Interpretable Representations in Artificial Neural Networks

A DOCTORAL THESIS

Alexander F. Spies

IMPERIAL COLLEGE LONDON · MMXXVI

This thesis was submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College London and the Diploma of Imperial College London.

Department of Computing
Imperial College London
June 2026

This private keepsake edition was prepared by the author and printed in an edition of twenty copies. It is not for sale.

Typeset by the author in Charter and TEX Gyre Pagella using LATEX. Cover design by the author; cover and division artwork generated by the author using AI image tools.

Printed and bound in the United Kingdom by Bookvault.

For Rachel St. John-Crees

You opened my eyes to the wonders of the universe

ABSTRACT

This thesis investigates interpretable representations in Artificial Neural Networks, examining both engineered approaches and structure discovered within existing architectures. Whether imposing structure on networks yields benefits over simply scaling model size and data remains debated; our work contributes to understanding this tension between structure and flexibility.

We first engineer interpretable representations through modular architectures. Extending Slot-Attention and combining it with spatial transformers, we build networks that segment images into objects with disentangled representations of pose, shape, and appearance, and show these are easier to interpret than less structured counterparts.

To probe the link between interpretability and generalisation, we pair these object-centric representations with specialised relational reasoning modules. Although this yields sparser reasoning, combining structured approaches often underperforms both the individual methods and standard monolithic networks. We trace this to compounding rigidity: as multiple structural biases interact, they create information bottlenecks between network stages.

Given these difficulties, we turn to emergent structure in state-of-the-art architectures, which perform complex reasoning without strong inductive biases. Focusing on planning as a form of reasoning, we show that foundation models solving maze navigation develop internal world models reflecting spatial structure. These representations can be causally perturbed to alter the model's behaviour, revealing how the networks internally represent and reason about spatial relationships.

Together, these complementary approaches — engineering interpretable architectures and analysing emergent representations — advance our understanding of structure in neural networks. While imposed structure can enhance interpretability, its relationship with performance and generalisation remains complex. Our findings suggest a promising direction: rather than imposing rigid constraints, future architectures may achieve interpretability by leveraging and amplifying the natural emergence of structured representations in flexible, scaled systems.

ACKNOWLEDGEMENTS

For all that was required to get this far, I thank my parents for making me who I am — for sacrificing so much so that their son could wander from one ivory tower to another, and for believing in him even when he did not.

For the existence of this work, I thank my supervisors, Alessandra and Murray, for seeing potential in the naïve young man who sought them out years ago — for endowing me with some of their wisdom and taste, and for keeping me on course in spite of myself.

For broadening my horizons, I thank my collaborators, especially those in SPIKE and the 井上 Lab, for shaping the way I think — for meeting my ideas with their own, and for venturing side-by-side into the unknown.

For the will to persevere, I thank Emmi and my friends for carrying me through when the work was hardest — for shaping me in ways innumerable, and for keeping me afloat come hell or high water.

Thank you, all of you. You turned my existence into a life.

Contents

Preliminaries

Chapter 1

Introduction

Behind the capabilities of modern AI systems lies a question as old as the field itself: should intelligent systems be carefully designed, with explicit representations of knowledge and mechanisms for reasoning, or through the scaling of generic learning architectures? And, regardless of how they are built, can we peer inside to understand what they have learned? These questions, and the extent to which they are intertwined, are central to the work presented in this thesis.

When the work presented in this thesis was started, it was commonplace for academics, the author included, to discuss the lofty goals of Artificial Intelligence, and to bet on how many more decades of intellectual toil would be required to achieve them. In the half decade since, the field has seen truly rapid progress — foundation models now demonstrate capabilities in visual reasoning, language understanding, and planning that once seemed decades away (Brown et al. 2020; Kocijan et al. 2022; Reed et al. 2022; Saharia et al. 2022). Whilst sceptics and naysayers still point out the shortcomings of what has been wrought (Chomsky et al. 2023), most have been shocked by the speed of progress (Bengio, Geoffrey Hinton, et al. 2024). The new generation of PhD students no longer debate the possibility of general intelligence in our lifetime, but instead carry out their work and discussions with AI systems whose capabilities draw close to those we once dreamt of. We now find ourselves thrust into a world of dizzying progress, where hundreds of millions of people use such systems to write for them, think for them, paint for them and even to make them feel loved (Ebner et al. 2025; Heerden 2025; Kosmyna et al. 2025; Liang et al. 2025).

1.1 Structure vs. Scale

This tension between designing interpretable systems or relying on the scaling of generic ones is not new (Cingillioglu 2022). Similar choices were deliberated upon even in the field's founding at the 1956 Dartmouth Summer Research Project (McCarthy et al. 1955). These have manifested across decades of research in different forms but with consistent trade-offs. The early divide between symbolic AI, championed by pioneers like Newell and Simon (Newell et al. 1976), and the connectionist revival of the 1980s (Rumelhart, McClelland, et al. 1986) prefigured modern debates around structure vs. scale. The trade-offs between these two approaches can be viewed in many ways, but we choose to elucidate them along three axes, as shown in Figure 1.1:

