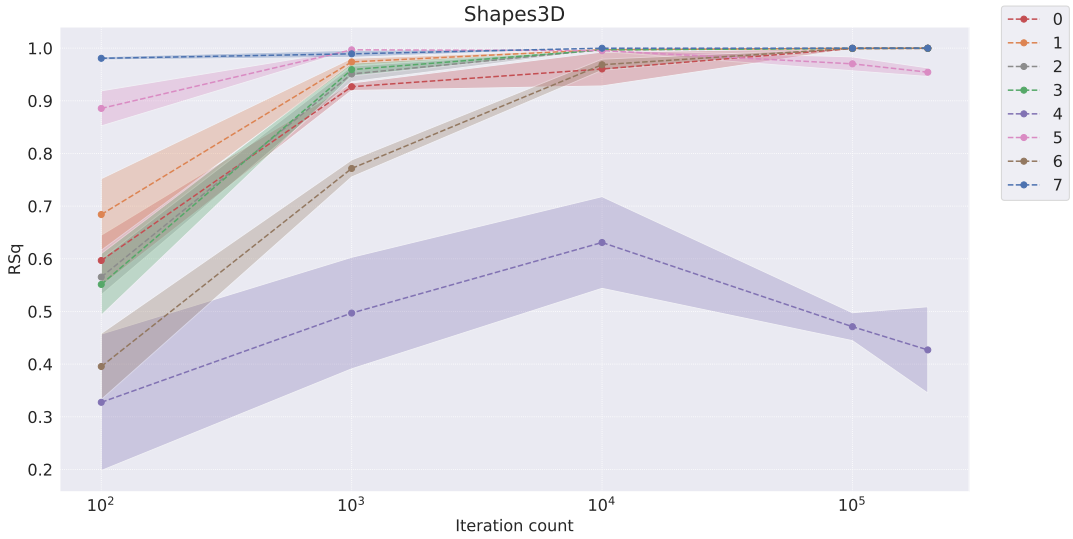

Figure 17: Convergence of representation learners as measured by FoV regression on the Shapes3D dataset (original setting)

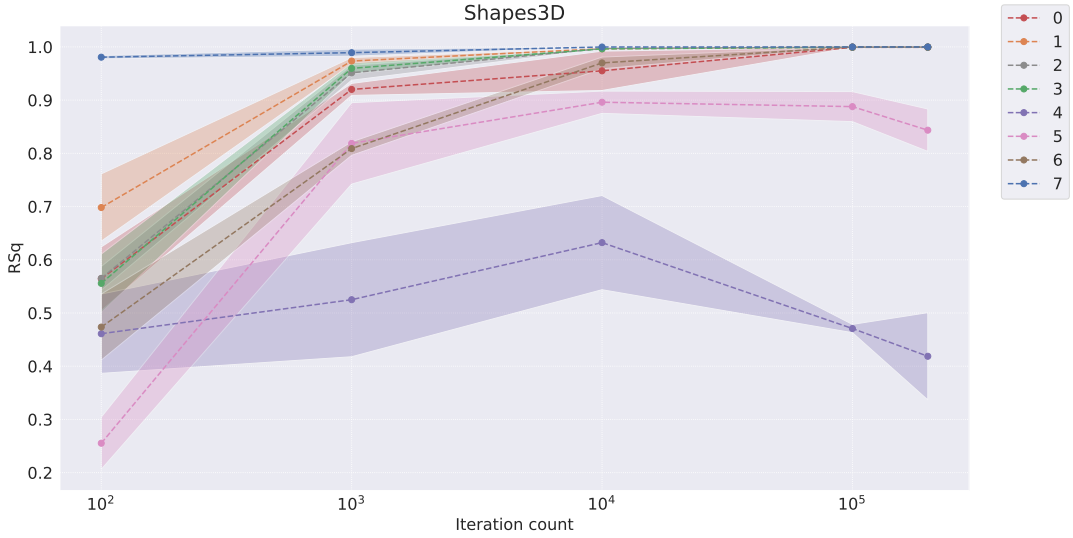

Figure 18: Convergence of representation learners as measured by FoV regression on the Shapes3D dataset (dimensionality-controlled setting)

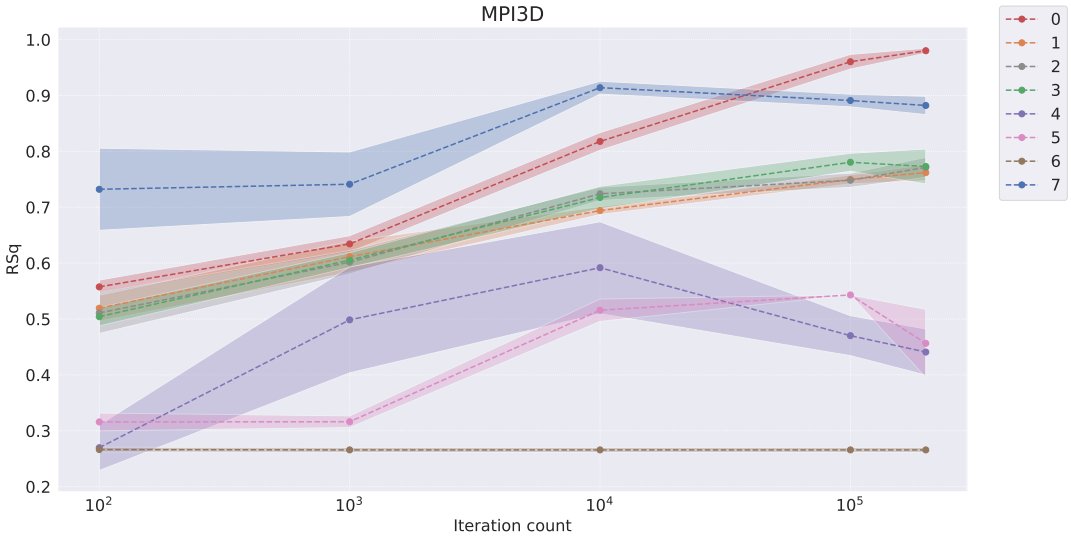

Figure 19: Convergence of representation learners as measured by FoV regression on the MPI3D dataset (original setting)

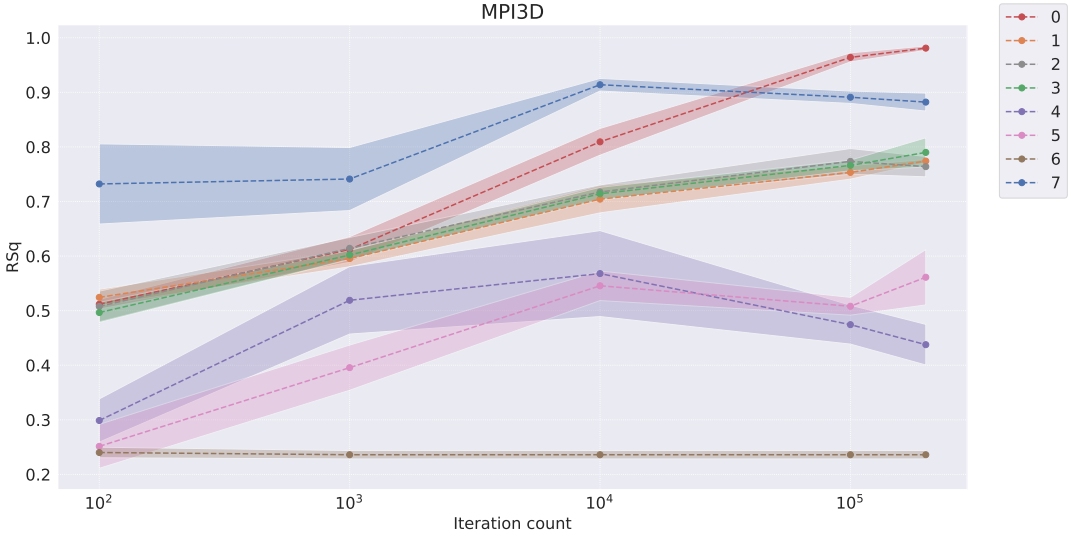

Figure 20: Convergence of representation learners as measured by FoV regression on the MPI3D dataset (dimensionality-controlled setting)

Table 30: Convergence of representation learners as measured by FoV regression on the Cars3D dataset

Models	R^2 score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.233 ± 0.048	0.816 ± 0.017	0.892 ± 0.007	0.925 ± 0.005	0.935 ± 0.004
Slow-VAE†	0.203 ± 0.074	0.797 ± 0.009	0.891 ± 0.006	0.917 ± 0.006	0.937 ± 0.001
Ada-GVAE-k	0.307 ± 0.084	0.821 ± 0.016	0.896 ± 0.009	0.944 ± 0.007	$\mathbf{0.957 \pm 0.005}$
Ada-GVAE-k†	0.264 ± 0.088	0.805 ± 0.015	0.888 ± 0.009	0.940 ± 0.007	$\mathit{0.951 \pm 0.008}$
GVAE	$\mathit{0.319 \pm 0.073}$	0.831 ± 0.012	$\mathit{0.901 \pm 0.009}$	$\mathit{0.942 \pm 0.004}$	0.947 ± 0.005
GVAE†	0.282 ± 0.056	0.821 ± 0.008	0.895 ± 0.009	0.936 ± 0.007	0.943 ± 0.004
MLVAE	0.317 ± 0.058	0.820 ± 0.007	0.904 ± 0.007	$\mathbf{0.945 \pm 0.004}$	0.948 ± 0.007
MLVAE†	0.26 ± 0.061	0.808 ± 0.005	0.900 ± 0.009	0.939 ± 0.004	0.948 ± 0.007
Shu	-0.016 ± 0.016	0.003 ± 0.019	0.343 ± 0.037	0.076 ± 0.058	0.075 ± 0.051
Shu†	0.012 ± 0.007	-0.009 ± 0.02	0.343 ± 0.042	0.074 ± 0.070	0.094 ± 0.055
Symbolic vector-tokened compositional representations					
VCT	0.080 ± 0.001	$\mathit{0.829 \pm 0.015}$	0.877 ± 0.007	0.832 ± 0.010	0.813 ± 0.014
VCT†	0.038 ± 0.014	0.243 ± 0.03	0.655 ± 0.011	0.631 ± 0.029	0.623 ± 0.036
COMET	-0.029 ± 0.011	-0.035 ± 0.023	-0.035 ± 0.023	-0.035 ± 0.023	-0.035 ± 0.023
COMET†	-0.484 ± 0.477	-0.495 ± 0.512	-0.495 ± 0.512	-0.515 ± 0.47	-0.715 ± 0.295
Fully continuous compositional representations					
Ours	$\mathbf{0.531 \pm 0.054}$	$\mathbf{0.855 \pm 0.012}$	$\mathbf{0.920 \pm 0.009}$	0.914 ± 0.008	0.910 ± 0.010

Table 31: Convergence of representation learners as measured by FoV regression on the Shapes3D dataset

Models	R^2 score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.597 ± 0.048	0.927 ± 0.007	0.960 ± 0.032	$\mathit{0.999 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
Slow-VAE†	0.564 ± 0.060	0.920 ± 0.011	0.955 ± 0.037	$\mathit{0.999 \pm 0.000}$ (=)	$\mathit{0.999 \pm 0.000}$
Ada-GVAE-k	0.684 ± 0.068	0.974 ± 0.007	0.997 ± 0.000	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
Ada-GVAE-k†	0.698 ± 0.063	0.973 ± 0.007	0.997 ± 0.000	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
GVAE	0.565 ± 0.034	0.951 ± 0.015	0.997 ± 0.000	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
GVAE†	0.565 ± 0.022	0.951 ± 0.014	$\mathit{0.998 \pm 0.001}$	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
MLVAE	0.551 ± 0.059	0.959 ± 0.012	0.997 ± 0.000	$\mathit{0.999 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
MLVAE†	0.556 ± 0.055	0.960 ± 0.010	0.996 ± 0.000	$\mathit{0.999 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
Shu	0.327 ± 0.13	0.497 ± 0.106	0.631 ± 0.087	0.471 ± 0.027	0.427 ± 0.082
Shu†	0.461 ± 0.075	0.525 ± 0.107	0.632 ± 0.088	0.471 ± 0.008	0.419 ± 0.082
Symbolic vector-tokened compositional representations					
VCT	$\mathit{0.886 \pm 0.033}$	$\mathbf{0.997 \pm 0.000}$	0.995 ± 0.001	0.97 ± 0.013	0.954 ± 0.008
VCT†	0.255 ± 0.049	0.819 ± 0.077	0.896 ± 0.021	0.888 ± 0.028	0.843 ± 0.04
COMET	0.395 ± 0.063	0.772 ± 0.016	0.968 ± 0.011	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
COMET†	0.474 ± 0.062	0.809 ± 0.013	0.970 ± 0.011	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)
Fully continuous compositional representations					
Ours	$\mathbf{0.981 \pm 0.003}$	$\mathit{0.989 \pm 0.007}$	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)	$\mathbf{1.000 \pm 0.000}$ (=)

Table 32: Convergence of representation learners as measured by FoV regression on the MPI3D dataset

Models	R^2 score				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	*0.557 ± 0.012*	*0.634 ± 0.015*	*0.818 ± 0.016*	*0.960 ± 0.013*	*0.980 ± 0.005*
Slow-VAE†	0.512 ± 0.009	0.612 ± 0.023	0.809 ± 0.025	**0.964 ± 0.008**	**0.981 ± 0.004**
Ada-GVAE-k	0.519 ± 0.023	0.611 ± 0.028	0.694 ± 0.007	0.750 ± 0.008	0.762 ± 0.014
Ada-GVAE-k†	0.524 ± 0.015	0.595 ± 0.015	0.704 ± 0.025	0.753 ± 0.012	0.774 ± 0.003
GVAE	0.511 ± 0.037	0.602 ± 0.021	0.724 ± 0.012	0.748 ± 0.013	0.771 ± 0.018
GVAE†	0.508 ± 0.028	0.614 ± 0.020	0.717 ± 0.013	0.773 ± 0.024	0.764 ± 0.019
MLVAE	0.504 ± 0.016	0.605 ± 0.013	0.717 ± 0.020	0.780 ± 0.016	0.773 ± 0.031
MLVAE†	0.497 ± 0.019	0.602 ± 0.008	0.714 ± 0.012	0.766 ± 0.010	0.790 ± 0.027
Shu	0.270 ± 0.041	0.498 ± 0.095	0.592 ± 0.082	0.470 ± 0.036	0.441 ± 0.041
Shu†	0.299 ± 0.040	0.519 ± 0.062	0.568 ± 0.079	0.474 ± 0.036	0.438 ± 0.037
Symbolic vector-tokened compositional representations					
VCT	0.316 ± 0.016	0.316 ± 0.011	0.516 ± 0.02	0.543 ± 0.000	0.456 ± 0.061
VCT†	0.251 ± 0.04	0.396 ± 0.041	0.546 ± 0.027	0.508 ± 0.016	0.561 ± 0.051
COMET	0.266 ± 0.004	0.266 ± 0.004	0.266 ± 0.004	0.266 ± 0.004	0.266 ± 0.004
COMET†	0.240 ± 0.010	0.236 ± 0.008	0.236 ± 0.008	0.236 ± 0.008	0.236 ± 0.008
Fully continuous compositional representations					
Ours	**0.732 ± 0.073**	**0.741 ± 0.058**	**0.914 ± 0.012**	0.891 ± 0.011	0.882 ± 0.016

Abstract Visual Reasoning

We present now present our full suite of results for the downstream task of abstract visual reasoning. As demonstrated in Table 33 and the corresponding Figures 21 and 22, representations produced by our model at *only* 10^2 iterations of training are able to be leveraged by downstream models to achieve a 80.04% accuracy for the challenging abstract visual reasoning task, in contrast to the value of 63.1% obtained by the best performing baseline, representing a 26.78% performance increase. This again provides strong empirical support for our hypothesis that the *approximate*, continuously-instantiated compositional structure embodied by our Soft TPR can be learnt more quickly by representation learners than alternative, symbolic representations of compositional structure, thereby allowing downstream models to effectively leverage these representations despite a small number of representation learner training iterations.