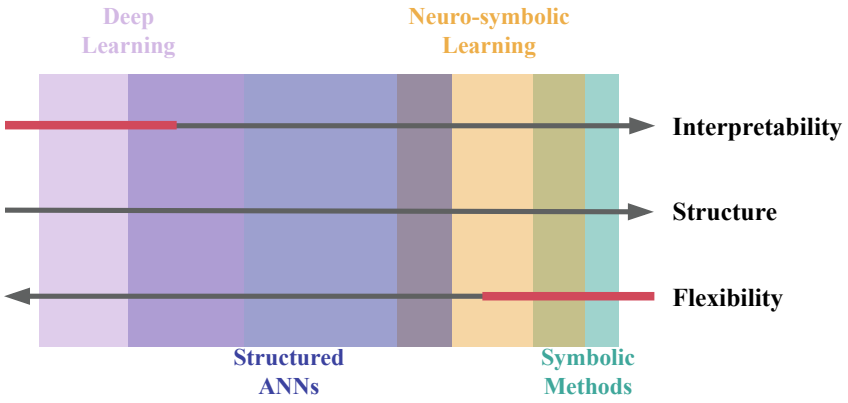


Figure 1.1: Interpretability Flexibility Trade-off. Existing methods mostly suffer from shortcomings at either end of the spectrum shown, sacrificing flexibility or interpretability. In Part I we focus on structured Artificial Neural Networks (ANNs), attempting to find a middle-ground between interpretability and flexibility (the middle region). Another tack is to attempt to find ways of interpreting flexible systems (top left shortcoming). We take this approach in Part II for scale-driven approaches, by finding ways to reverse-engineer the representations that emerge naturally. Though not crisply delineated from structured ANNs, Neuro-symbolic approaches attempt to make interpretable symbolic systems scalable, by combining them with neural networks; whilst promising we do not explore such approaches in this thesis for reasons discussed in the text.

- **Interpretability:** How easy is it to understand the systems behaviour? Either from thousands of hand-written logical rules, or billions of floating point numbers.
- **Structure:** How much structure is imposed on the system by the designers, at the level of input/output formats or through architectural constraints.
- **Flexibility:** (closely related to **Scale**) How easily can the system be applied to new domains, learn from more data or leverage more compute at inference time?

Logic-based Systems During the 1970s-1990s, a significant portion of the AI research community pursued the construction of comprehensive

knowledge bases intended to encode common-sense and practical knowledge about the world (Davis 2016).

Proponents argued that such systems could capture human-like high-level reasoning and provide formally verifiable behaviour across multiple domains. While these approaches have demonstrated success in specialised domains where formal verification is critical (Alrajeh et al. 2018; Čyras et al. 2021; Law et al. 2020), they face well-documented challenges in flexibility and scalability — as modifying or extending, by hand or through learning, previous rules and facts becomes increasingly difficult as such systems grow (Brooks 1991; Dreyfus 1971; McDermott 1987). A key advantage of logic-based systems is their interpretability: decisions can be traced through explicit inference chains (Rudin 2019). However, as rule bases grow to thousands of rules, the comprehensibility of such traces is not self-evident, though recent work has made progress on generating human-readable explanations from complex rule systems (Alviano et al. 2023; Fandinno et al. 2018).

Neuro-symbolic Systems The limitations of purely logic-based approaches have motivated extensive research into hybrid neuro-symbolic architectures, as comprehensively surveyed by Garcez et al. 2020, and discussed in section A.1. Broadly speaking, such approaches attempt to combine the flexibility of ANNs with the strong priors and interpretability provided by logic-based methods. This diverse field spans a spectrum of integration strategies. At one end, recent work employs large language models to automatically generate logical rules (Albinhassan et al. 2025; Wong et al. 2023), or leverages LLMs to interface with existing logic-based reasoning systems (Sadowski et al. 2025).

At the other end, researchers have developed methods for differentiably learning symbolic programs directly from data (Evans et al. 2018; Manhaeve et al. 2018; Minervini et al. 2020). A particularly promising direction combines these symbolic reasoning capabilities with neural networks for perception tasks (Aspis et al. 2024; Cingillioglu and Alessandra Russo 2022; Cunnington et al. 2023), exploiting the complementary strengths of pattern recognition and structured reasoning.

However, these approaches face significant technical challenges: learning the highly sparse representations required for symbolic reasoning remains difficult in gradient-based frameworks, where credit assignment becomes problematic when only a small subset of rules or symbols are relevant to any given input (Evans et al. 2018). Furthermore, scaling differ-

entiable logic-based methods beyond toy domains has proven challenging, with computational costs growing prohibitively as the space of possible rules or predicates expands (H. Dong et al. 2019; Zimmer et al. 2023). These hybrid approaches nonetheless aim to preserve the interpretability benefits of logic-based systems while addressing their shortcomings through neural components.

Structured Neural Networks All neural networks possess inherent structure through their input-output specifications and architectural design. Beyond this baseline, researchers have developed approaches that impose explicit structural priors, including object-centric learning (Greff, van Steenkiste, et al. 2020), graph neural networks (P. W. Battaglia et al. 2018), capsule networks (Sabour et al. 2017), to list but a few. These approaches are sometimes characterised as "differentiable programming" (Blondel et al. 2025), where domain-specific computational structures are embedded within gradient-based optimisation frameworks. As illustrated in Figure 1.1, the boundaries between neuro-symbolic systems, structured neural networks, and conventional neural architectures are not formally defined and exhibit significant overlap. Despite this taxonomic ambiguity, these research programs are distinguished by their initial design priorities along our three axes, with structured ANNs placing more weight on scalability and flexibility over interpretability compared to neuro-symbolic approaches.

In summary , these varied approaches to combining structure with learning highlight a fundamental challenge: how to balance the interpretability of explicit symbolic reasoning with the flexibility and scalability of neural computation. Rather than viewing this as a forced choice between extremes, we can seek approaches that preserve interpretability while remaining grounded in the practical realities of modern deep learning — though doing so present daunting scientific challenges that may take many years to resolve.

1.2 The Urgency of Interpretability

Whilst progress continues at breakneck speed, with no signs of relenting, we must face realities that a short while ago were deemed fiction. How can we be sure that these systems are reliable and safe, as they are handed increasing autonomy and replace ever greater amounts of human labour? More

fundamentally, how can we understand what these systems have learned, and ensure their internal representations align with human concepts? The challenge of interpretability, understanding the internal workings of neural networks, has become paramount (Amodei 2025). Had we twenty years, we could strive to understand these systems methodically — making them safer, and increasing our understanding of intelligence in the process. But increasingly fewer researchers now believe that we have that long to solve the technical challenges that building thoroughly reliable and trustworthy AI systems will likely require (see Figure 1.2).