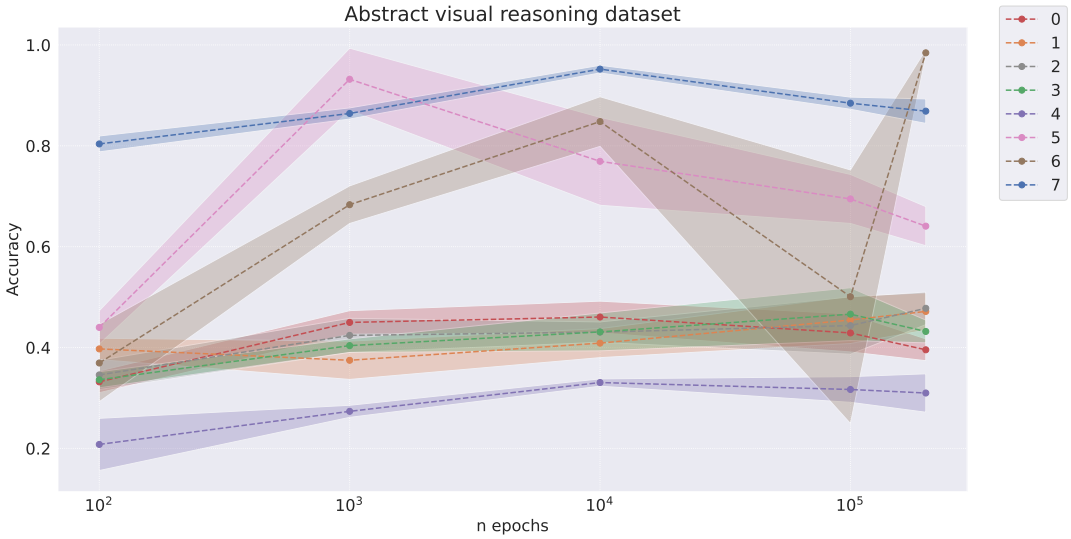

Figure 21: Convergence of representation learners as measured by classification performance on the abstract visual reasoning dataset (original setting)

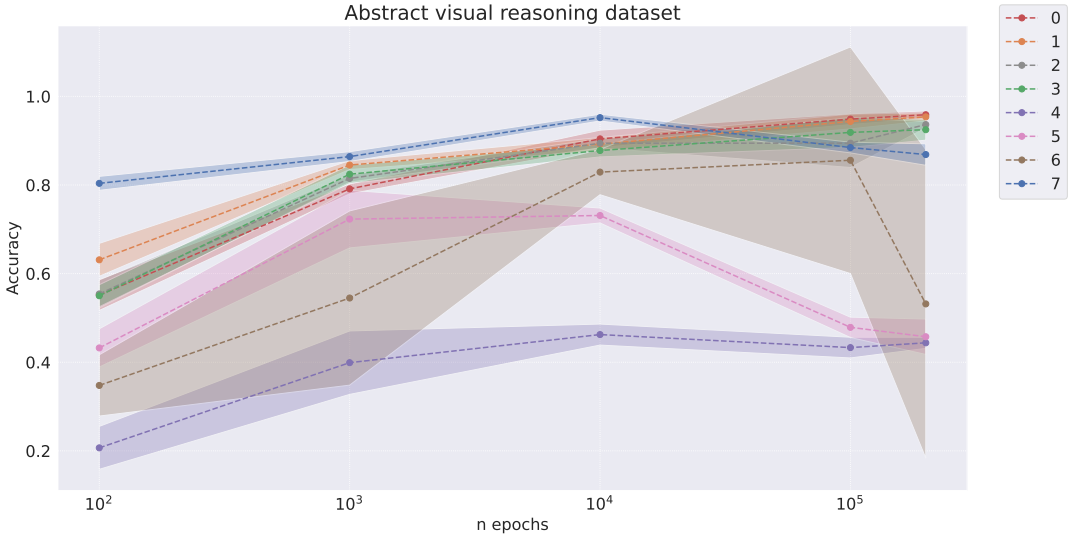

Figure 22: Convergence of representation learners as measured by classification performance on the abstract visual reasoning dataset (dimensionality-controlled setting)

Table 33: Convergence of representation learners as measured by classification performance on the abstract visual reasoning dataset

Models	Classification accuracy				
	10^2 iterations	10^3 iterations	10^4 iterations	10^5 iterations	2×10^5 iterations
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.332 ± 0.022	0.450 ± 0.023	0.460 ± 0.031	0.429 ± 0.038	0.396 ± 0.022
Slow-VAE†	0.552 ± 0.035	0.791 ± 0.011	0.904 ± 0.020	$\mathbf{0.949 \pm 0.012}$	*0.959 ± 0.009*
Ada-GVAE-k	0.397 ± 0.021	0.375 ± 0.038	0.409 ± 0.029	0.455 ± 0.047	0.472 ± 0.037
Ada-GVAE-k†	*0.631 ± 0.037*	0.845 ± 0.010	0.892 ± 0.015	*0.943 ± 0.017*	0.954 ± 0.009
GVAE	0.346 ± 0.029	0.424 ± 0.034	0.431 ± 0.020	0.444 ± 0.057	0.478 ± 0.032
GVAE†	0.554 ± 0.031	0.815 ± 0.008	0.894 ± 0.011	0.894 ± 0.055	0.936 ± 0.008
MLVAE	0.336 ± 0.016	0.404 ± 0.013	0.431 ± 0.038	0.466 ± 0.052	0.432 ± 0.023
MLVAE†	0.550 ± 0.025	0.824 ± 0.021	0.878 ± 0.014	0.919 ± 0.034	0.925 ± 0.024
Shu	0.208 ± 0.052	0.273 ± 0.012	0.331 ± 0.007	0.317 ± 0.025	0.310 ± 0.038
Shu†	0.207 ± 0.048	0.399 ± 0.072	0.462 ± 0.024	0.433 ± 0.023	0.444 ± 0.013
Symbolic vector-tokened compositional representations					
VCT	0.440 ± 0.033	$\mathbf{0.932 \pm 0.062}$	0.769 ± 0.087	0.695 ± 0.049	0.641 ± 0.039
VCT†	0.432 ± 0.043	0.723 ± 0.065	0.731 ± 0.017	0.479 ± 0.023	0.458 ± 0.040
COMET	0.369 ± 0.077	0.683 ± 0.037	0.848 ± 0.049	0.501 ± 0.251	$\mathbf{0.984 \pm 0.006}$
COMET†	0.348 ± 0.069	0.545 ± 0.197	0.829 ± 0.051	0.856 ± 0.256	0.532 ± 0.348
Fully continuous compositional representations					
Ours	$\mathbf{0.804 \pm 0.016}$	*0.864 ± 0.011*	$\mathbf{0.952 \pm 0.008}$	0.884 ± 0.012	0.869 ± 0.024

C.5 Downstream Performance

We present our full suite of experimental results that empirically demonstrate the utility of our Soft TPR representation from the perspective of downstream models, by considering the sample efficiency, and low-sample regime performance of downstream models on the tasks of FoV regression, and abstract visual reasoning.

C.5.1 Sample Efficiency Results

For sample efficiency, as mentioned in Section 5.2, and in line with [25], we compute a ratio-based metric obtained by dividing the performance of the downstream model when trained using a restricted number of 100, 250, 500, 1,000 and 10,000 samples, by its performance when trained using all samples. The total number of samples corresponds to 19,104, 480,000, 1,036,800 and 100,000 for the tasks of regression on the Cars3D, Shapes3D, MPI3D datasets, and the abstract visual reasoning task respectively. As this metric is dependent on the performance of downstream models when trained using all samples, we do not compute this metric for representation learners where the corresponding downstream models achieve an R^2 score of less than 0.5 for regression, as this may produce sample efficiency scores with very little semantic meaning (e.g. a model that achieves a sample efficiency score of 0.9 when its final R^2 score is 0.1). This corresponds to removing the Shu model from Shapes3D sample efficiency calculations, and COMET and Shu from the Cars3D sample efficiency calculations.

As many models for the abstract visual reasoning task have low classification accuracies on the held-out test set following training with the maximal number of 100,000 samples, we do not compute sample efficiencies for this task, and instead refer readers to results in Section C.5.2 for the raw classification accuracies associated with each model.

Note that for all box plots, we follow standard convention, and display the median in each box with a solid line, where the box shows the quartiles of the corresponding values, and the whiskers extend to 1.5 times the interquartile range. We again, use the same legend, where grey denotes SlowVAE, orange denotes AdaGVAE-k, green denotes GVAE, red denotes MLVAE, purple denotes Shu, pink denotes VCT, brown denotes COMET, and blue denotes our model, Soft TPR Autoencoder.

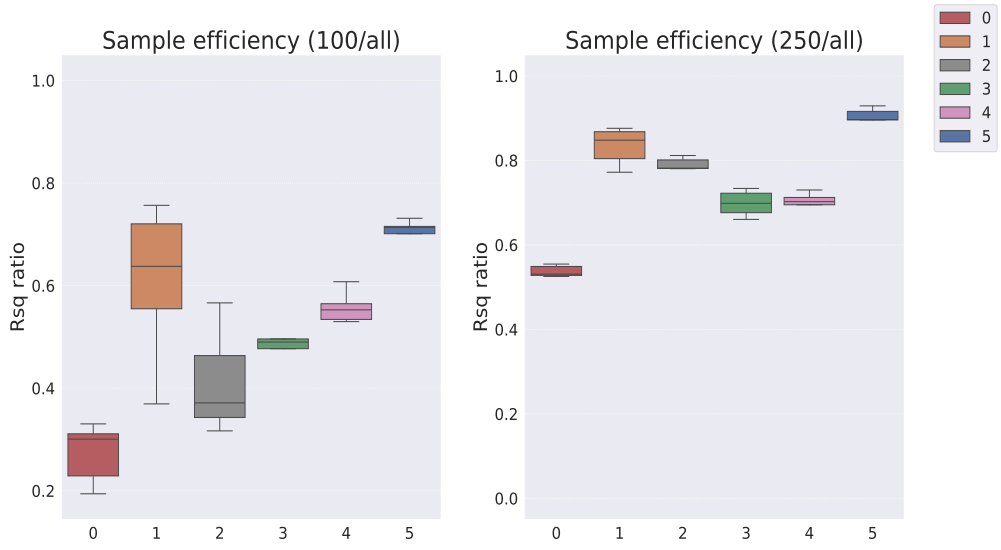

Figure 23: Downstream regression model sample efficiency on the Cars3D dataset (original setting).

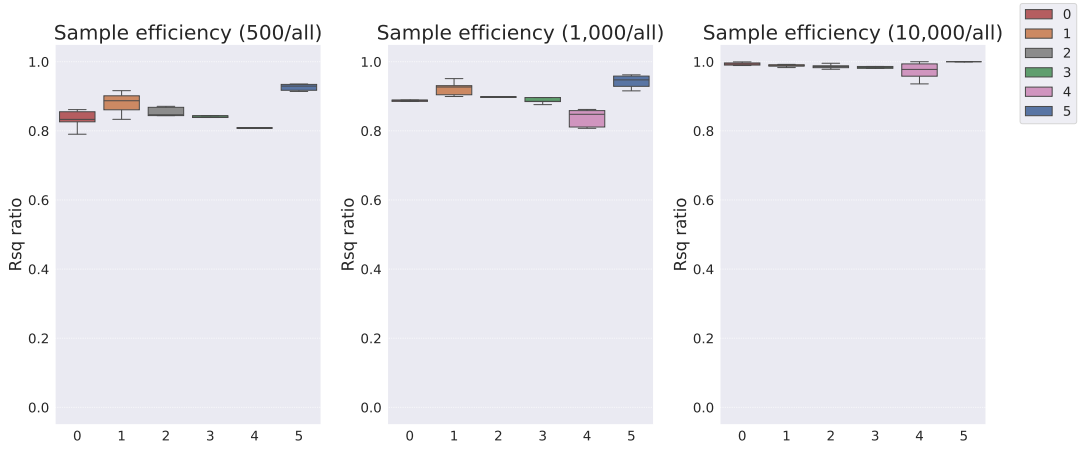

Figure 24: Downstream regression model sample efficiency on the Cars3D dataset (original setting).

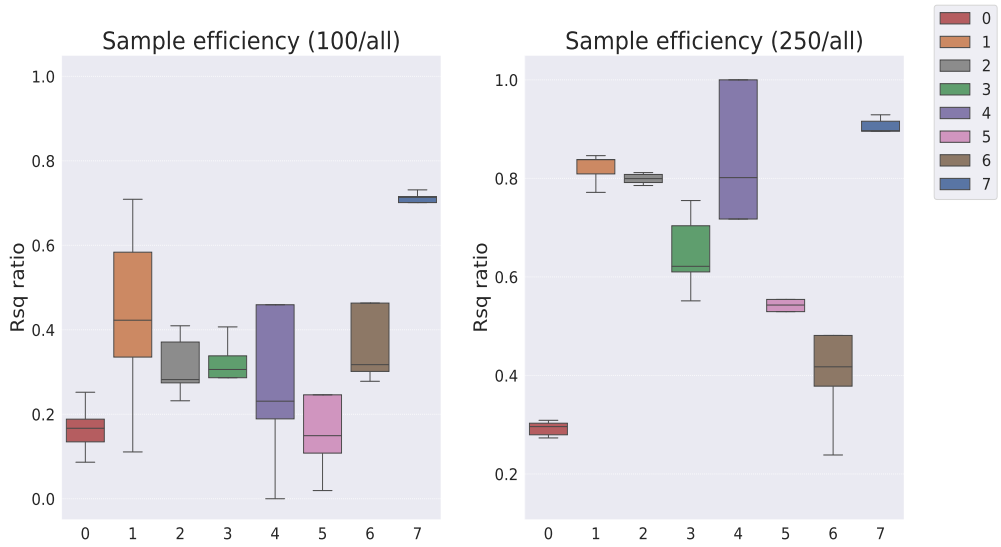

Figure 25: Downstream regression model sample efficiency on the Cars3D dataset (dimensionality-controlled setting).

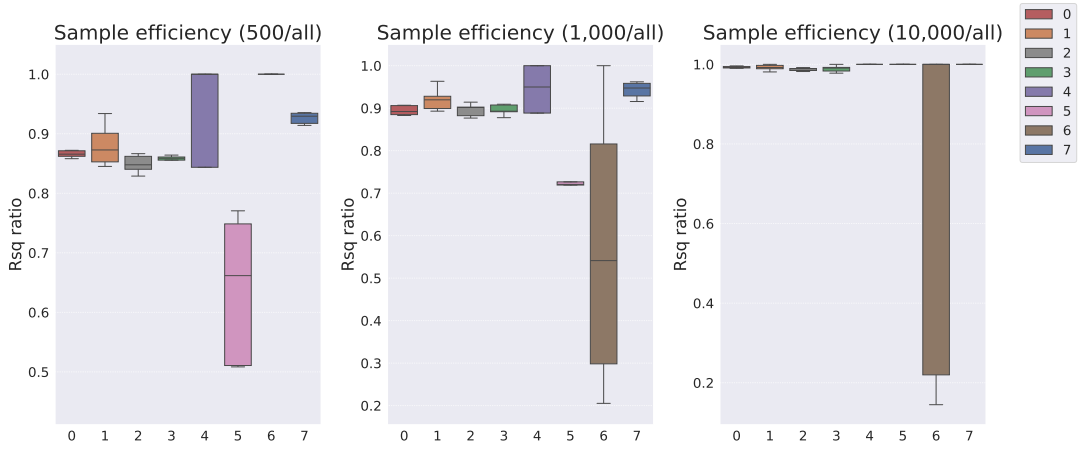

Figure 26: Downstream regression model sample efficiency on the Cars3D dataset (dimensionality-controlled setting).

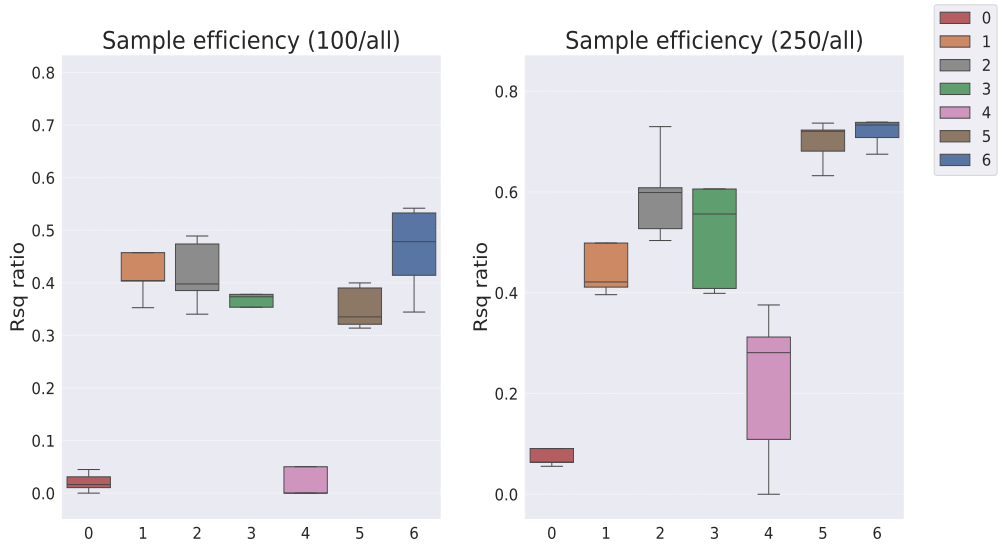

Figure 27: Downstream regression model sample efficiency on the Shapes3D dataset (original setting).

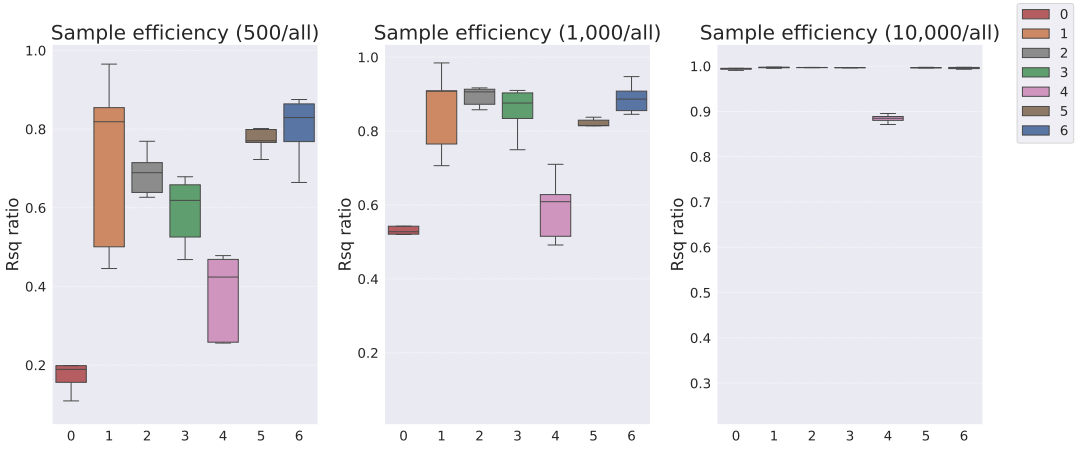

Figure 28: Downstream regression model sample efficiency on the Shapes3D dataset (original setting).

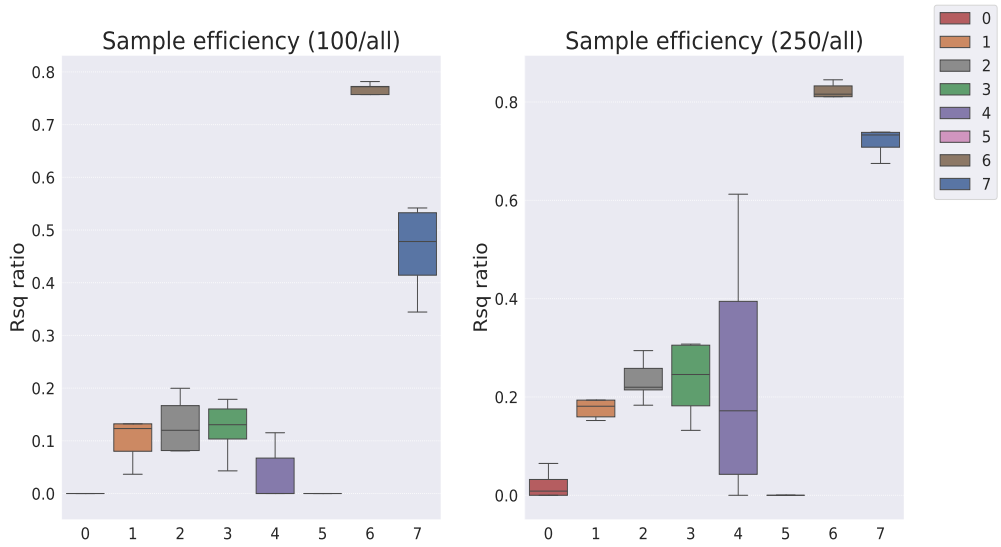

Figure 29: Downstream regression model sample efficiency on the Shapes3D dataset (dimensionality-controlled setting).