The asymmetry between our ability to build capable systems and our ability to understand them continues to widen. Capability research scales readily with compute and data; interpretability research, still in its infancy, as of yet does not — and models reach hundreds of millions of users before we thoroughly understand them. We can probe, visualise, and sometimes steer model behaviour, but we cannot yet guarantee that our tools capture all safety-relevant phenomena, nor that our interpretations are faithful rather than convenient.

Beneath this practical urgency lies an even deeper problem: we lack a true theory of interpretability itself. What does it mean to "understand" a neural network? When is an explanation sufficient? How do we verify that our interpretations aren't merely compelling confabulations? These unanswered questions suggest that the challenge is not merely technical but also epistemological—and that no single paradigm is likely to resolve it alone.

In light of these challenges, rather than betting on a single paradigm, this thesis takes a dual approach to understanding representations in artificial neural networks. First, we investigate how engineering structured architectures — specifically object-centric representations (Locatello, Weissenborn, et al. 2020) — can promote interpretable and compositional reasoning (Part I).

We demonstrate that models equipped with explicitly structured decoders learn more disentangled representations, and show how explicit relational reasoning architectures can leverage these structured representations for systematic generalisation.

Second, we analyse how structure emerges naturally in existing models, revealing that even simple transformers (Vaswani et al. 2017) trained on maze-solving tasks develop linear representations of spatial structure without explicit architectural biases (Part II).

Through mechanistic interpretability techniques (Bereska et al. 2024), we identify specific attention heads that respect maze topology and find

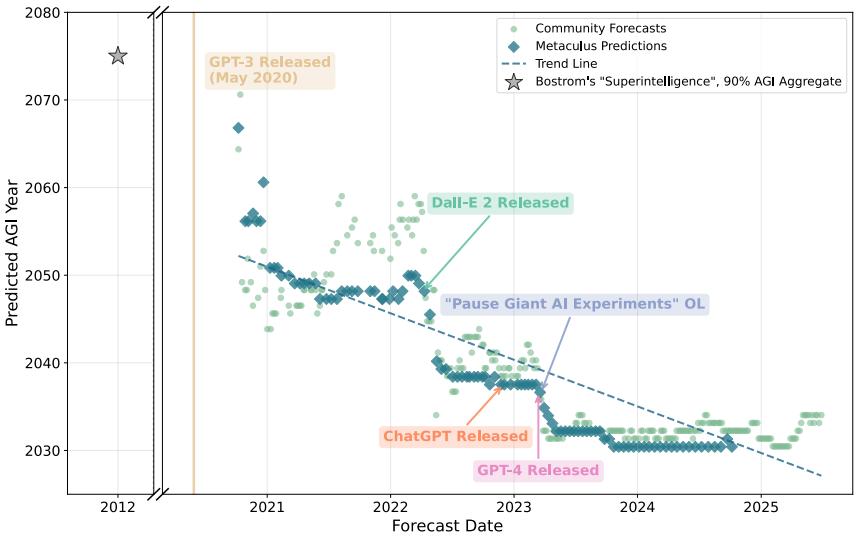


Figure 1.2: Expert and Community Predictions for creation of the first Artificial General Intelligence (AGI) Green circles plot the community median for the Metaculus question "Date of Artificial General Intelligence" (Metaculus 2025), while blue diamonds plot Metaculus's algorithmic aggregate prediction on the same day (data collected daily, 23 Aug 2020 — 6 Jun 2025). A downward-sloping ordinary least-squares trend (dashed) shows that expected AGI arrival has been moving forward by $\approx 5\frac{1}{2}$ years for every year of real time. Vertical markers highlight capability and policy inflection points — GPT-3, DALL-E 2, ChatGPT, GPT-4, and the March 2023 "Pause Giant AI Experiments" open letter — each followed by an abrupt left-shift in the crowd timelines. The silver star at forecast-year 2012 indicates the 90% upper-quantile date (2075) from the expert-survey aggregate reported by Bostrom 2014; this pools four expert surveys conducted between 2011 and 2013, from which we take the simple unweighted mean of the 90th-percentile ($p = 0.9$) dates (≈ 2075) and plot it at 2012 (the approximate midpoint of the studies) to give a single aggregate reference point.

that causal "world models" form within such models and underlie their behaviour.

These approaches reflect a deep tension in modern AI research. The structured architectures explored in Part I represent the classical vision of building interpretability into systems by design — ensuring that internal representations decompose scenes into objects and their relationships, as advocated by the object-centric learning community (Greff, van Steenkiste, et al. 2020; Locatello, Weissenborn, et al. 2020). The mechanistic analyses

in Part II acknowledge the reality that the most capable systems today leverage scale rather than structure (Amodei et al. 2018; Sutton 2019), yet demonstrate that interpretable representations can arise naturally from gradient descent on well-designed tasks.

It remains unclear which paradigm will ultimately prevail — whether we will achieve safe and interpretable AI through careful architectural design or through post-hoc analysis of emergent behaviours — and whether either approach will mature quickly enough to address the pressing challenges posed by increasingly powerful systems. However, the greatest insights likely lie at the intersection of these approaches: understanding how to build structure that promotes interpretability while also developing tools to analyse the representations that emerge naturally. This thesis aims to serve as a bridge between these research paradigms, demonstrating that they are not competing philosophies but complementary tools in our urgent quest to build AI systems we can understand and trust — whether by constructing them to be interpretable from the ground up, or by developing techniques to peer inside the black boxes we have already built.