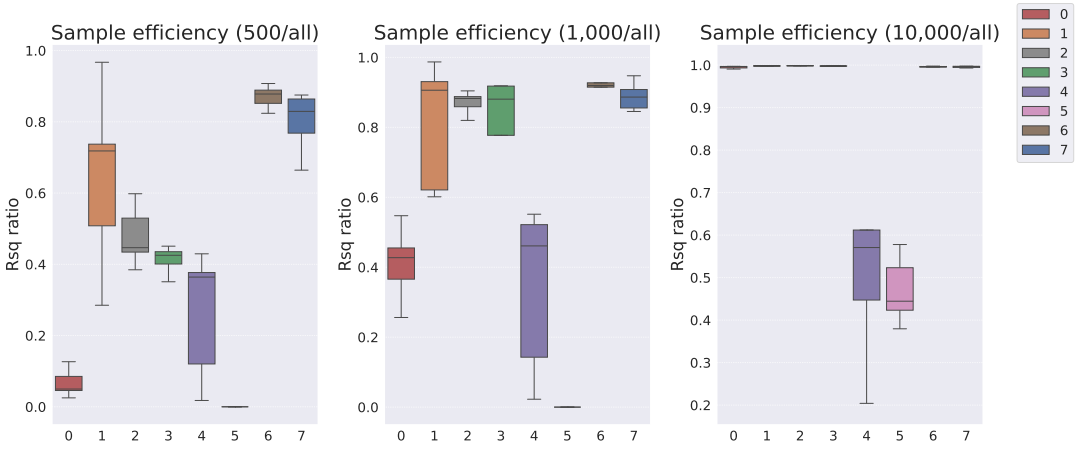

Figure 30: Downstream regression model sample efficiency on the Shapes3D dataset (dimensionality-controlled setting).

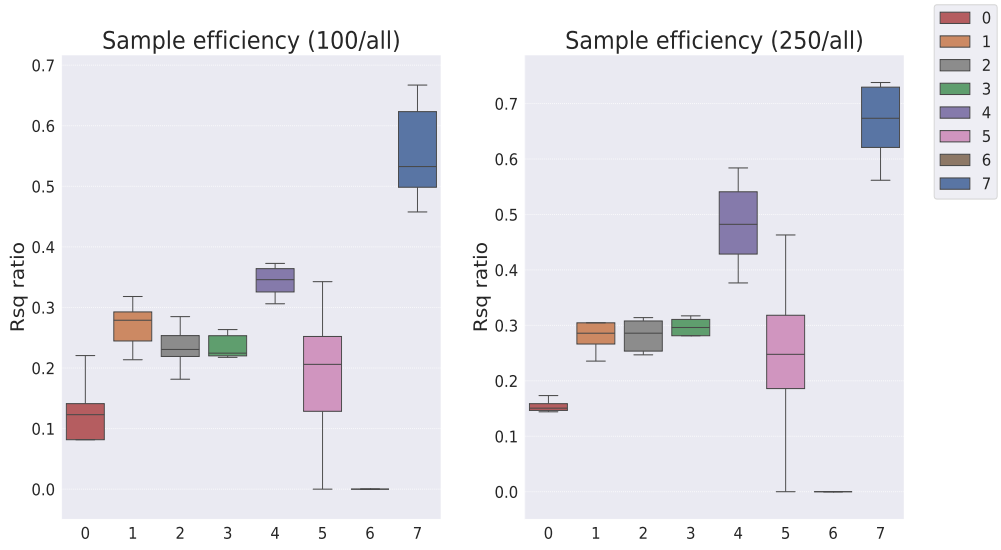

Figure 31: Downstream regression model sample efficiency on the MPI3D dataset (original setting).

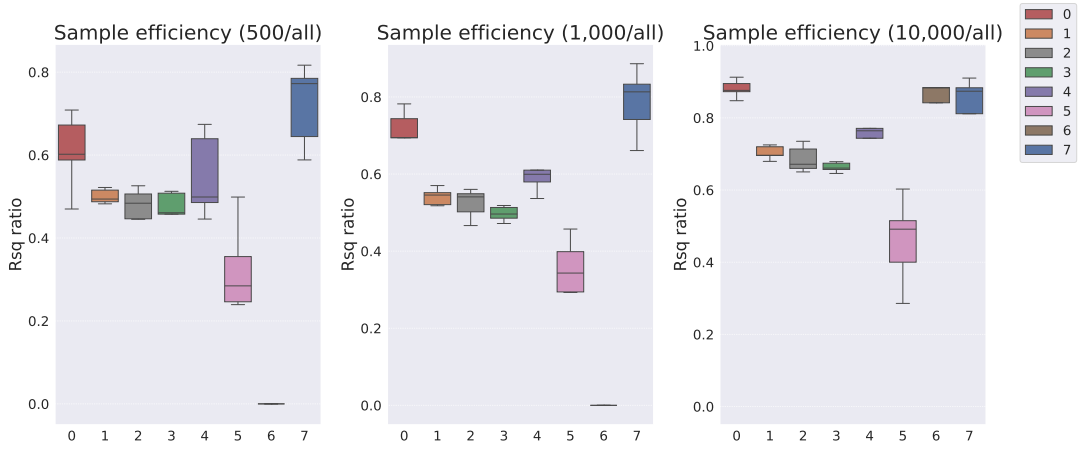

Figure 32: Downstream regression model sample efficiency on the MPI3D dataset (original setting).

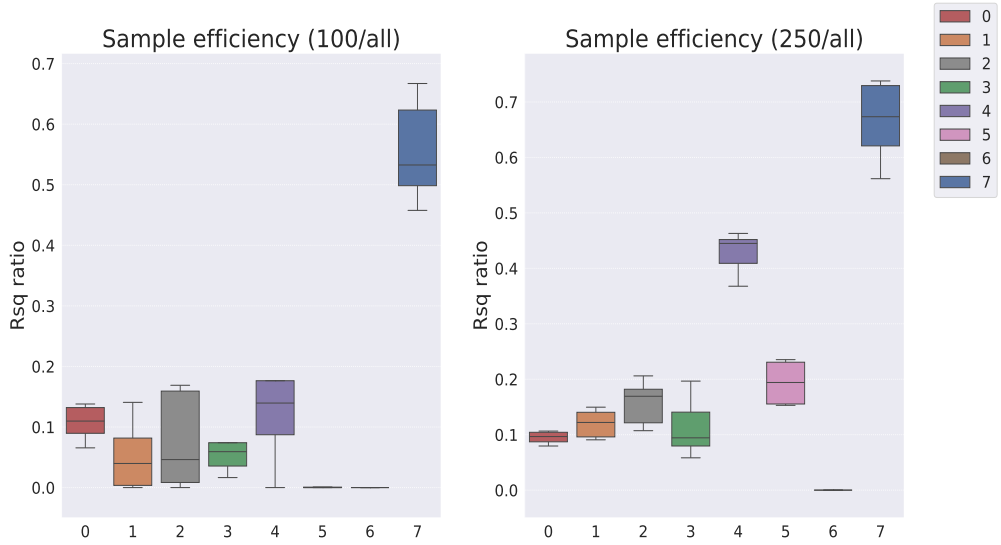

Figure 33: Downstream regression model sample efficiency on the MPI3D dataset (dimensionality-controlled setting).

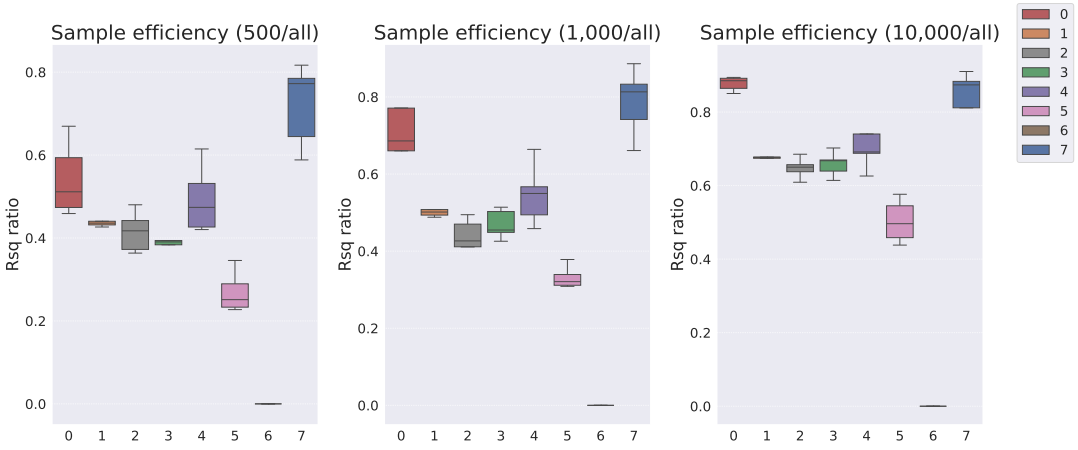

Figure 34: Downstream regression model sample efficiency on the MPI3D dataset (dimensionality-controlled setting).

Table 34: Downstream regression model sample efficiency on the Cars3D dataset

Models	R^2 ratio				
	10^2/all samples	2.5×10^2/all samples	5×10^3/all samples	10^4/all samples	10^5/all samples
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.273 ± 0.052	0.537 ± 0.012	0.833 ± 0.025	0.882 ± 0.018	*0.994 ± 0.004*
Slow-VAE†	0.166 ± 0.055	0.292 ± 0.014	0.866 ± 0.005	0.895 ± 0.010	0.993 ± 0.002
Ada-GVAE-k	0.610 ± 0.131	0.834 ± 0.040	*0.881 ± 0.026*	*0.922 ± 0.019*	0.989 ± 0.005
Ada-GVAE-k†	0.452 ± 0.156	0.821 ± 0.028	0.879 ± 0.029	0.920 ± 0.021	0.992 ± 0.007
GVAE	0.412 ± 0.092	0.791 ± 0.013	0.855 ± 0.012	0.900 ± 0.008	0.986 ± 0.006
GVAE†	0.314 ± 0.066	*0.799 ± 0.010*	0.849 ± 0.014	0.895 ± 0.014	0.987 ± 0.004
MLVAE	0.477 ± 0.069	0.698 ± 0.027	0.844 ± 0.015	0.893 ± 0.016	0.984 ± 0.010
MLVAE†	0.305 ± 0.071	0.648 ± 0.072	0.854 ± 0.013	0.896 ± 0.012	0.989 ± 0.008
Symbolic vector-tokened compositional representations					
VCT	0.558 ± 0.028	0.694 ± 0.034	0.808 ± 0.013	0.837 ± 0.023	0.973 ± 0.024
VCT†	0.197 ± 0.151	0.640 ± 0.113	0.640 ± 0.113	0.725 ± 0.038	**1.000 ± 0.000** (=)
Fully continuous compositional representations					
Ours	**0.705 ± 0.023**	**0.889 ± 0.042**	**0.926 ± 0.009**	**0.942 ± 0.018**	**1.000 ± 0.000** (=)

Table 35: Downstream regression model sample efficiency on the Shapes3D dataset

Models	R^2 ratio				
	10^2/all samples	2.5×10^2/all samples	5×10^3/all samples	10^4/all samples	10^5/all samples
Symbolic scalar-tokened compositional representations					
Slow-VAE	0.021 ± 0.016	0.085 ± 0.035	0.183 ± 0.051	0.563 ± 0.070	0.994 ± 0.002
Slow-VAE†	0.003 ± 0.008	0.021 ± 0.025	0.066 ± 0.036	0.419 ± 0.107	0.995 ± 0.002
Ada-GVAE-k	*0.480 ± 0.155*	0.510 ± 0.160	0.717 ± 0.206	0.855 ± 0.103	0.997 ± 0.001
Ada-GVAE-k†	0.133 ± 0.088	0.199 ± 0.058	0.643 ± 0.231	0.809 ± 0.164	*0.998 ± 0.000* (=)
GVAE	0.417 ± 0.056	0.594 ± 0.079	0.688 ± 0.052	0.893 ± 0.024	0.997 ± 0.000
GVAE†	0.13 ± 0.047	0.234 ± 0.038	0.479 ± 0.076	0.871 ± 0.029	*0.998 ± 0.000* (=)
MLVAE	0.371 ± 0.041	0.515 ± 0.093	0.590 ± 0.080	0.855 ± 0.059	0.997 ± 0.000
MLVAE†	0.123 ± 0.048	0.235 ± 0.069	0.413 ± 0.035	0.808 ± 0.141	**0.999 ± 0.000**
Symbolic vector-tokened compositional representations					
VCT	0.020 ± 0.025	0.216 ± 0.139	0.377 ± 0.100	0.591 ± 0.079	0.884 ± 0.008
VCT†	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.009 ± 0.018	0.470 ± 0.071
COMET	0.352 ± 0.036	0.699 ± 0.038	0.772 ± 0.028	0.812 ± 0.025	0.997 ± 0.001
COMET†	**0.752 ± 0.039**	**0.810 ± 0.034**	**0.870 ± 0.029**	**0.916 ± 0.031**	0.996 ± 0.001
Fully continuous compositional representations					
Ours	0.464 ± 0.071	*0.730 ± 0.038*	*0.804 ± 0.075*	*0.889 ± 0.035*	0.996 ± 0.001

Table 36: Downstream regression model sample efficiency on the MPI3D dataset

Models	R^2 ratio				
	10^2/all samples	2.5×10^2/all samples	5×10^3/all samples	10^4/all samples	10^5/all samples
	Symbolic scalar-tokened compositional representations				
Slow-VAE	0.130 ± 0.051	0.155 ± 0.011	*0.608 ± 0.082*	*0.692 ± 0.081*	*0.881 ± 0.022*
Slow-VAE†	0.107 ± 0.027	0.095 ± 0.011	0.541 ± 0.079	0.668 ± 0.117	0.877 ± 0.017
Ada-GVAE-k	*0.270 ± 0.037*	0.279 ± 0.026	0.500 ± 0.016	0.541 ± 0.020	0.703 ± 0.017
Ada-GVAE-k†	0.053 ± 0.053	0.120 ± 0.023	0.442 ± 0.018	0.504 ± 0.015	0.680 ± 0.012
GVAE	0.234 ± 0.035	0.282 ± 0.027	0.481 ± 0.032	0.524 ± 0.035	0.686 ± 0.033
GVAE†	0.077 ± 0.073	0.157 ± 0.037	0.415 ± 0.043	0.443 ± 0.034	0.648 ± 0.025
MLVAE	0.236 ± 0.019	0.288 ± 0.030	0.462 ± 0.051	0.497 ± 0.017	0.663 ± 0.012
MLVAE†	0.065 ± 0.042	0.114 ± 0.049	0.387 ± 0.024	0.469 ± 0.034	0.659 ± 0.030
Shu	0.343 ± 0.024	*0.482 ± 0.075*	0.549 ± 0.091	0.601 ± 0.047	0.750 ± 0.058
Shu†	0.143 ± 0.103	0.427 ± 0.035	0.493 ± 0.073	0.547 ± 0.070	0.714 ± 0.067
	Symbolic vector-tokened compositional representations				
VCT	0.189 ± 0.107	0.246 ± 0.137	0.294 ± 0.145	0.312 ± 0.14	0.418 ± 0.180
VCT†	0.039 ± 0.088	0.168 ± 0.082	0.230 ± 0.110	0.279 ± 0.127	0.502 ± 0.052
COMET	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.823 ± 0.139
COMET†	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.187 ± 0.374
	Fully continuous compositional representations				
Ours	**0.556 ± 0.078**	**0.665 ± 0.067**	**0.721 ± 0.089**	**0.787 ± 0.078**	**0.858 ± 0.040**

C.5.2 Low Sample Regime Results

To evaluate the utility of our Soft TPR representation from the perspective of downstream models, we additionally evaluate the raw performance of downstream models as a function of the number of samples they have been trained on (again considering 100, 250, 500, 1,000, 10,000 and all samples). We find that our Soft TPR representation contributes to a substantial performance boost in the downstream model's performance in a low-sample regime where the downstream model has been trained with 100, 250, 500, and 1,000 samples. We present our full suite of experimental results, and highlight the particular performance differentials conferred by our representational form in the low-sample regime.

FoV Regression

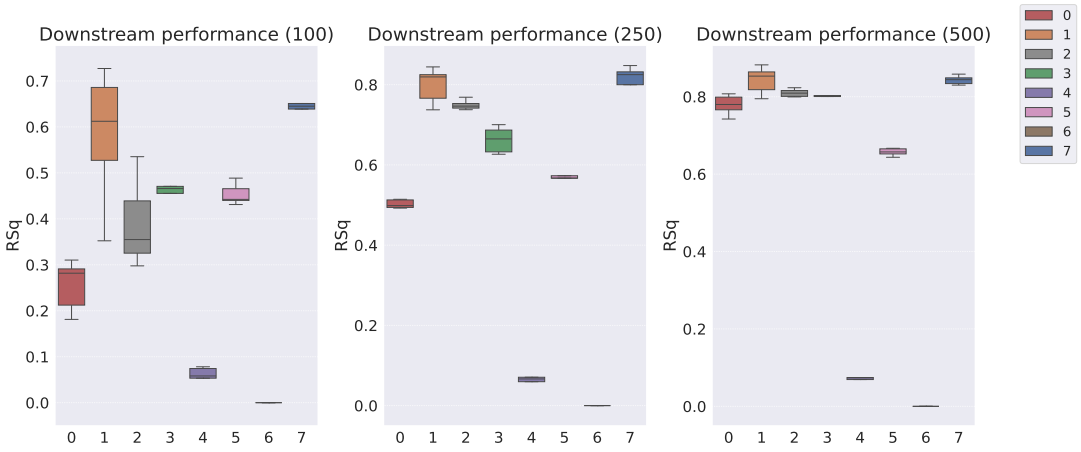

Figure 35: Downstream regression model R^2 scores on the Cars3D dataset (original setting).

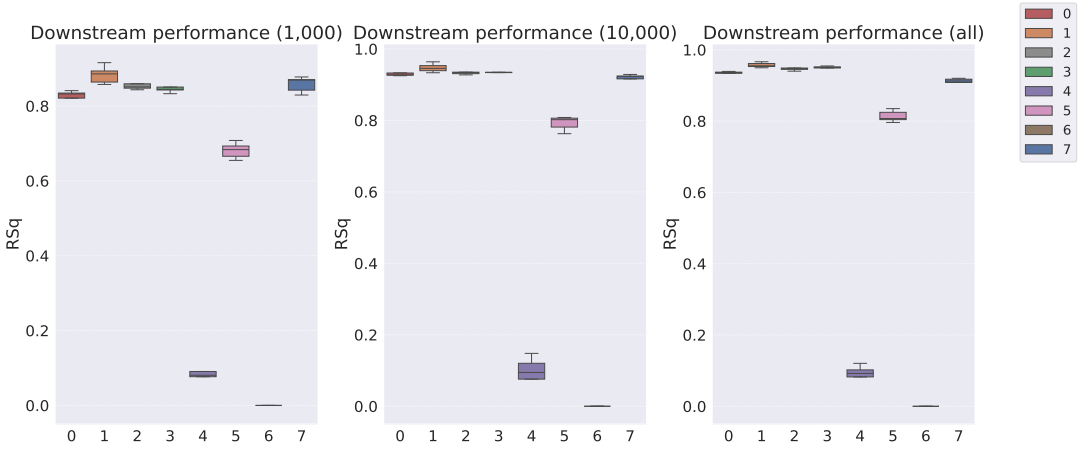

Figure 36: Downstream regression model R^2 scores on the Cars3D dataset (original setting).

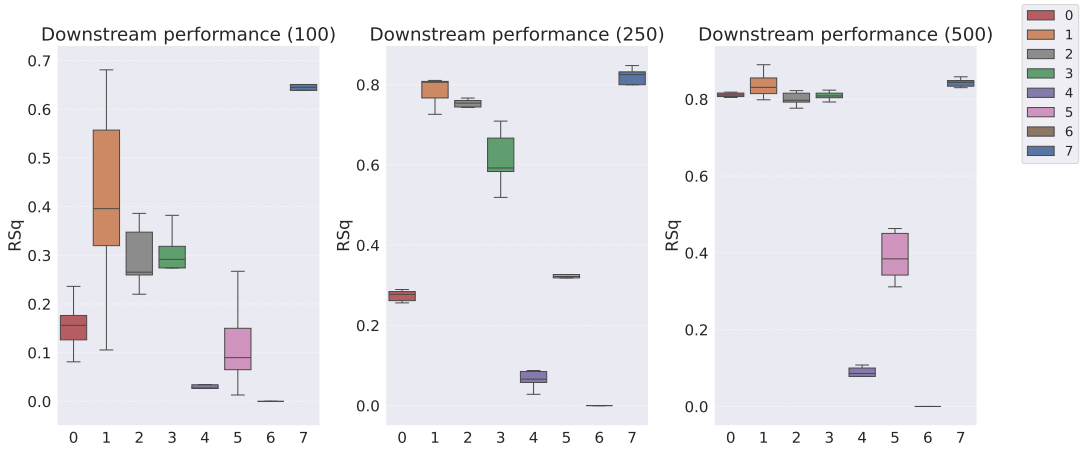

Figure 37: Downstream regression model R^2 scores on the Cars3D dataset (dimensionality-controlled setting).

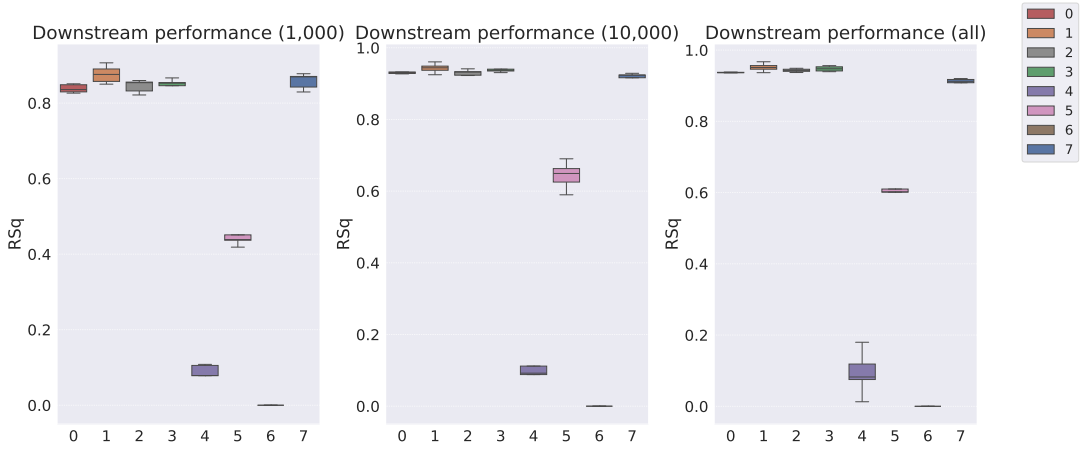

Figure 38: Downstream regression model R^2 scores on the Cars3D dataset (dimensionality-controlled setting).

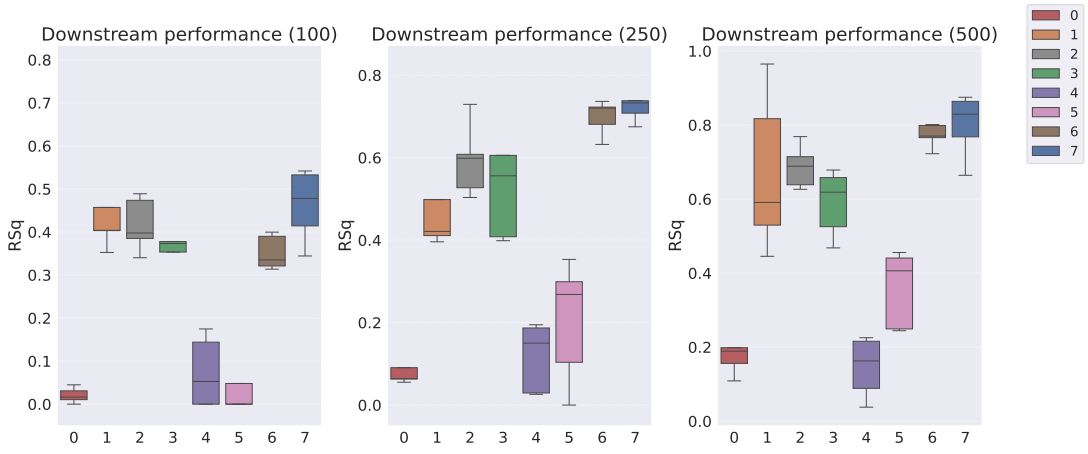

Figure 39: Downstream regression model R^2 scores on the Shapes3D dataset (original setting).

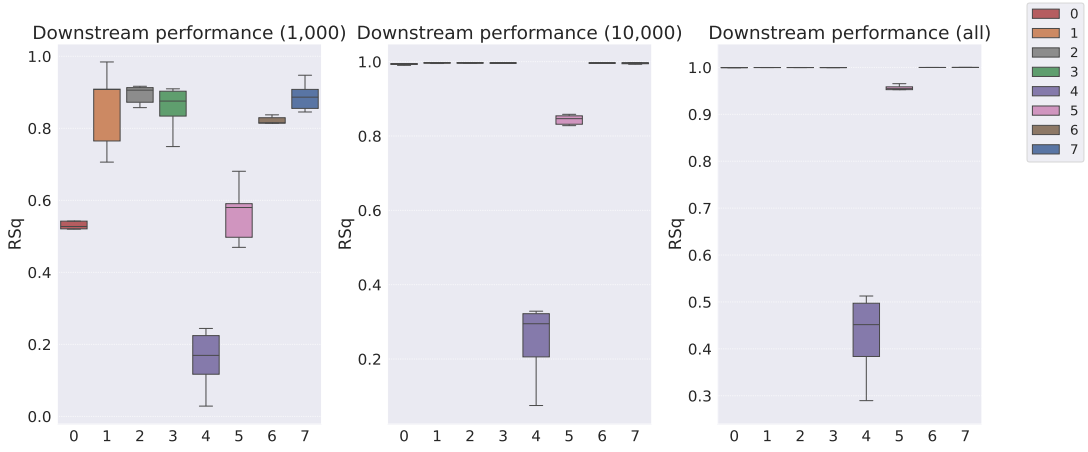

Figure 40: Downstream regression model R^2 scores on the Shapes3D dataset (original setting).

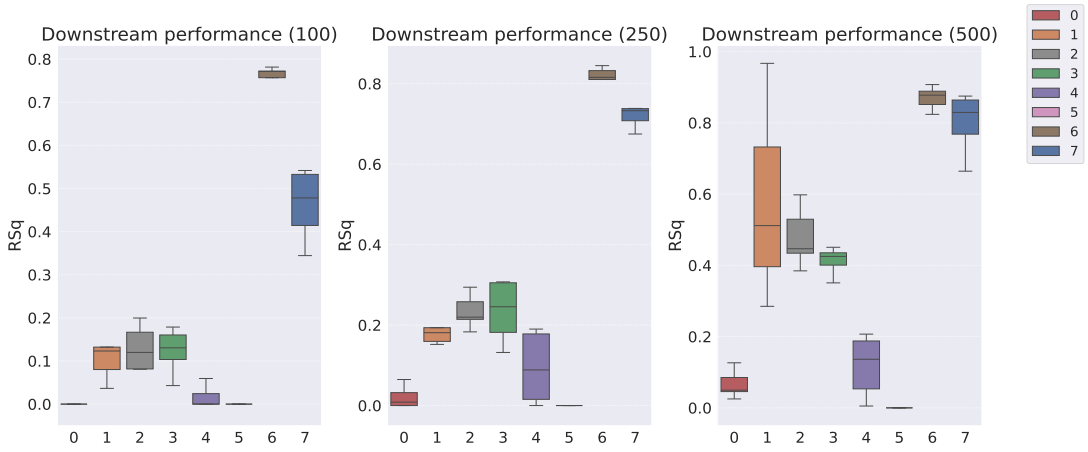

Figure 41: Downstream regression model R^2 scores on the Shapes3D dataset (dimensionality-controlled setting).

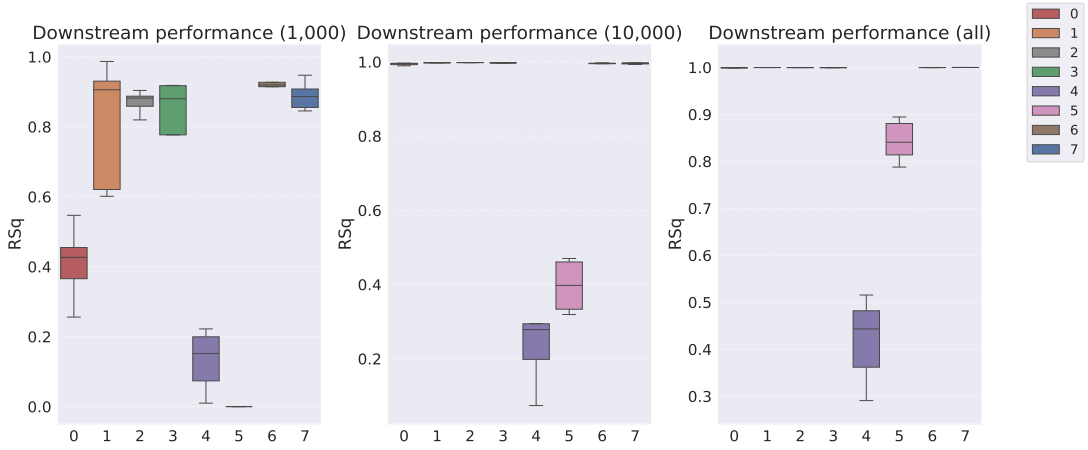

Figure 42: Downstream regression model R^2 scores on the Shapes3D dataset (dimensionality-controlled setting).

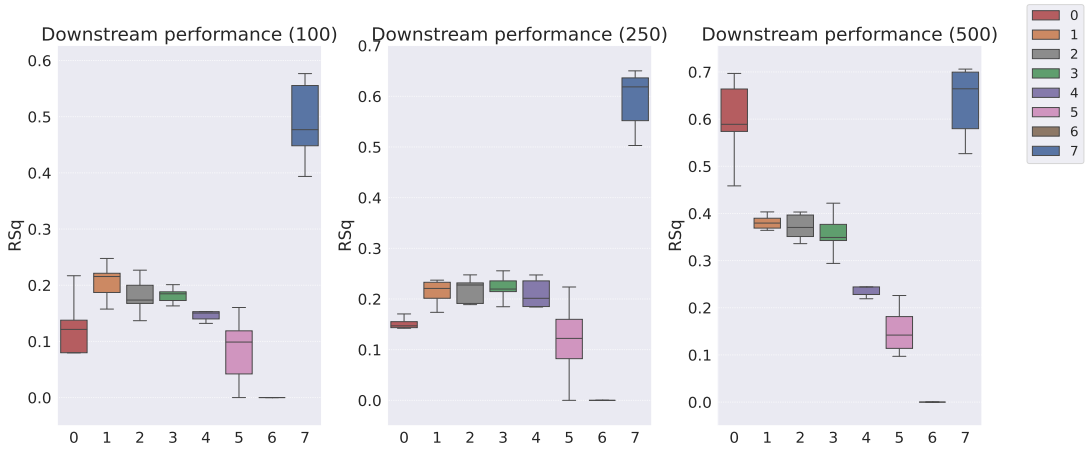

Figure 43: Downstream regression model R^2 scores on the MPI3D dataset (original setting).

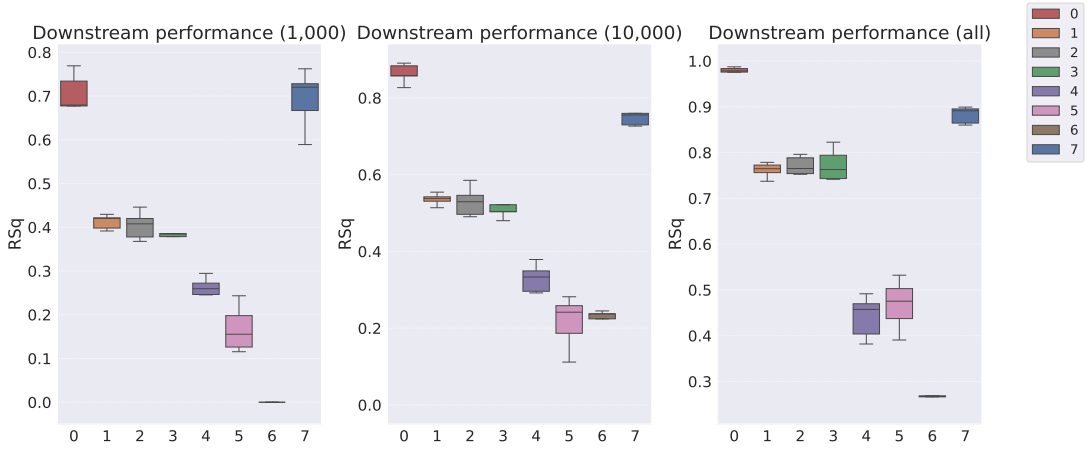

Figure 44: Downstream regression model R^2 scores on the MPI3D dataset (original setting).

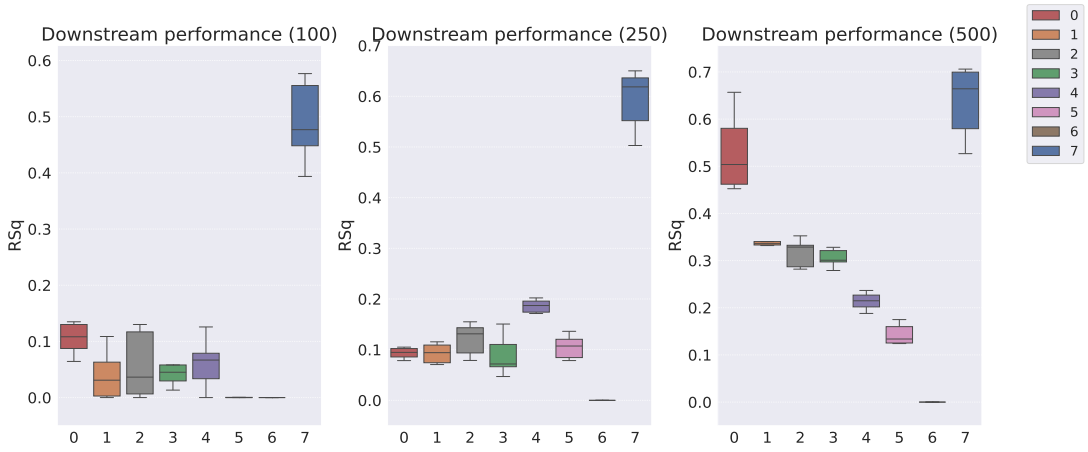

Figure 45: Downstream regression model R^2 scores on the MPI3D dataset (dimensionality-controlled setting).