1.3 Thesis Outline

The thesis begins with shared background material (chapter 2), introducing foundational concepts in deep learning—including neural network architectures, attention mechanisms, and representation learning—that underpin both parts of the thesis. Part I then turns to the question of whether architectural constraints can promote interpretable, compositional representations. After reviewing the foundations of object-centric learning in chapter 3, chapter 4 examines how factorising object pose and appearance through spatial transformer constraints yields more disentangled latent representations.

Chapter 5 then takes a step back and investigates whether object-centric representations improve relational reasoning, finding that whilst sometimes detrimental, sparsity constraints on relational modules can lead to simpler decision boundaries.

Having established both the value and the limitations of engineered structure, we take a step back in the Interlude (chapter 6) to ask whether structure is needed at all, systematically evaluating multimodal foundation models on visual relational reasoning tasks. While scale alone yields impressive capabilities, a notable gap remains between language-mediated and visually grounded reasoning. This finding motivates the shift in per-

spective that defines the second half of the thesis: rather than building structure in, we turn to reverse-engineering the structure that emerges on its own.

Part II begins by introducing the methodological framework for mechanistic interpretability in chapter 7, including probing techniques, sparse autoencoders, and the maze-solving domain that serves as a testbed throughout. Chapter 8 then shows that small transformers trained on maze navigation develop linearly decodable spatial representations whose emergence coincides with a generalisation phase transition. Building on this, chapter 9 reveals that these models construct causal world models: sparse autoencoder features encoding maze connectivity are causally implicated in path-finding behaviour and can be manipulated through targeted interventions. Finally, chapter 10 reflects on the complementary nature of these two approaches—building structure in and reverse-engineering structure out—and outlines future directions for interpretable AI systems.

1.4 Publications

The following articles and publications form the basis of the work presented in this thesis. Where contributions are not entirely the author's own, acknowledgements are given at the beginning of the relevant chapters, as well as for any relevant results or figures. Chapters 4 and 6 contain work not previously published at the time of writing.

Sparse Relational Reasoning with Object-Centric Representations

Spies, A.F., Russo, A., Shanahan, M.

ICML 2022 DyNN Workshop (spotlight)

The focus of chapter 5. We investigate whether object-centric representations improve compositional relational reasoning and what role sparsity plays, finding that whilst object-centric representations alone do not guarantee improvement, L1 sparsity constraints on relational modules yield simpler, more interpretable decision boundaries without sacrificing task accuracy. The techniques used to analyse the disentanglement of the representations are introduced in chapter 4, which itself contains mostly unpublished work.

maze-dataset: Maze Generation with Algorithmic Variety and Representational Flexibility

Ivanitskiy, M.I., Sandoval, A., Spies, A.F., Räuker, T., Knutson, B., Diniz Behn, C., Wu Fung, S.

Journal of Open Source Software, 10(114), 8633 (2025)

Supports chapter 8 and chapter 9. This open-source library provides configurable maze generation for mechanistic interpretability research, underpinning the experimental infrastructure throughout Part II. The author contributed to library design, documentation, and integration of the maze generation algorithms used in Chapters 8 and 9.

Linearly Structured World Representations in Maze-Solving Transformers

Ivanitskiy, M.I., Spies, A.F., Räuker, T., Corlouer, G., Mathwin, C., Quirke, L., Rager, C., Shah, R., Valentine, D., Behn, C.D., Inoue, K., Fung, S.W.

NeurIPS 2023 UniReps Workshop (in proceedings)

The focus of chapter 8. We investigate whether transformers trained on maze navigation develop interpretable spatial representations without explicit architectural biases, finding that linearly decodable spatial structure emerges during training in concert with a generalisation phase transition. The author led the probing analyses, residual-stream geometry experiments, and training-dynamics study.

Transformers Use Causal World Models in Maze-Solving Tasks

Spies, A.F., Edwards, W., Ivanitskiy, M.I., Skapars, A., Räuker, T., Inoue, K., Russo, A., Shanahan, M.

ICLR 2025 Workshop on World Models

The focus of chapter 9. Building on the correlational evidence from Ivanitskiy, Spies, et al. 2024, we ask whether emergent internal representations are causally implicated in model behaviour. Using sparse autoencoders we identify features encoding maze-wall connectivity and show, through targeted activation patching, that these features causally drive path-finding decisions—providing the first evidence of manipulatable causal world models in maze-solving transformers leveraging a multi-pronged analysis. The author led this work and carried out the SAE training and experiments.

Chapter 2

Shared Background

In this chapter we outline the key concepts central to developing our study of representations in Artificial Neural Networks (ANNs). The two central parts of our thesis are preceded by their own, more detailed, background sections and readers familiar with the fundamental concepts in modern "Deep Learning" may skip ahead to those.

2.1 Mathematical Notation

Symbol	Category	Definition
General Notation		
t	Time	Time step
T	Time	Total number of time steps
$\vec{v}$	Vector	Typically used for inputs or internal (latent/hidden) representations
$\mathbf{A}$	Matrix/Tensor	Typically used for images, sequences, or other high-dimensional data
$\mathbf{A}^\top$	Operation	The transpose of a Matrix/Tensor (note $\top$ differs from T)
Neural Networks		
$\vec{z}$	Vector	Latent representation
$\mathbf{W}$	Matrix	Weight matrix
$\vec{b}$	Vector	Bias vector
σ	Function	Activation function
$\mathcal{L}$	Loss Function	Loss function used for training

Table 2.1: General mathematical notation used throughout this thesis. **See also** Table 10.3 for a glossary of terms.

2.2 Deep Learning

Throughout this thesis we will be working with Artificial Neural Networks (ANNs) consisting of multiple “layers", in which information flows from the input layer to the output layer, along a forward pass. When there are

more than a few layers, such a neural network is said to be “deep", and thus the term “deep learning" is used to refer to the study of such networks and their learning. As we shall discuss, the stacking of ever more layers is what enables relatively simple neural networks to learn increasingly complex and hierarchical representations.

2.2.1 Neural Networks

Many excellent treatments of ANNs and artificial intelligence more broadly exist (Bishop 2007; Goodfellow et al. 2016; Russell et al. 2020), and we will not attempt to provide a comprehensive overview here. Instead, we will focus on the key concepts and nomenclature that will be built upon in later chapters.

Artificial neurons, conceived of as a simplified model of their biological equivalents, were first introduced by McCulloch et al. 1943. In their simplest form, a single neuronal unit computes a weighted sum of its inputs followed by a nonlinear activation: $y = \sigma(\vec{w}^T \vec{x} + b)$ where $\vec{w}$ is a weight vector, $\vec{x}$ is an input vector, b is a scalar bias, and σ is an activation function. When multiple such neurons are arranged in parallel, they form a layer, whose computation can be expressed as $\vec{y} = \sigma(\mathbf{W}\vec{x} + \vec{b})$ where $\mathbf{W}$ is now a weight matrix (each row containing the weights for one neuron), $\vec{b}$ is a bias vector, and the activation function σ may be applied element-wise or operate on the entire vector. This model, known as the perceptron (or a Linear Layer), can be extended to a Multi-Layer Perceptron (MLP) by stacking multiple layers, with the output of one layer serving as the input to the next.

Some parameters in a network are learnable, and others are non-learnable. Learnable parameters are typically called “weights", and are learned from the data directly via gradient descent. Non-learnable parameters are typically called “hyperparameters", and define the architecture of the network, such as the number of layers, the number of units in each layer, and the activation functions used; these influence the performance of the network, but are not learned directly via gradient descent[1].

Whilst such a simple construction is not conceptually far from the state of the art, early progress in the field was hampered by discussions around the limitations of such simple neural networks to model certain classes

[1] Meta-learning and neural architecture search are two approaches to learning the architecture of a network, but are beyond the scope of this thesis and in these cases one may simply consider that the set of “hyperparameters" has changed.

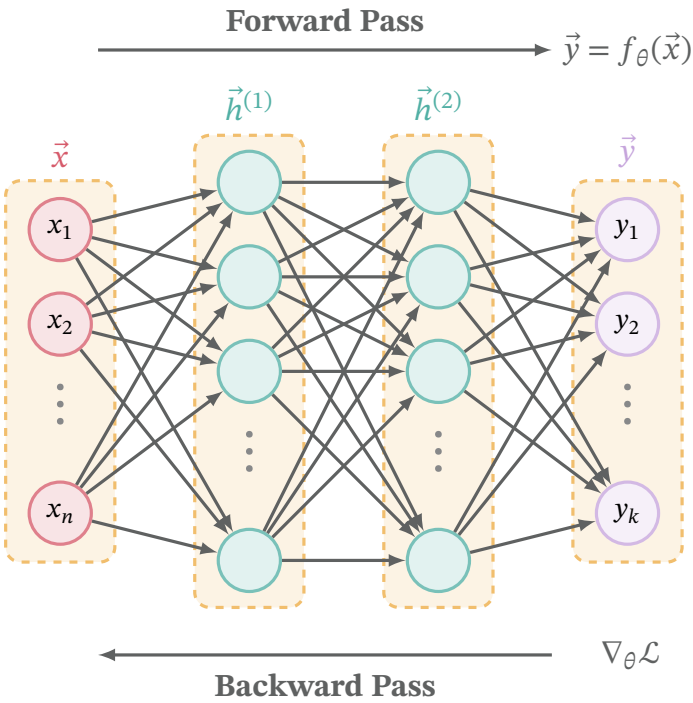


Figure 2.1: A simple feed-forward neural network, also referred to as a Multi-Layer Perceptron, with two hidden layers, $h^{(1)}$ and $h^{(2)}$. During the forward pass a set of inputs are passed through the network and outputs are computed, during the backward pass the gradient of the loss function with respect to the network parameters is computed via backpropagation. The network is parameterised by the weights θ, which are learned from the data via gradient descent. The loss function $\mathcal{L}$ is used to compute the prediction error. Adapted from Love 2024.

of functions (Minsky et al. 1969), but as we shall explore, modern neural network “architectures" consist (at least conceptually) of variations of this simple idea (Goodfellow et al. 2016).

In particular, early issues around the stability of learning have been substantially improved through the use of regularisation techniques such as weight decay (Krogh et al. 1991) and dropout (Srivastava et al. 2014), improved optimisers such as Adam (Diederik P Kingma et al. 2015), techniques such as batch normalisation (Ioffe et al. 2015), and better activation functions such as ReLU (Nair et al. 2010).

Forward Pass

We typically refer the computation of a network's outputs as the "forward pass", and the computation of the gradient of the loss function with respect to its parameters by the term "backpropagation". During the forward pass, information flows from the input layer to the output layer through successive transformations. The forward pass is given by $\vec{y} = f_\theta(\vec{x})$ where $\vec{y}$ is the output of the network, $\vec{x}$ is the input to the network, and f_θ is the network function, parameterised by its set of weights θ. In a simple feed-forward network, this f_θ may be decomposed into a purely sequential composition of transformations. Each such "hidden layer" is a function of the previous layer's output: $\vec{h}^{(l+1)} = f^{(l)}_{\theta^l}(\vec{h}^{(l)})$, where $\vec{h}^{(l)}$ is the output of layer l and $f^{(l)}_\theta$ is the function of layer l.

The compositional nature of such networks enables complex function approximation (Hornik et al. 1989).

Residual connections (also called "skip" connections) are an architectural trick of note, which allow information to flow directly from the input to the output of a layer, bypassing the intermediate layers. This is particularly useful for training deep networks, as it allows the gradient to flow through the network without "vanishing" or "exploding" (see Goodfellow et al. 2016, §8.2.5) (Balduzzi et al. 2017; Hochreiter 1991; Zaeemzadeh et al. 2020), and also making the loss landscape more amenable to stochastic optimisation (Kawaguchi et al. 2019; H. Li et al. 2018). In such architectures, the layer-wise forward pass takes the form $\vec{h}^{(l+1)} = f^{(l)}_\theta(\vec{h}^{(l)}) + \vec{h}^{(l)}$. As we shall see in Part II, the presence of residual connections also strongly influences the ways in which networks form hierarchical representations.