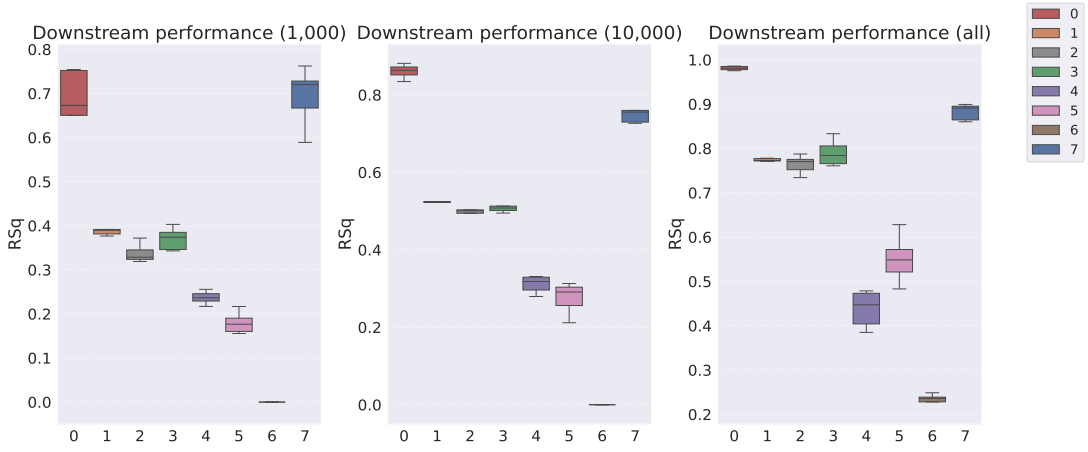

Figure 46: Downstream regression model R^2 scores on the MPI3D dataset (dimensionality-controlled setting).

Table 37: Downstream regression model performance on the Cars3D dataset

Models	R^2 score					
	10^2 samples	2.5×10^2 samples	5×10^3 samples	10^4 samples	10^5 samples	all samples
Symbolic scalar-tokened compositional representations						
Slow-VAE	0.255 ± 0.05	0.502 ± 0.009	0.779 ± 0.023	0.825 ± 0.016	0.929 ± 0.003	0.935 ± 0.004
Slow-VAE†	0.155 ± 0.052	0.274 ± 0.013	0.811 ± 0.005	0.838 ± 0.010	0.930 ± 0.002	0.935 ± 0.004
Ada-GVAE-k	*0.584 ± 0.125*	*0.798 ± 0.040*	*0.843 ± 0.028*	**0.883 ± 0.020**	**0.947 ± 0.009**	**0.957 ± 0.005** (=)
Ada-GVAE-k†	0.430 ± 0.149	0.784 ± 0.033	0.836 ± 0.028	*0.875 ± 0.018*	*0.944 ± 0.010*	**0.957 ± 0.005** (=)
GVAE	0.390 ± 0.087	0.750 ± 0.011	0.810 ± 0.009	0.852 ± 0.006	0.934 ± 0.006	*0.947 ± 0.005* (=)
GVAE†	0.296 ± 0.061	0.754 ± 0.009	0.801 ± 0.016	0.845 ± 0.015	0.931 ± 0.007	*0.947 ± 0.005* (=)
MLVAE	0.452 ± 0.063	0.662 ± 0.029	0.800 ± 0.018	0.847 ± 0.010	0.933 ± 0.007	0.948 ± 0.007
MLVAE†	0.289 ± 0.066	0.614 ± 0.067	0.809 ± 0.011	0.849 ± 0.014	0.938 ± 0.006	0.948 ± 0.007
Shu	0.044 ± 0.045	0.055 ± 0.044	0.061 ± 0.047	0.067 ± 0.050	0.085 ± 0.056	0.075 ± 0.051
Shu†	0.010 ± 0.042	0.065 ± 0.022	0.079 ± 0.031	0.079 ± 0.030	0.101 ± 0.046	0.075 ± 0.051
Symbolic vector-tokened compositional representations						
VCT	0.454 ± 0.021	0.565 ± 0.033	0.657 ± 0.009	0.681 ± 0.019	0.792 ± 0.018	0.813 ± 0.014
VCT†	0.117 ± 0.087	0.340 ± 0.073	0.390 ± 0.059	0.444 ± 0.019	0.643 ± 0.034	0.813 ± 0.014
COMET	-0.058 ± 0.027	-0.098 ± 0.054	-0.074 ± 0.036	-0.147 ± 0.101	-0.036 ± 0.028	-0.035 ± 0.023
COMET†	-0.379 ± 0.303	-0.258 ± 0.085	-0.966 ± 0.470	-0.306 ± 0.103	-0.739 ± 0.751	-0.035 ± 0.023
Fully continuous compositional representations						
Ours	**0.642 ± 0.023**	**0.809 ± 0.037**	**0.843 ± 0.010**	0.858 ± 0.019	0.917 ± 0.013	0.910 ± 0.010

Table 38: Downstream regression model performance on the Shapes3D dataset

Models	R^2 score					
	10^2 samples	2.5×10^2 samples	5×10^3 samples	10^4 samples	10^5 samples	all samples
Symbolic scalar-tokened compositional representations						
Slow-VAE	0.019 ± 0.017	0.085 ± 0.035	0.183 ± 0.051	0.562 ± 0.070	0.993 ± 0.002	**1.000 ± 0.000** (=)
Slow-VAE†	-0.030 ± 0.034	0.018 ± 0.028	0.066 ± 0.036	0.418 ± 0.107	0.994 ± 0.003	*0.999 ± 0.000*
Ada-GVAE-k	0.480 ± 0.155	0.510 ± 0.160	0.666 ± 0.170	0.854 ± 0.103	*0.997 ± 0.001* (=)	**1.000 ± 0.000** (=)
Ada-GVAE-k†	0.133 ± 0.088	0.199 ± 0.058	0.563 ± 0.209	0.809 ± 0.164	*0.997 ± 0.001* (=)	**1.000 ± 0.000** (=)
GVAE	0.417 ± 0.056	0.593 ± 0.079	0.687 ± 0.052	*0.893 ± 0.024*	*0.997 ± 0.000* (=)	**1.000 ± 0.000** (=)
GVAE†	0.130 ± 0.047	0.234 ± 0.038	0.478 ± 0.076	0.870 ± 0.029	**0.998 ± 0.000**	**1.000 ± 0.000** (=)
MLVAE	0.370 ± 0.041	0.515 ± 0.093	0.590 ± 0.080	0.854 ± 0.059	0.996 ± 0.000	**1.000 ± 0.000** (=)
MLVAE†	0.123 ± 0.048	0.234 ± 0.069	0.412 ± 0.035	0.807 ± 0.141	*0.997 ± 0.000* (=)	**1.000 ± 0.000** (=)
Shu	0.062 ± 0.086	0.117 ± 0.075	0.146 ± 0.073	0.157 ± 0.078	0.245 ± 0.096	0.427 ± 0.082
Shu†	-0.029 ± 0.076	0.093 ± 0.08	0.118 ± 0.077	0.131 ± 0.079	0.228 ± 0.085	0.419 ± 0.082
Symbolic vector-tokened compositional representations						
VCT	-0.201 ± 0.289	0.202 ± 0.136	0.359 ± 0.093	0.564 ± 0.075	0.844 ± 0.012	0.954 ± 0.008
VCT†	-4.183 ± 1.679	-1.464 ± 0.492	-0.552 ± 0.307	-0.205 ± 0.134	0.396 ± 0.063	0.843 ± 0.040
COMET	0.352 ± 0.036	0.699 ± 0.038	0.772 ± 0.029	0.812 ± 0.025	0.996 ± 0.001	**1.000 ± 0.000** (=)
COMET†	**0.751 ± 0.039**	**0.870 ± 0.029**	**0.996 ± 0.001**	**0.915 ± 0.031**	0.996 ± 0.001	**1.000 ± 0.000** (=)
Fully continuous compositional representations						
Ours	*0.464 ± 0.071*	*0.730 ± 0.038*	*0.804 ± 0.075*	0.889 ± 0.035	0.995 ± 0.002	**1.000 ± 0.000** (=)

Table 39: Downstream regression model performance on the MPI3D dataset

Models	R^2 score					
	10^2 samples	2.5×10^2 samples	5×10^3 samples	10^4 samples	10^5 samples	all samples
Symbolic scalar-tokened compositional representations						
Slow-VAE	0.127 ± 0.059	0.152 ± 0.011	*0.596 ± 0.083*	*0.678 ± 0.082*	**0.863 ± 0.023**	*0.980 ± 0.005*
Slow-VAE†	0.105 ± 0.026	0.093 ± 0.011	0.531 ± 0.077	0.655 ± 0.113	*0.860 ± 0.016*	**0.981 ± 0.004**
Ada-GVAE-k	*0.206 ± 0.031*	0.213 ± 0.023	0.381 ± 0.014	0.412 ± 0.015	0.536 ± 0.013	0.762 ± 0.014
Ada-GVAE-k†	0.037 ± 0.045	0.093 ± 0.018	0.342 ± 0.014	0.390 ± 0.012	0.527 ± 0.010	0.774 ± 0.003
GVAE	0.181 ± 0.030	0.217 ± 0.023	0.371 ± 0.026	0.404 ± 0.028	0.530 ± 0.035	0.771 ± 0.018
GVAE†	0.056 ± 0.057	0.120 ± 0.029	0.316 ± 0.027	0.338 ± 0.019	0.495 ± 0.020	0.764 ± 0.019
MLVAE	0.182 ± 0.013	*0.222 ± 0.024*	0.357 ± 0.042	0.384 ± 0.023	0.513 ± 0.025	0.773 ± 0.031
MLVAE†	0.051 ± 0.032	0.089 ± 0.037	0.305 ± 0.018	0.370 ± 0.023	0.520 ± 0.033	0.790 ± 0.027
Shu	0.151 ± 0.016	0.211 ± 0.026	0.238 ± 0.019	0.264 ± 0.018	0.330 ± 0.033	0.441 ± 0.041
Shu†	0.057 ± 0.048	0.186 ± 0.012	0.214 ± 0.017	0.237 ± 0.013	0.311 ± 0.020	0.438 ± 0.037
Symbolic vector-tokened compositional representations						
VCT	0.047 ± 0.139	0.110 ± 0.087	0.124 ± 0.104	0.105 ± 0.175	0.164 ± 0.183	0.455 ± 0.071
VCT†	-0.005 ± 0.058	0.084 ± 0.062	0.106 ± 0.092	0.153 ± 0.072	0.276 ± 0.036	0.550 ± 0.046
COMET	-0.051 ± 0.015	-0.042 ± 0.018	-0.037 ± 0.012	-0.053 ± 0.024	0.218 ± 0.036	0.266 ± 0.004
COMET†	-0.118 ± 0.045	-0.089 ± 0.050	-0.074 ± 0.039	-0.179 ± 0.098	0.027 ± 0.099	0.236 ± 0.008
Fully continuous compositional representations						
Ours	**0.490 ± 0.06**8	**0.594 ± 0.056**	**0.635 ± 0.070**	**0.693 ± 0.060**	0.757 ± 0.03	0.882 ± 0.016

b) Abstract Visual Reasoning

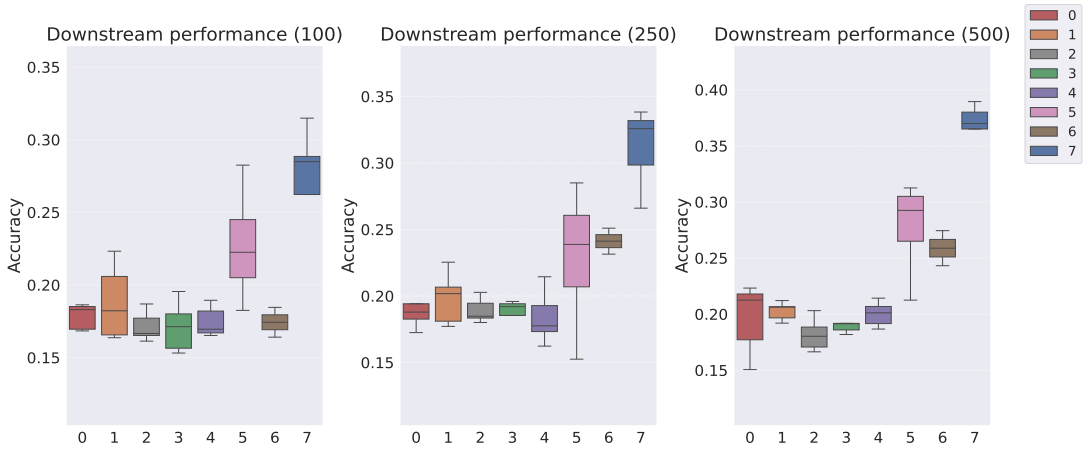

Figure 47: Downstream WReN model classification accuracy on the abstract visual reasoning dataset (original setting).

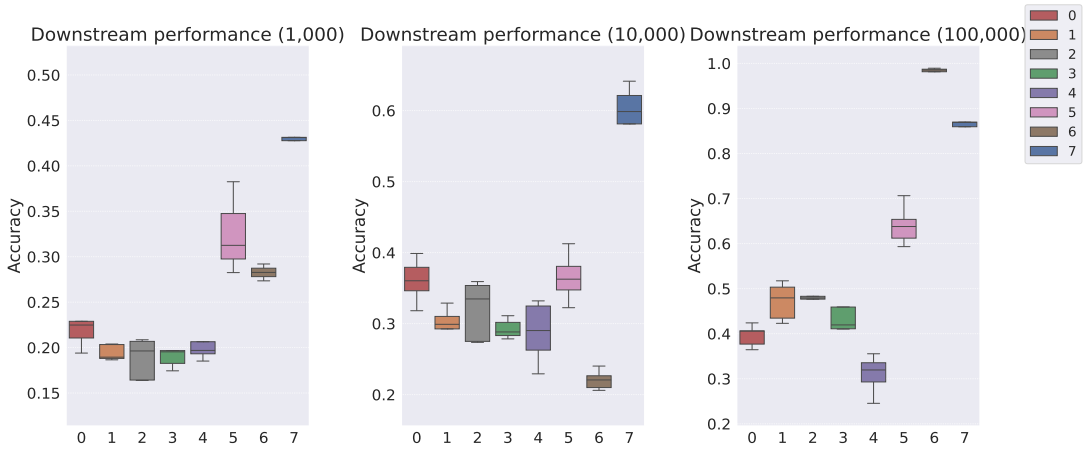

Figure 48: Downstream WReN model classification accuracy on the abstract visual reasoning dataset (original setting).

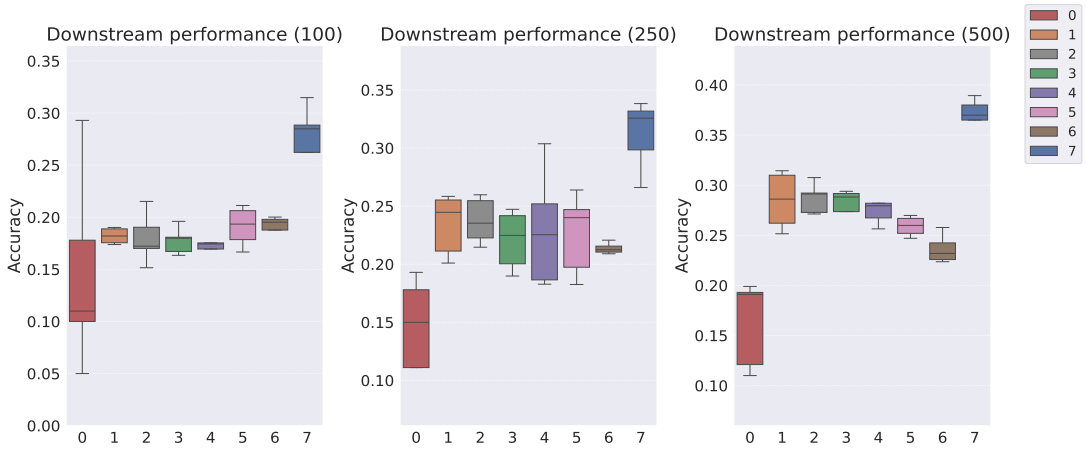

Figure 49: Downstream WReN model classification accuracy on the abstract visual reasoning dataset (dimensionality-controlled setting).

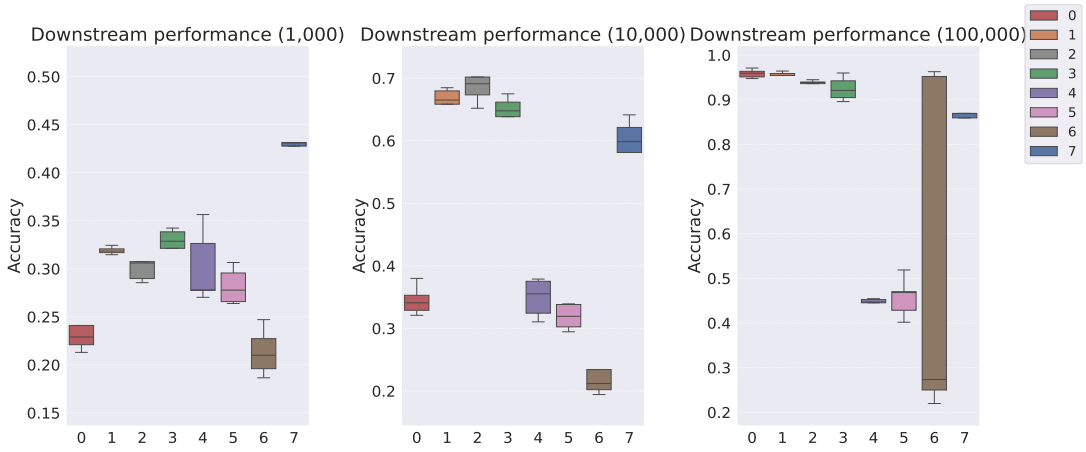

Figure 50: Downstream WReN model classification accuracy on the abstract visual reasoning dataset (dimensionality-controlled setting).