Learning in Neural Networks

The problem of "credit assignment" — choosing which parts of a network should be updated, and by how much, for a given provided output — has an elegant solution in ANNs, provided by the backpropagation algorithm (Rumelhart, G. E. Hinton, et al. 1986). This largely amounts to the automated and repeated application of the chain rule to networks, carried out in combination with gradient descent during the **backward pass**.

The "prediction error" is computed by comparing the output of the network to the desired output, and is encapsulated by a loss function, $\mathcal{L}$. For example, in supervised learning, one might use the sum of squared

differences between the outputs and a *known* "target". Notably, any differentiable term which depends on the parameters of a network may be added to provide additional regularisation, leading to significant differences in the learned representations, and potentially generalisation (ideas we explore in chapter 5) (Burgess, Higgins, et al. 2018). For example, a multilayer network may map an input $\mathbf{x}$ to a lower dimensional, intermediate representation (a so-called "latent representation"), $\mathbf{z}$; a neural network designer may then choose to add a regularisation term, such as the L1 norm of the latent representation, $\mathcal{L}(\mathbf{z}) = \lambda|\mathbf{z}|_1$, where λ trades-off between the sparsity of the latent representation and the reconstruction error.

Whilst backpropagation provides a simple method for credit assignment, early ANNs often struggled with pathological gradients which either (i) "vanished" (where the repeated application of the chain rule resulted in gradients close to zero, leading to no learning) or, (ii) "exploded" (where the repeated application of the chain rule resulted in very large, or overflowing gradients, leading to unstable learning). Another common critique of backpropagation is that whilst efficient in ANNs, it does not appear biologically plausible, as it relies on global information propagation from the output layer to the input layer (T. P. Lillicrap et al. 2020). There has however been much work exploring possible analogues for credit-assignment in biological brains (Whittington et al. 2019).

2.2.2 Neural Network Optimisation

The choice of optimisation algorithm strongly influences training dynamics and the regions of the "loss landscape" that are explored. In a convex loss landscape, suitably tuned gradient descent will converge to the global optimum, but in a non-convex loss landscape, this is not guaranteed (Boyd et al. 2004). Modern optimisers, such as Adam (Diederik P Kingma et al. 2015), employ a combination of gradient descent and momentum to navigate the loss landscape. More specifically, Adam maintains exponentially decaying estimates of both the first and second moments of the stochastic gradients to form element-wise adaptive step sizes.

The momentum term helps to compensate for the "noise" in the gradient, which is the variance between the optimal step and the actual step taken that arises due to the estimate of the gradient from a batch (a subset of the full dataset). As the name suggests, the momentum term is a weighted average of the gradient at the current step and the gradient at the previous step, smoothing out noisy gradient steps and improving learning in the majority of cases (Mandt et al. 2017; Polyak 1964).

In practice, the loss landscape of complex neural networks is highly non-convex but exhibits several remarkable properties:

- Most minima are nearly global and reside in relatively flat basins—this flatness has frequently been linked to improved generalisation (Hochreiter and Schmidhuber 1997a; Izmailov et al. 2018; Keskar et al. 2017; Nguyen et al. 2017)

- The extremely high-dimensional loss landscape (due to overparameterisation) allows networks to explore many local minima, which can be thought of as connected by "paths" of low-loss regions. This phenomenon, known as "mode connectivity" suggests that seemingly distinct minima often lie within larger, gently sloping basins (Garipov et al. 2018; Gotmare et al. 2018)

- Large networks frequently contain sparse "winning ticket" subnetworks that can achieve comparable accuracy to the fully-trained network (Frankle et al. 2019), while the relationship between model capacity and performance can follow a "double descent" curve where increasing the capacity of the model at first leads to overfitting, but beyond a certain point actually improves generalisation performance (Belkin et al. 2019; Nakkiran et al. 2019).

The non-convexity of the loss landscape is strongly related to the "inductive bias" of the network, constituting the prior knowledge about the learnable function classes encoded into a network. The more such inductive biases are encoded into the network, the more heavily encouraged the network is to learn a solution that satisfies these biases—this may be beneficial when data is scarce, but may also lead to overfitting, or the learning of suboptimal solutions (with respect to the true data generating process) (Wilson 2025).

One particular such bias of note, which strongly influences non-convexity, is that of sparsity—both in the information flow of the forward pass, and in the representations a network is encouraged to learn. Overly aggressive sparsification removes the overparameterised flexibility of a network and can increase the risk of getting stuck in suboptimal solutions (Bartoldson et al. 2020; Evci et al. 2020; Gale et al. 2019). Though here again, tricks such as gradient noise, weight decay, and dropout can help to mitigate this risk, or alternative choices of optimiser may be necessary (D'Angelo et al. 2024; Srivastava et al. 2014).

2.2.3 Architectures of Note

Over the decades researchers have invented many clever "architectures" of neural networks whose parameters manipulate information through carefully selected operations to ensure a desired inductive bias — usually resulting in a more efficient learning process in a specific domain. Below we introduce a few commonly used inductive biases, and the architectures which leverage them.

Types of Inductive Biases

- Spatial biases (locality, translation invariance)
- Temporal biases (recurrence, causal interactions between features across time)
- Relational biases (causal interactions between entities)

As we shall see in chapter 5, all of these inductive biases may be applied in parallel to form a network which is strongly predisposed toward forming representations which are sparse, hierarchical, and relational.