Table 40: Downstream WReN model performance on the abstract visual reasoning dataset

Models	Classification accuracy					
	10^2 samples	2.5×10^2 samples	5×10^2 samples	10^3 samples	10^4 samples	10^5 samples
Symbolic scalar-tokened compositional representations						
Slow-VAE	0.178 ± 0.008	0.196 ± 0.024	0.196 ± 0.028	0.228 ± 0.030	0.361 ± 0.028	0.396 ± 0.022
Slow-VAE†	0.146 ± 0.084	0.149 ± 0.034	0.163 ± 0.039	0.237 ± 0.023	0.345 ± 0.021	*0.959 ± 0.009*
Ada-GVAE-k	0.188 ± 0.023	0.198 ± 0.018	0.203 ± 0.007	0.194 ± 0.008	0.295 ± 0.028	0.472 ± 0.037
Ada-GVAE-k†	0.182 ± 0.007	*0.234 ± 0.024*	*0.285 ± 0.025*	0.319 ± 0.004	*0.662 ± 0.023*	0.954 ± 0.009
GVAE	0.171 ± 0.009	0.189 ± 0.008	0.182 ± 0.013	0.188 ± 0.020	0.319 ± 0.038	0.478 ± 0.032
GVAE†	0.180 ± 0.022	0.237 ± 0.018	0.287 ± 0.014	0.306 ± 0.020	**0.684 ± 0.020**	0.936 ± 0.008
MLVAE	0.171 ± 0.016	0.188 ± 0.009	0.193 ± 0.012	0.194 ± 0.015	0.293 ± 0.012	0.432 ± 0.023
MLVAE†	0.177 ± 0.012	0.221 ± 0.023	0.277 ± 0.023	0.325 ± 0.017	0.644 ± 0.026	0.925 ± 0.024
Shu	0.175 ± 0.009	0.184 ± 0.018	0.200 ± 0.010	0.202 ± 0.014	0.288 ± 0.038	0.310 ± 0.038
Shu†	0.177 ± 0.014	0.230 ± 0.045	0.285 ± 0.028	0.302 ± 0.034	0.444 ± 0.013	0.444 ± 0.013
Symbolic vector-tokened compositional representations						
VCT	*0.228 ± 0.036*	0.229 ± 0.049	0.277 ± 0.039	*0.326 ± 0.042*	0.371 ± 0.040	0.641 ± 0.039
VCT†	0.191 ± 0.017	0.226 ± 0.031	0.259 ± 0.009	0.282 ± 0.017	0.319 ± 0.018	0.458 ± 0.040
COMET	0.174 ± 0.010	0.241 ± 0.010	0.259 ± 0.016	0.283 ± 0.009	0.221 ± 0.012	**0.983 ± 0.006**
COMET†	0.190 ± 0.012	0.214 ± 0.004	0.236 ± 0.013	0.213 ± 0.023	0.532 ± 0.348	0.532 ± 0.348
Fully continuous compositional representations						
Ours	**0.273 ± 0.033**	**0.312 ± 0.027**	**0.360 ± 0.033**	**0.412 ± 0.066**	0.560 ± 0.103	0.869 ± 0.024

C.6 More Ablation Experiments

C.6.1 Soft TPR vs TPR

To verify that our Soft TPR confers exclusive benefits compared to TPRs, we repeat all our experiments using the explicit TPR that our TPR decoder produces, which represents the Soft TPR's greedily optimal explicit TPR counterpart, ψ^*_{tpr}. In all plots, we use the same legend, denoting the explicit TPR as yellow (0) and the Soft TPR as blue (1). Across all considered cases: i.e., 1) convergence rate of representation learning (as measured by the downstream model's ability to effectively leverage representations produced at different stages of training), 2) sample efficiency of downstream models, and 3) raw performance of downstream models in the low sample regime, the Soft TPR confers differential performance boosts compared to the explicit TPR. This offers strong empirical evidence for our hypothesis that embodying a more approximate form of explicitly compositional structure helps representation learners by alleviating the strigent requirement of having to produce explicit TPRs, and additionally provides representation learners with more knowledge of the entire manifold underlying compositional structure, which TPRs cannot fully capture either due to representation learner deficits, or the inherent quasi-compositional structure of represented objects.

Convergence Rate of Representation Learning

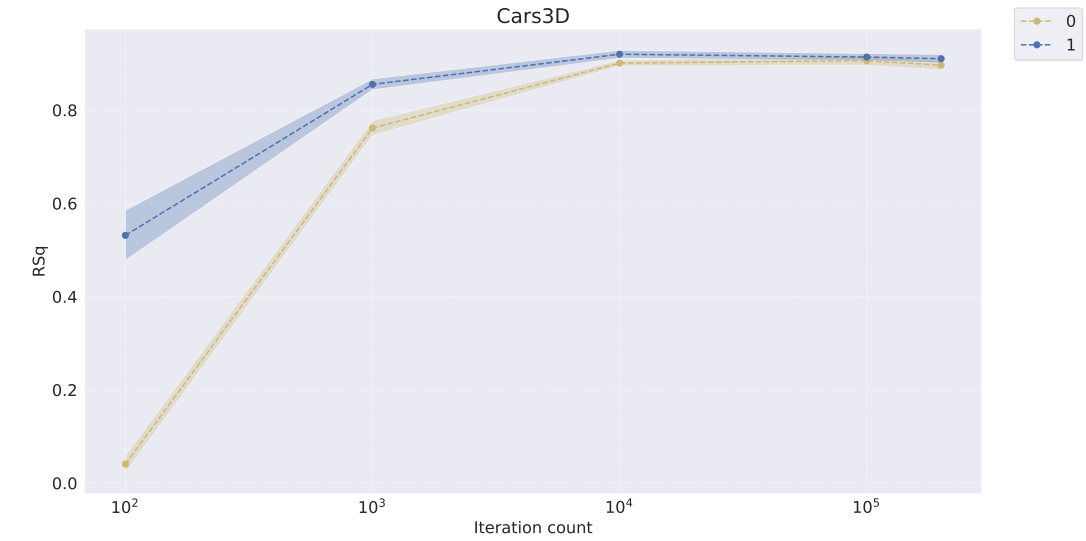

Figure 51: Convergence of Soft TPR (0) vs TPR (1) as measured by FoV regression on the Cars3D dataset

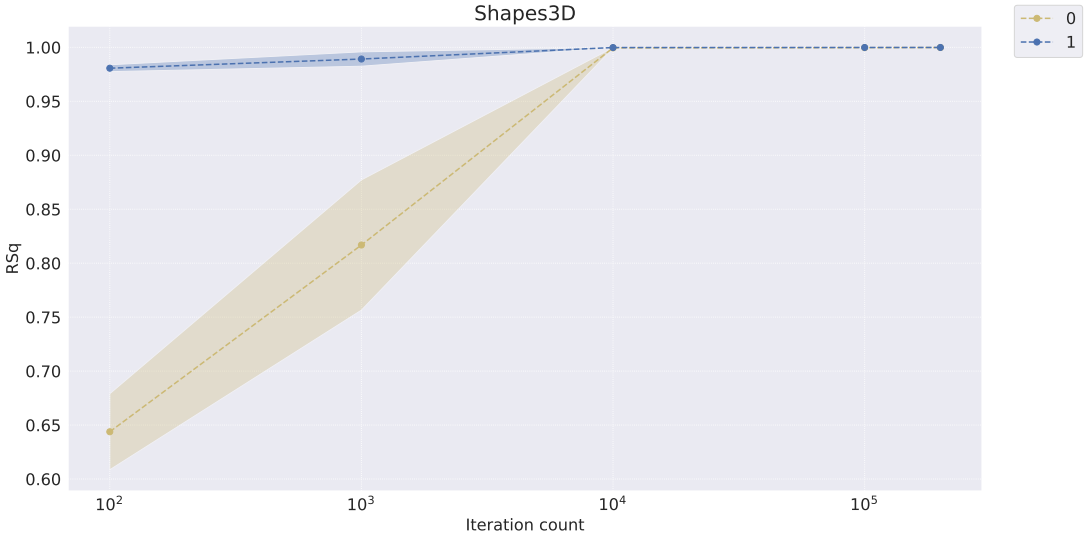

Figure 52: Convergence of Soft TPR (0) vs TPR (1) as measured by FoV regression on the Shapes3D dataset

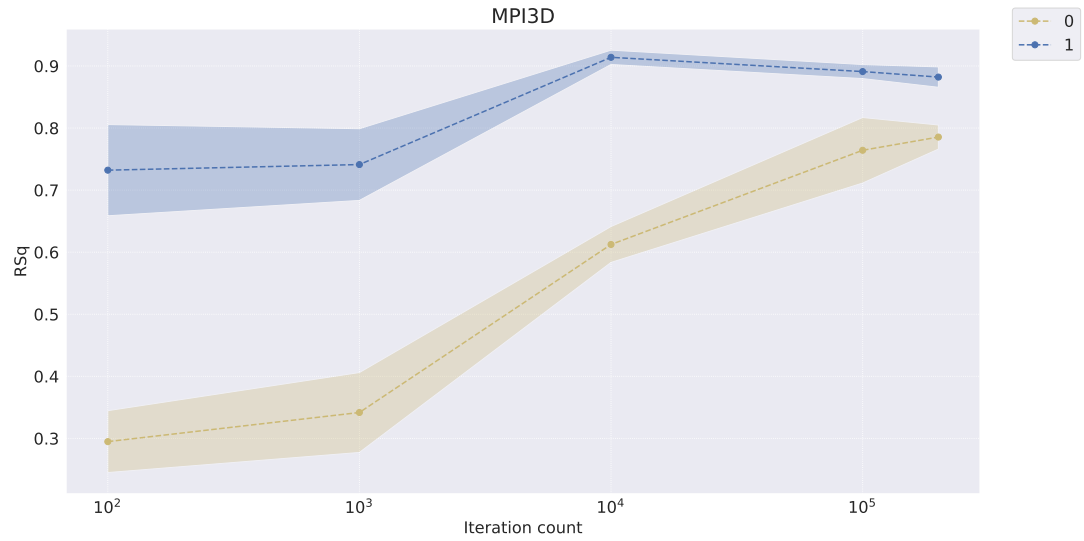

Figure 53: Convergence of Soft TPR (0) vs TPR (1) as measured by FoV regression on the MPI3D dataset

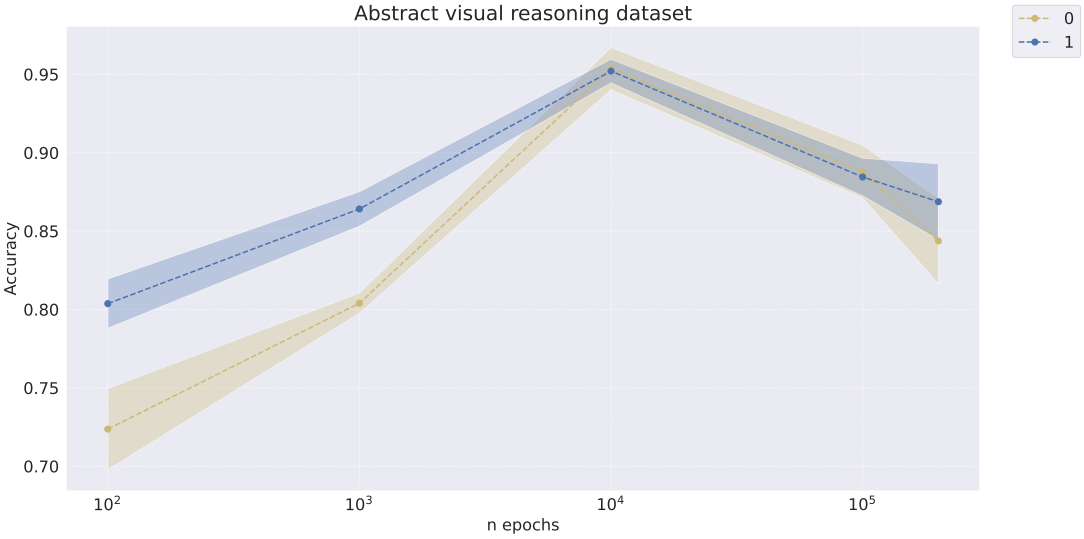

Figure 54: Convergence of Soft TPR (0) vs TPR (1) as measured by classification performance on the abstract visual reasoning dataset

Sample Efficiency of Downstream Models

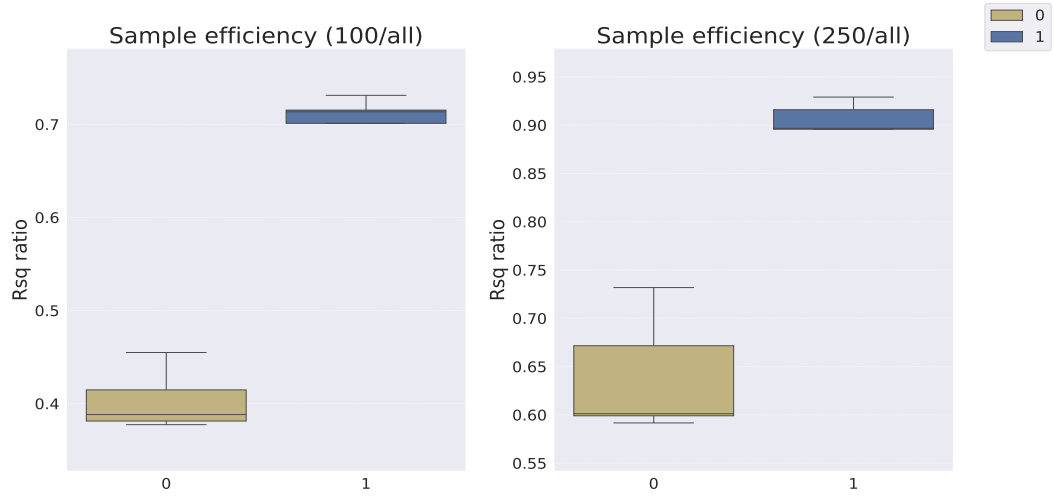

Figure 55: Downstream regression model sample efficiency on the Cars3D dataset using TPR representations (0) vs Soft TPR representations (1)

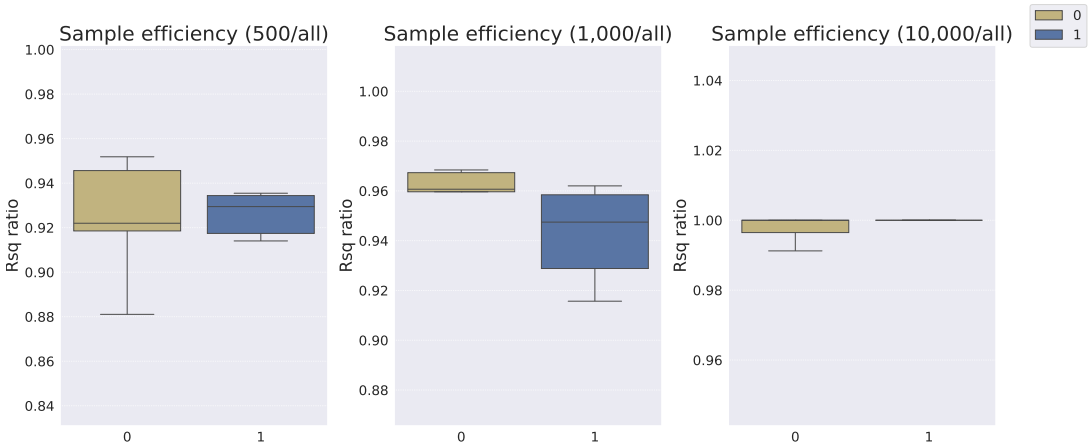

Figure 56: Downstream regression model sample efficiency on the Cars3D dataset using TPR representations (0) vs Soft TPR representations (1)

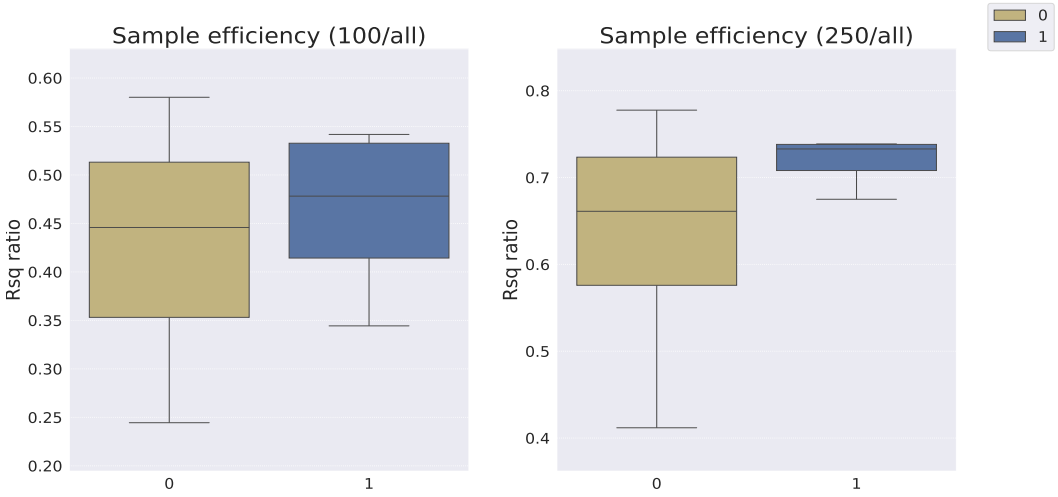

Figure 57: Downstream regression model sample efficiency on the Shapes3D dataset using TPR representations (0) vs Soft TPR representations (1)

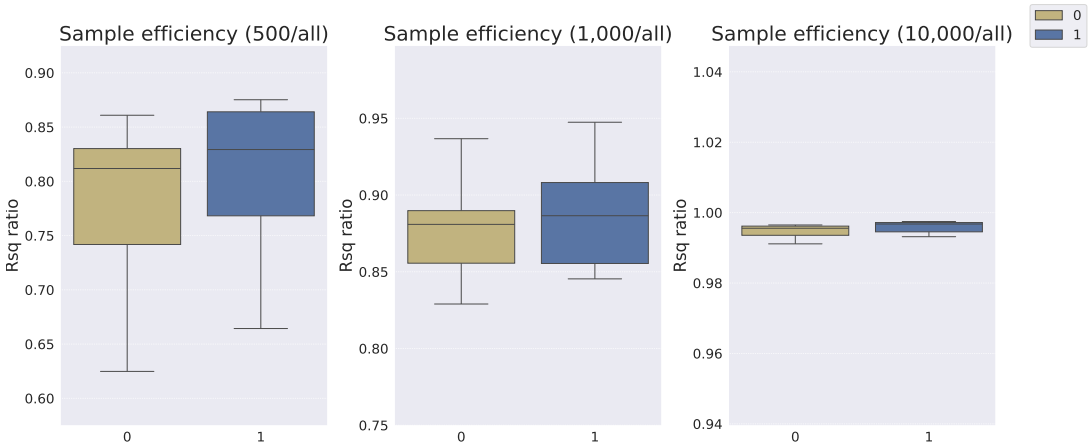

Figure 58: Downstream regression model sample efficiency on the Shapes3D dataset using TPR representations (0) vs Soft TPR representations (1)