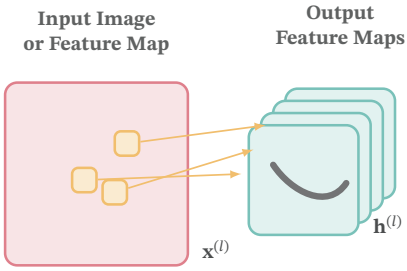
Convolutional Neural Networks

Convolutional Neural Networks (CNNs) apply a spatial inductive bias by encouraging the network to learn features which are invariant to translation. This is achieved through the use of convolution operations, which essentially "slide" a filter over an image, and compute a dot product between the filter and the image at each position.

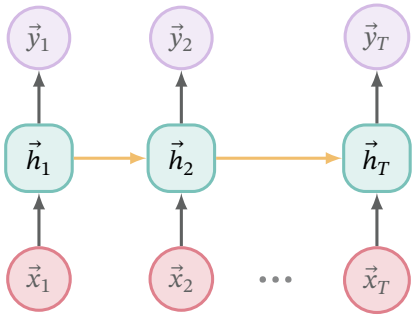
CNNs were the first largely successful architecture in the field of deep learning, and are still widely used today, though other architectures, based on the transformer architecture discussed shortly, have become more prevalent in recent years. There is some debate around the relative merits of CNNs and transformers for image processing, with transformers being more flexible yet still performant, whereas CNNs were directly designed for the task of image recognition (Smith et al. 2023).

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) apply a temporal inductive bias by encouraging the network to learn features which are preserved or built-upon over time. This is achieved through the use of recurrent connections, which allow the network to maintain a "memory" of the previous inputs.



(a) Convolutional neural network, modified from Stutz 2022. Here the frontmost feature map is shown, which is the result of convolving a learned filter across an input tensor. The curved edge suggests that the filter is a simple edge detector, which is a common feature of CNNs (Olah, Cammarata, et al. 2020).



(b) An unfolded recurrent neural network, modified from Love 2024. The recurrent connections are shown in yellow, and the feedforward connections are shown in gray. To avoid problematic gradients, the recurrent connections are typically gated, such that the gradient is not multiplied by the weight matrix at each time step.

Figure 2.2: Two foundational neural network architectures that encode distinct inductive biases: (a) CNNs leverage spatial locality through shared filters that detect features across image regions, while (b) RNNs process sequential data by maintaining temporal state through recurrent connections.

Simply applying the same operation repeatedly leads to the aforementioned gradient vanishing and exploding issues which hamper learning, as $y \propto \mathbf{W}^{\mathbf{T}} x$ for T time steps. To address this, the use of Gated Recurrent Units (GRUs) (Cho et al. 2014) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997b) units was introduced, which make the temporal updates additive rather than multiplicative, whilst still allowing the network to maintain a "memory" of the previous inputs.

2.2.4 Attention Mechanisms and Transformers

Transformers (Vaswani et al. 2017) are a type of neural network architecture which apply a relational inductive bias by encouraging the network to learn features between pairs of embedded inputs through a differentiable routing mechanism known as attention.

Dot-Product Attention

The central building block of modern attention mechanisms is scaled dot-product attention. In general, this is given by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} = \text{softmax}(\mathbf{A})\mathbf{V}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, $\mathbf{K} \in \mathbb{R}^{m \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{m \times d_v}$ are the queries (dimensionality d_k), keys (dimensionality d_k), and values (dimensionality d_v) respectively — their names alluding to the fact that attention may be viewed as a "soft" form of dictionary retrieval. The scaling factor $\sqrt{d_k}$ prevents the dot products from growing too large in magnitude, which would cause the softmax to saturate. The attention matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ captures the similarity between each query and key, and is used to compute the weighted sum of the values.

In practice, for a sequence of n inputs, embedded into a vector space of dimension d_{model}, the attention matrix is simply square $n \times n$ matrix, and the queries, keys and values are typically learned as a linear projection without bias, e.g., $\mathbf{Q} = \mathbf{W}^Q\mathbf{X}$ where $\mathbf{W}^Q \in \mathbb{R}^{d_{model} \times d_k}$.

Causal Masking For sequences with a temporal ordering, it is often desirable to mask the attention mechanism so that the network cannot attend to future inputs. This is achieved by setting the attention weights to $-\infty$ for all future positions (after token j) within the attention matrix, A, such that $\text{softmax}(A) = 0$ for all sequence positions, $i > j$.

Multi-Head Attention (MHA)

Multi-head attention extends this by learning multiple attention functions in parallel, allowing the model to jointly attend to information from different representation subspaces (these notions will be expanded upon in Part II):

$$\text{MHA}(\mathbf{X}) = \text{Concat}(\text{head}_1, ..., \text{head}_h)\mathbf{W}^O$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(\mathbf{X}\mathbf{W}_i^Q, \mathbf{X}\mathbf{W}_i^K, \mathbf{X}\mathbf{W}_i^V),$$

with learned projections $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$.

The Transformer Architecture

Making extensive use of attention mechanisms, a decoder-only transformer architecture (Vaswani et al. 2017) consists of stacked layers (aka "blocks"), each containing:

- Multi-head self-attention,
- Layer normalisation,
- Position-wise feed-forward networks (MLPs),
- Residual connections.

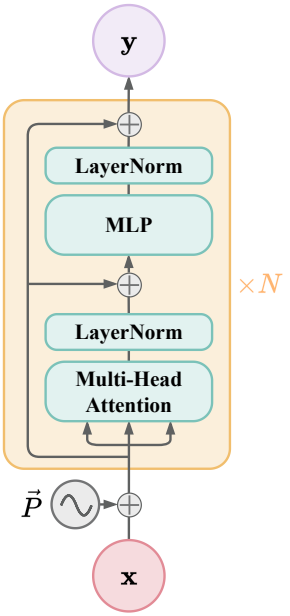


Figure 2.3: Decoder-only transformer architecture consisting of N stacked blocks, in which each "block" contains a multi-head self-attention mechanism, layer normalisation, and feed-forward networks with residual connections, after Vaswani et al. 2017. $\vec{P}$ denotes the positional encoding added to the input tokens.