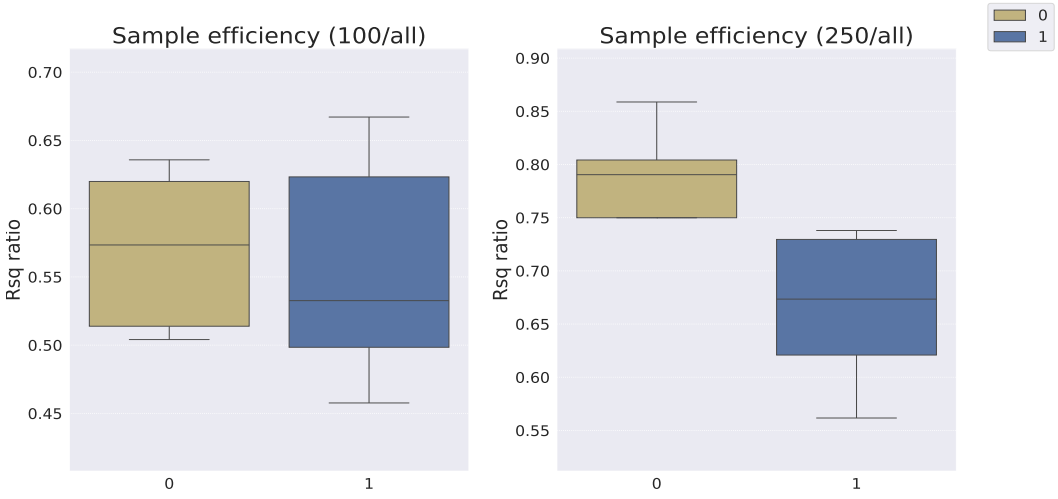

Figure 59: Downstream regression model sample efficiency on the MPI3D dataset using TPR representations (0) vs Soft TPR representations (1)

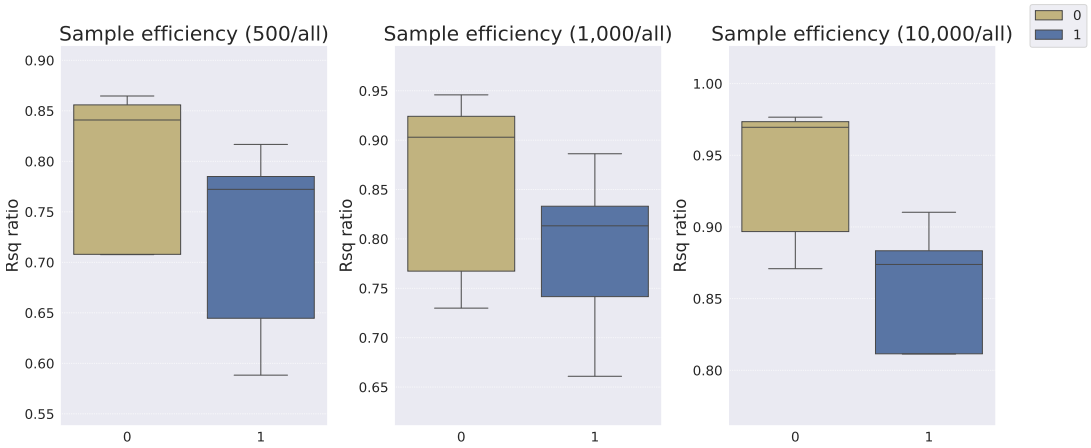

Figure 60: Downstream regression model sample efficiency on the MPI3D dataset using TPR representations (0) vs Soft TPR representations (1)

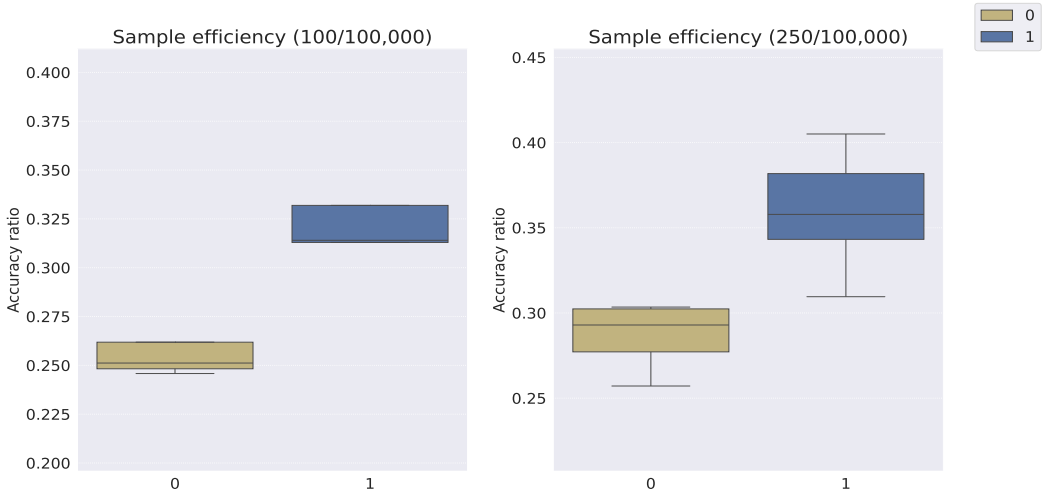

Figure 61: Downstream WReN model sample efficiency on the abstract visual reasoning dataset using TPR representations (0) vs Soft TPR representations (1)

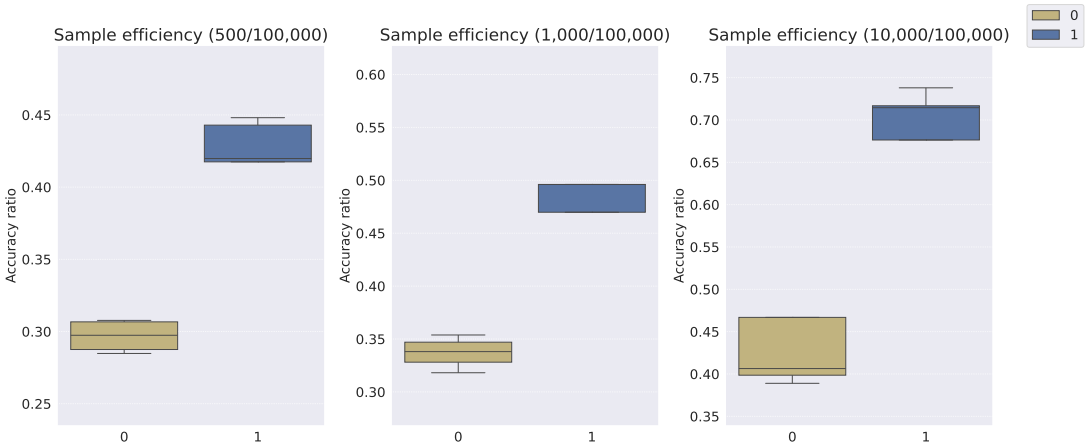

Figure 62: Downstream WReN model sample efficiency on the abstract visual reasoning dataset using TPR representations (0) vs Soft TPR representations (1)

Low Sample-Regime Performance of Downstream Models

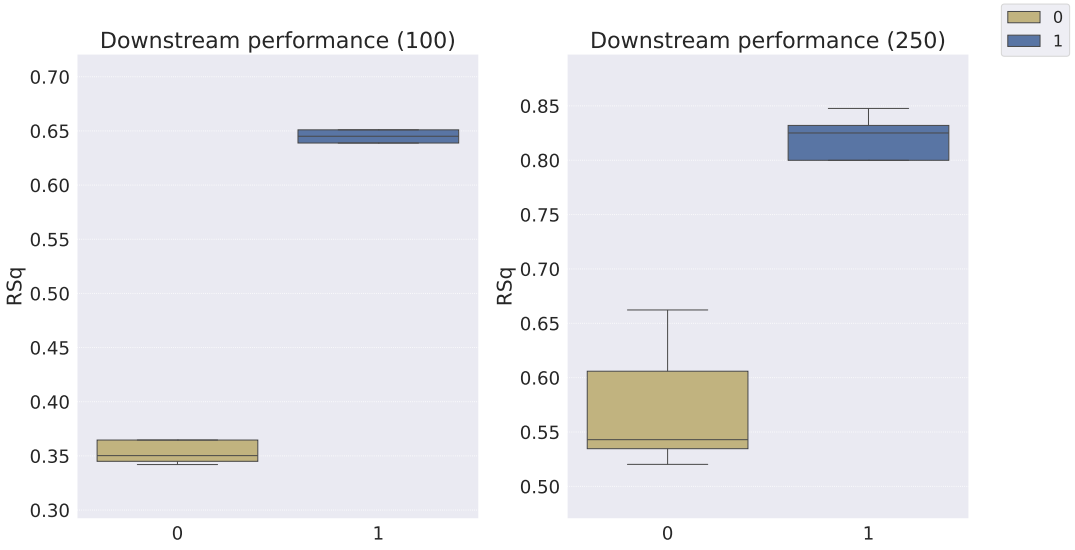

Figure 63: Downstream regression model R^2 scores on the Cars3D dataset using TPR representations (0) vs Soft TPR representations (1)

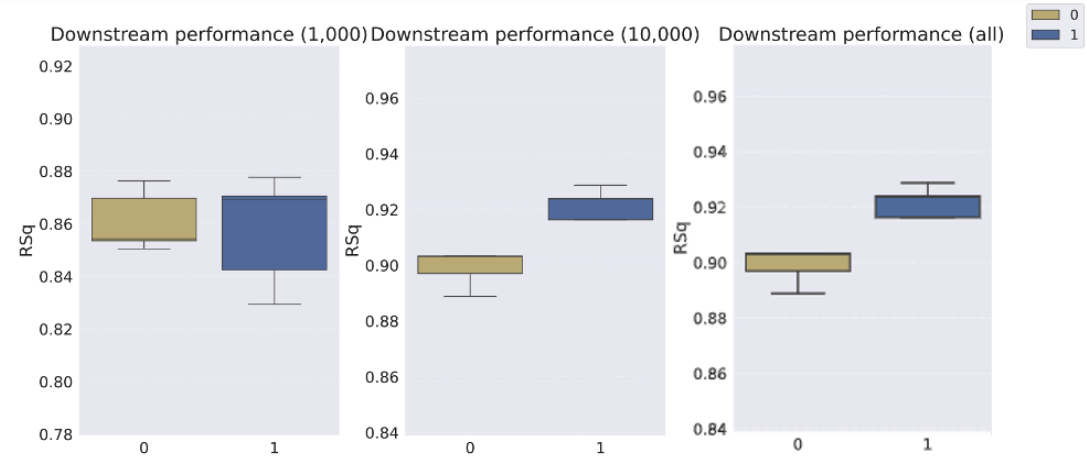

Figure 64: Downstream regression model R^2 scores on the Cars3D dataset using TPR representations (0) vs Soft TPR representations (1)

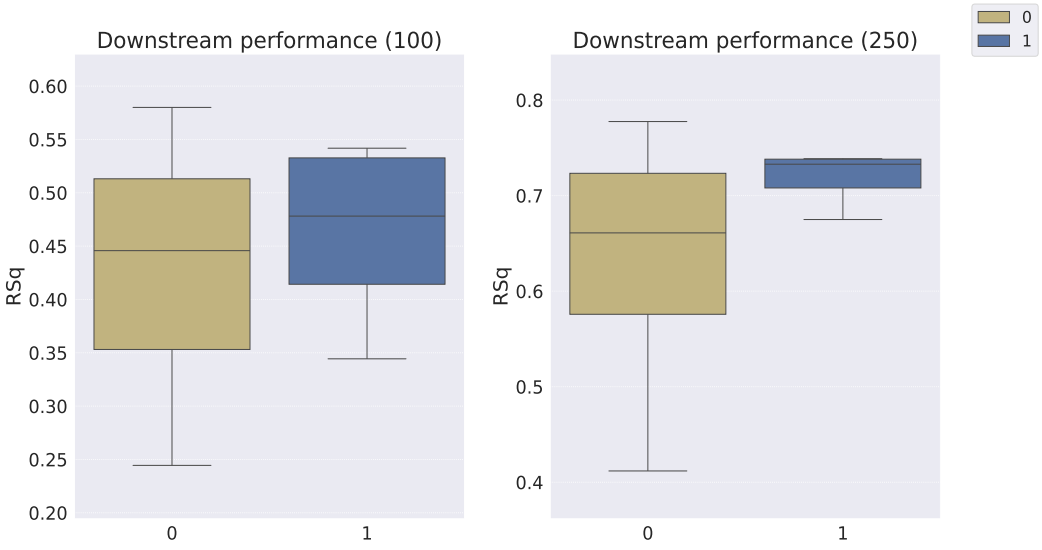

Figure 65: Downstream regression model R^2 scores on the Shapes3D dataset using TPR representations (0) vs Soft TPR representations (1)

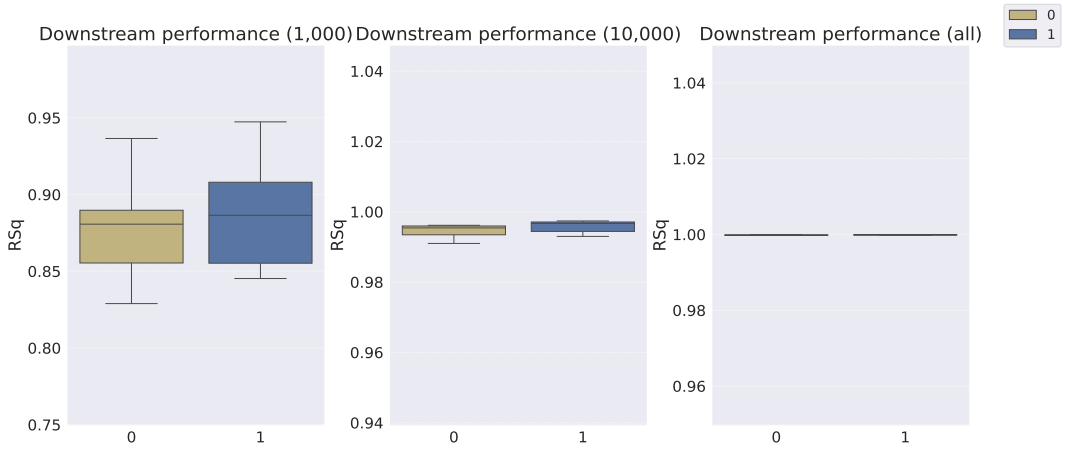

Figure 66: Downstream regression model R^2 scores on the Shapes3D dataset using TPR representations (0) vs Soft TPR representations (1)

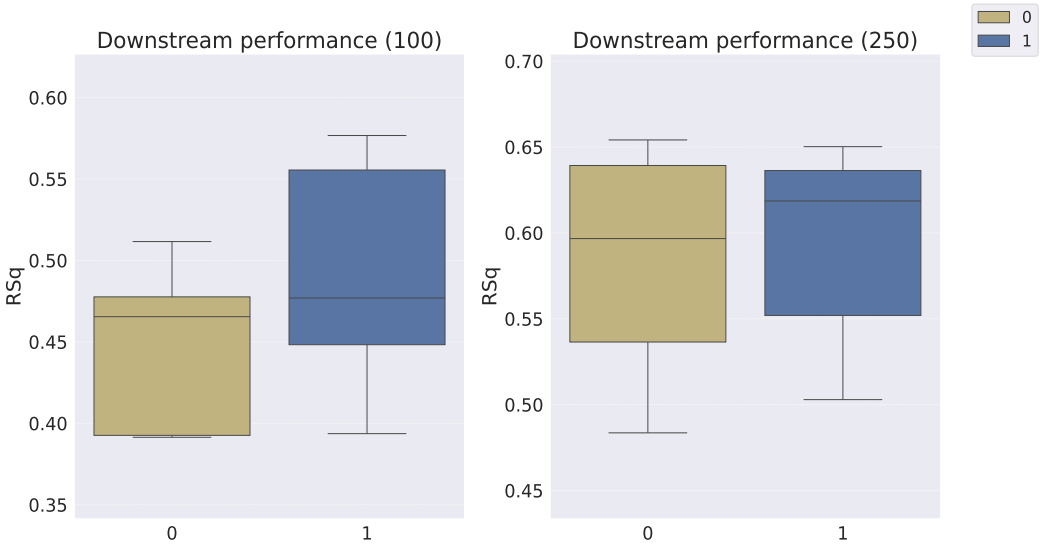

Figure 67: Downstream regression model R^2 scores on the MPI3D dataset using TPR representations (0) vs Soft TPR representations (1)

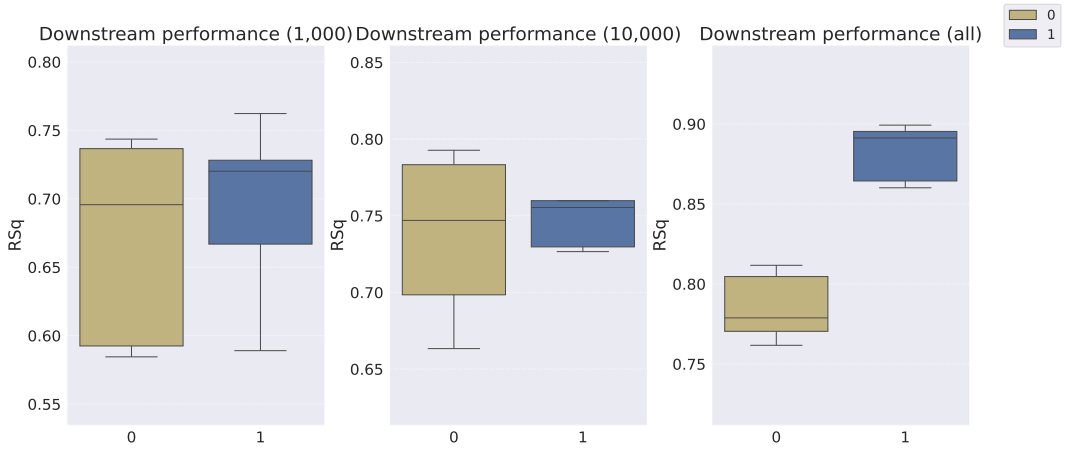

Figure 68: Downstream regression model R^2 scores on the MPI3D dataset using TPR representations (0) vs Soft TPR representations (1)

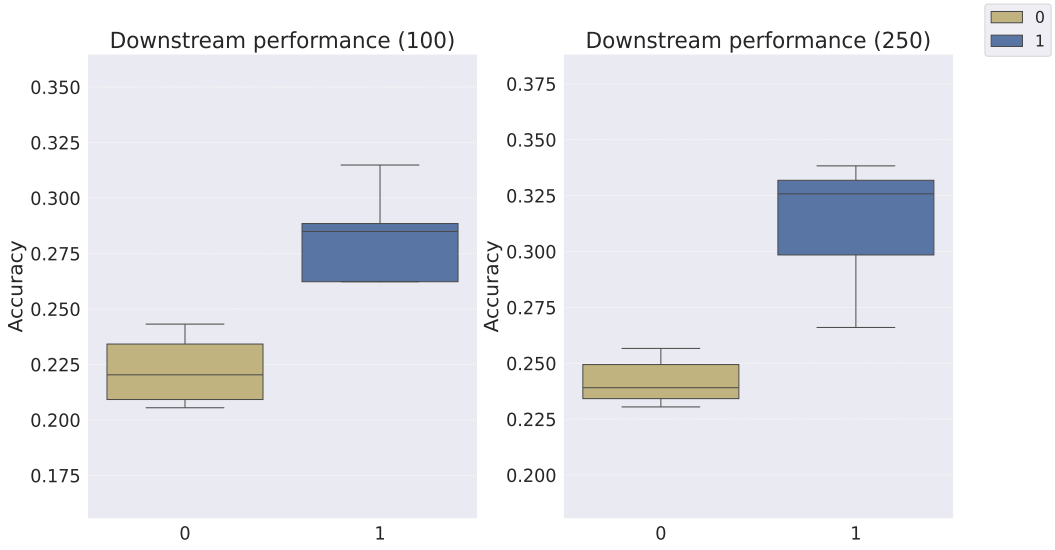

Figure 69: Downstream WReN model classification accuracy on the abstract visual reasoning dataset using TPR representations (0) vs Soft TPR representations (1)

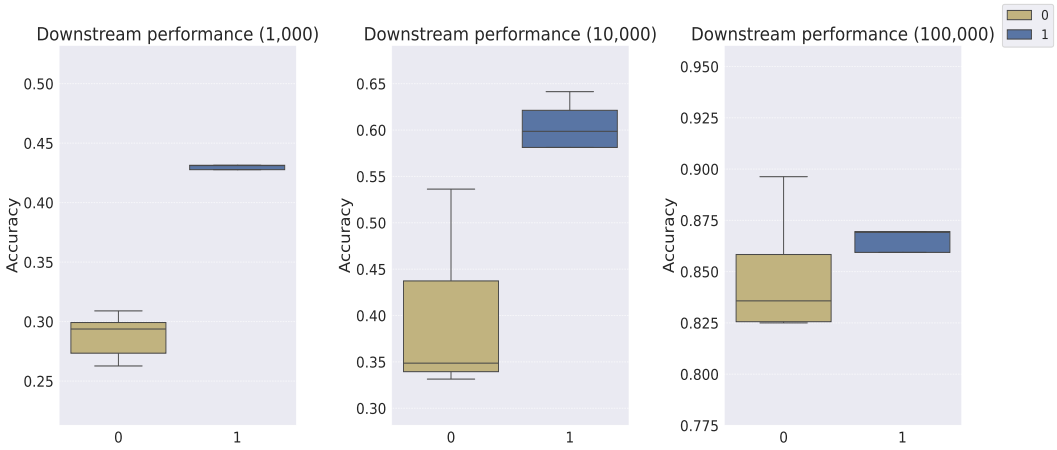

Figure 70: Downstream WReN model classification accuracy on the abstract visual reasoning dataset using TPR representations (0) vs Soft TPR representations (1)

C.6.2 Robustness to Hyperparameter Choices

We perform an additional experiment to empiricially verify that our model is robust to different hyperparameter choices. For the MPI3D dataset, which disentanglement models experience the greatest difficulty in producing disentangled representations for (see Table 1), we randomly pick another set of hyperparameters, shown in Table 41, from the top 5 models based on the MSE loss criterion. For this randomly chosen hyperparameter configuration, we evaluate the disentanglement of the representations on the MPI3D dataset produced by the resulting model.

Table 41: Hyperparameter values of ablation setting

Hyperparameter	MPI3D
	Architectural hyperparameters
D_R	16
N_R *(fixed)*	10
D_F	64
N_F	50
	Loss function hyperparameters
λ_1	0.0000
λ_2	0.25378
β *(fixed)*	0.5

Table 42: Disentanglement metric scores on the MPI3D dataset

Hyperparameter configuration	FactorVAE score	DCI score	BetaVAE score	MIG score
Original	0.949 ± 0.032	0.828 ± 0.015	1.000 ± 0.000	0.620 ± 0.067
Ablation	0.911 ± 0.029	0.798 ± 0.031	1.000 ± 0.000	0.590 ± 0.047

D Limitations and Future Work

D.1 Extension to Linguistic Domains

Applying the Soft TPR to the TPR's typical domain of language is an intriguing future direction, especially as language can deviate from strict algebraic compositionality – for instance, idiomatic expressions such as 'spill the beans' cannot be understood as a function of their constituents alone. Soft TPR's more flexible specification allows it to capture approximate forms of compositionality precluded from the TPR's strict algebraic definition (Eq 1), thereby potentially providing the Soft TPR the ability to better handle the nuance and complexity of language.

To adapt our framework to language, we replace our Conv/Deconv encoder/decoders with simple RNNs, retrain our TPR decoder, and remove the semi-supervised loss, using Eq 6 as the full loss. Preliminary results in Table 43 on the BaBI [7] dataset are compared with TPR baselines from AID [51]. Our Soft TPR Autoencoder does not presently surpass AID, but notable points include:

1. Our use of simpler RRN-based encoders and an MLP-based downstream network, unlike the more sophisticated architectures of [51].
2. Soft TPR retains its performance improvement above the corresponding explicit TPR it can be quantised into.
3. The smaller gap between systematic vs non-systematic dataset splits in our model compared to TPR-RNN (+AID) and FWM.
4. We train our representation learner using self-supervision (reconstruction loss) alone, only employing supervision on the downstream prediction network, while the baselines employ strong supervision and end-to-end training to produce representations.

Table 43: Mean word error rate [%] on the sys-bAbI task [51]

Model	w/o sys diff	w/ sys diff	Gap
TPR-RNN	0.79 ± 0.16	8.74 ± 3.74	7.95
TPR-RNN + AID	*0.69 ± 0.08*	5.61 ± 1.78	4.92
FWM	0.79 ± 0.14	*2.85 ± 1.61*	2.06
FWM + AID	**0.45 ± 0.16**	**1.21 ± 0.66**	**0.76**
Ours (TPR)	4.54 ± 0.61	6.51 ± 1.23	1.97
Ours (Soft TPR)	2.79 ± 0.24	3.64 ± 0.69	*0.85*

D.2 Need for Weak Supervision

To produce a compositional representation, $\psi(x) = C(\psi_1(a_1), \ldots, \psi_n(a_n))$ (as in Section 3.1), each representational constituent, $\psi_i(a_i)$, must map 1-1 to a data constituent, a_i. Without supervision (i.e., access only to observational data, $\{x_i\}$), this is challenging, because the data constituents $\{a_i\}$ underlying each object, x, are unknown and cannot be identified.

We can frame the above intuition in a more mathematically rigorous way, using the generative framework. Essentially, as formally proved in [25], it is impossible to identify the true distribution for the data constituents (generative factors), $p(a)$, using observational data, $\{x_i\}$, alone as there are infinitely many bijective functions $f : \text{supp}(a) \to a$ such that: 1) a and $f(a)$ are completely entangled (i.e. non-diagonal Jacobian) *and* 2) the marginal distributions of a and $f(a)$ are identical (meaning the marginal distributions of the observations are also identical, i.e., $\int p(x|a)p(a)da = \int p(x|f(a))p(f(a))da$). Thus, without inductive biases, it is impossible to infer the data constituents $\{a_i\}$ of any observation, x, from observational data $\{x_i\}$ alone.

To combat this non-identifiability result, in line with [13, 22, 31, 33, 35], we use weak supervision, presenting the model with data pairs (x, x') where x and x' differ in a subset of FoVs, e.g. $\beta(x) = \{\text{shape/cube}, \text{colour/}\textit{purple}, \text{size/large}\}$, $\beta(x') = \{\text{shape/cube}, \text{colour/}\textit{cyan}, \text{size/large}\}$, where the FoV *types* corresponding to the different FoV values are *known* to the model. Note that this weak supervision is minimal, only providing the model to access to the differing FoV *types* in an index set, I, (i.e., $I := \{\text{colour}\}$) and not *any* of the FoV values (i.e., $\{\text{cube}, \text{purple}, \text{cyan}, \text{large}\}$ are all not known by the model).

Some possible future extensions to reduce this level of weak supervision, or alternative forms of weak supervision include:

1. **Embodied Learning**: In the visual domain, some roles, e.g. object position correspond to affordances. Embodied agents may be able to reduce the need for explicit supervision by collecting (x, x') and I through interaction with their environment.
2. **Pretrained Filler Embeddings**: Initialising the filler embedding matrix, M_{ξ_F} with embeddings learnt by a pre-trained vision model could impart knowledge of domain-agnostic fillers (e.g., colours, shapes), reducing the need to explicitly provide (x, x') and I to the model.
3. **Segmentation Masks**: Segmentation masks for each object may potentially reduce the need to explicitly provide (x, x') and I to the model.

D.3 Downstream Utility

Our investigation of downstream utility centers on two selected tasks – FoV regression/classification and abstract visual reasoning, which aligns with the standard framework for assessing the quality and downstream utility of compositional representations [26, 30, 33, 40, 44, 46]. While existing work [30, 33] demonstrates that explicitly compositional representations enhance downstream sample efficiency compared to non-compositional representations, a result we improve upon (C.5.1), the broader utility of compositional representations remains a topic of ongoing exploration [21, 40, 42, 44].

Theoretical perspectives [1, 2] argue that explicitly compositional representations are fundamental in the production of productive, systematic, and inferentially coherent thought – 3 key properties characterising human cognition. Investigating how explicitly compositional representations can yield empirical benefits across these dimensions represents an essential avenue for future research. Although preliminary studies [40, 42, 44] do not find strong evidence that explicitly compositional representations improve compositional generalisation (a key aspect of systematicity), [44] suggests that this finding is because compositional representations are necessary, but not sufficient to induce systematicity; an explicitly compositional processing approach [6] is also required.

Future work could extend our theoretical framework with the hope of producing empirical results consistent with the theoretical arguments of [1, 2]. In particular, as our unbinding module is designed to provably and efficiently recover structured role-filler constituents from the Soft TPR, it may be possible to exploit this module to systematically reconfigure roles and fillers from existing representations to create representations of novel combinations of role-filler bindings (i.e., novel

compositional data). This type of ability could prove extremely beneficial in areas such as concept learning and compositional generalisation.

D.4 Dimensionality

In this subsection, we denote the dimensions of the role and filler embedding spaces as D_F and D_R respectively. We also denote the number of fillers as N_F and the number of roles as N_R.

The Soft TPR belongs to $V_F \otimes V_R$, which is a $D_F \times D_R$ dimensional space, which grows multiplicatively in D_F, D_R. Several factors, however, mitigate scalability concerns in light of this fact:

1. **Independence of Embedding Space Dimensionality**: We note that the dimensionality of the role and filler embedding spaces (D_R, D_F respectively) are properties of the corresponding embedding functions ($\xi_R : R \to V_R$ and $\xi_F : F \to V_F$) and thus, can be fixed independently of the number of roles, N_R, the number of fillers, N_F, or the number of total role-filler bindings (which we denote by n) within a domain. Thus, it is possible to fix the Soft TPR's dimensionality ($D_F \times D_R$) to be smaller than $N_F \times N_R$ (the number of roles/FoV types multipled by the number of fillers/FoV tokens) or n (the number of bindings), which all may be large in complex visual domains. As illustrated in Table 44, the dimensionality of the TPR is smaller than $N_F \times N_R$ in both the Shapes3D and MPI3D domains.
2. **Relaxing Orthogonality**: While D_F can be set a priori with no regard to N_R, N_F or n, we require $D_R \geq N_R$ for semi-orthogonality of the role-embedding matrix M_{ξ_R}, which guarantees faithful (see proof 2 of A.2) and computationally efficient (see 'Unbinding' heading of B.3.1) recoverability of constituents. It is, however, possible to relax this constraint (i.e., to have $D_R < N_R$) to further reduce dimensionality. In this case, semi-orthogonality of M_{ξ_R} is impossible and hence the recoverability of constituents cannot be guaranteed, however, there are some less stringent guarantees on the outcome of unbinding that can still be derived (see p291 of [3] for more details).

We also more explicitly compare the dimensionality of the Soft TPR with baselines in Table 45. Scalar-tokened symbolic representations have a low dimensionality of 10 (N_R) at the expense of representational expressivity (each representational constituent $\psi_i(a_i)$ is scalar-valued). In contrast, Soft TPR has vector-valued representational constituents (i.e. $\approx \xi_F(f_{m(i)}) \otimes \xi_R(r_i)$), similar to VCT and COMET. When compared to these models, the Soft TPR has significantly lower dimensionality compared to VCT and is comparable with COMET.

Table 44: Comparison of multiplicative dimensionality

Parameter	Dataset		
	Cars3D	Shapes3D	MPI3D
D_R	12	16	12
N_R	10	10	10
D_F	128	32	32
N_F	106	57	50
$D_F \cdot D_R$ (*TPR dimension*)	1536	**512**	**384**
$N_F \cdot N_R$	**1060**	570	500

Table 45: Comparison of dimensionality of representations

Model	Cars3D	Shapes3D	MPI3D
	Representational dimension		
Symbolic scalar-tokened compositional representations			
SlowVAE	10	10	10
Ada-GVAE-k	10	10	10
GVAE	10	10	10
ML-VAE	10	10	10
Shu	10	10	10
Symbolic vector-tokened compositional representations			
VCT	5120	5120	5120
COMET	**640**	*640*	*640*
Fully continuous compositional representations			
Ours (Soft TPR)	*1536*	**512**	**384**

D.5 Computational Cost

In the Soft TPR Autoencoder, the expensive tensor product operation is employed to generate ψ^*_{tpr}. Given the computational cost of the tensor product, we more concretely examine the computational

cost of training the Soft TPR Autoencoder by computing the FLOPs for a single forward pass on a batch size of 16 using the open-source implementation of fvcore `https://github.com/facebookresearch/fvcore/tree/main/docs`, visible in Table 46. This data demonstrates that, despite the tensor product's computational cost, the mathematically-informed derivation of our model allows it to obtain compositional representations with vector-valued representational constituents at a significantly lower cost compared to relevant, vector-tokened baselines (2 orders of magnitude less than VCT, and 4 orders of magnitude less than COMET).

Future work could explore the use of tensor contraction techniques to reduce computational expense. For instance [29] uses a Hadamard product based tensor product compression technique. This reduces computational cost from n^2 (tensor product of 2 vectors) to n (Hadamard product), but comprises the theoretical guarantees on constituent recoverability. We believe developing tensor contraction techniques within the TPR framework is an important direction for future research, to ensure efficient TPR-based representations with provable recoverability of constituents.

Table 46: Comparison of FLOPs required for a forward pass of batch size 16

Model	Cars3D	Shapes3D	MPI3D
	Representational dimension		
Symbolic scalar-tokened compositional representations			
SlowVAE	1.47×10^8	1.47×10^8	1.47×10^8
Ada-GVAE-k	1.47×10^8	1.47×10^8	1.47×10^8
GVAE	1.47×10^8	1.47×10^8	1.47×10^8
ML-VAE	1.47×10^8	1.47×10^8	1.47×10^8
Shu	1.45×10^8	1.45×10^8	1.45×10^8
Symbolic vector-tokened compositional representations			
VCT	2.53×10^{11}	2.53×10^{11}	2.53×10^{11}
COMET	5.12×10^{13}	5.12×10^{13}	5.12×10^{13}
Fully continuous compositional representations			
Ours (Soft TPR)	3.21×10^9	2.93×10^9	2.89×10^9

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: All claims accurately reflect the paper's contributions and scope. We provide theoretical proofs in the Appendix, as well as our complete suite of experimental results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We explicitly state in the last paragraph of the paper that future work is to be done in developing hierarchical Soft TRP representations.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [Yes]

Justification: We include complete proofs for all theoretical results in the Appendix.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide all information (specification of model architecture, computing resources, hyperparameters) to replicate experimental results in the Appendix.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: All datasets are open-access. We provide sufficient instructions in our Appendix to reproduce experimental results. Furthermore, we provide the main code for our Soft TPR model architecture with this submission. We plan to release a streamlined version of our entire code base used to conduct all experiments if the paper is accepted.

Guidelines:

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

6. **Experimental Setting/Details**

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

Answer: [Yes]

Justification: We provide these details in the Appendix, and additionally, in the code.

Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [Yes]

Justification: We present results with standard deviations, as well as the IQR and whiskers extending to 1.5 times the IQR for all box plots.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer 'Yes' if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We specify the specific GPUs we use for all experiments in the Appendix, as well as the average amount of time it takes to train the Soft TPR Autoencoder for each dataset.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: Our research conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: As the central aim of our work relates to the representational form of compositional representations, our research is theoretical in nature and has no immediate societal impact.

Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our work poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: In the main paper, as well as the Appendix, we cite the papers associated with each dataset we use. We additionally make explicit references in the Appendix to all externally created code that we use in our experiments. Furthermore, if the paper is to be accepted, we will clearly provide licenses and attribution to the original authors where applicable in the code files.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New Assets**

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

    Answer: [Yes]

    Justification: The code we submit in our submission corresponds with our Soft TPR Autoencoder architecture, and is thus reasonably straightforward to understand, especially when supplemented by this paper. If this paper is accepted, however, we will include more extensive written documentation for the official, more extensive codebase we release.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: Our research does not involve crowdsourcing nor human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

Justification: Not applicable.

Guidelines:

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