LayerNormalization (Ba et al. 2016) acts to stabilise the learning process by both centring and scaling neuron activations. It operates on a single token's representation $\vec{x} \in \mathbb{R}^{d_{model}}$, and is given by

$$\text{LayerNorm}(\vec{x}) = \vec{\gamma} \odot \frac{\vec{x} - \text{E}[\vec{x}]}{\sqrt{\text{Var}[\vec{x}] + \epsilon}} + \vec{\beta}$$

where ϵ is a small constant to prevent division by zero, $\vec{\gamma}, \vec{\beta} \in \mathbb{R}^{d_{model}}$ are a learned gain and bias, and the expectation (E) and variance (Var) are taken over the d_{model} components of $\vec{x}$. Applied to a sequence $\mathbf{X} \in \mathbb{R}^{n \times d_{model}}$, the same operation is performed on each row independently. As opposed to other common normalisation techniques, such as batch normalisation (Ioffe et al. 2015), which normalises each neuron over the examples in a batch, layer normalisation normalises each example over its neurons, and is therefore independent of both the batch size and the sequence position.

It is worth noting that the transformer architecture in its original conception contained both an encoder and a decoder, and that the encoder-only architecture is a special case of the full transformer architecture. However, it has been observed that the encoder-only architecture is sufficient for many tasks in practice (Radford et al. 2019), whilst avoiding some complexities of using a decoder.

Input Representation and Tokenisation

For textual input, a transformer first parses a sequence, such as `‘‘The quick brown"` into a sequence of tokens `[‘‘The", ‘‘qu", ‘‘ick", ‘‘brown"]`. In standard applications, the *tokenisation scheme* is determined via the computation of subword statistics to find an optimal vocabulary that balances representation efficiency with model size.

Common approaches include Byte-Pair Encoding (Sennrich et al. 2016), WordPiece (Yonghui Wu et al. 2016), and SentencePiece (Kudo et al. 2018), which iteratively merge frequent character sequences based on corpus statistics.

This procedure yields a finite vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$, which allows the mapping of tokens to indices $t_i \in \{1, ..., |\mathcal{V}|\}$. These indices correspond to rows in the *embedding matrix* $\mathbf{W_E} \in \mathbb{R}^{|\mathcal{V}| \times d_{model}}$, where each row represents a learned dense vector representation of a token. The input sequence is thus transformed into a sequence of embeddings $\mathbf{X}^{(0)} = [\mathbf{e}_{t_1}, ..., \mathbf{e}_{t_n}]^T \in \mathbb{R}^{n \times d_{model}}$.

Since self-attention is permutation-invariant, transformers require explicit positional information. This is typically achieved through *positional encodings* that are added to the token embeddings:

$$\mathbf{X} = \mathbf{X}^{(0)} + \mathbf{P}$$

where $\mathbf{P} \in \mathbb{R}^{n \times d_{model}}$ contains position-dependent vectors. The original transformer (Vaswani et al. 2017) uses sinusoidal positional encodings, while many modern variants learn these encodings directly (Devlin et al. 2018; Radford et al. 2019).

A similar tokenisation procedure can be applied to non-linguistic inputs. For images, the most common approach is to divide the input into a grid of non-overlapping patches (Dosovitskiy et al. 2021), where each patch is flattened and linearly projected to form a token embedding. Alternative approaches include using convolutional layers to extract token representations (Xiao et al. 2021), or operating directly on pixels (T. Chen et al. 2022). For audio signals, common tokenisation strategies include extracting fixed-length frames from spectrograms (Gong et al. 2021), learning discrete codes via vector quantisation (Baevski et al. 2020), or using convolutional modules to produce token sequences (Gulati et al. 2020). These methods demonstrate the generality of the transformer, requiring only that inputs be representable as sequences of embeddings.

As we shall see in Part I of this thesis, the core idea of attention mechanism — that of differentiably routing information between subsets of

inputs or hidden representations — is a powerful tool for building interpretable and compositional representations. Indeed, variants (Locatello, Weissenborn, et al. 2020) and derivatives (Z. Wu et al. 2023) of the transformer architecture have seen considerable success in object-centric learning, which will be our focus in the immediate chapters.

The neural architectures and learning principles outlined in this chapter provide the foundation for both approaches explored in this thesis. While the scaling paradigm has demonstrated that minimal architectural constraints combined with vast amounts of data can produce remarkably capable systems, fundamental questions remain about interpretability, systematic generalisation, and sample efficiency. These concerns motivate our exploration of an alternative path in Part I: engineering interpretable structure directly into neural architectures.

Part I

Object-Centric Deep Learning

CHAPTER 3

Object-Centric Learning

We begin our technical work by exploring the engineered-interpretability agenda discussed in chapter 1. In particular, we explore inductive biases which encourage neural networks to represent and reason over "objects".

3.1 What is an object?

To ground our discussions we must first decide on what we mean when we say that something is *an object*. In order to facilitate the possibility of agreement, we shall pragmatically do away with the level of metaphysical rigour a philosopher might demand. This is permissible as our notion of objectness will serve *only* to guide the development of agents whose reasoning resembles that of natural intelligences — these were not cleverly designed but arose from blind evolution, developing the ability to recognise, represent and reason about objects in order to more effectively pass on their genetic material[1] (Shepard 1984).

Most, but not all, entities which constitute meaningful objects for agents in the physical world are likely to be easily agreed upon. However, the definition of object which we adopt is more akin to that of symbols or concepts, such that an observer may distill the task-relevant properties of some physical object into an internal symbolic representation, which we would still consider to be an object. In particular, following van Steenkiste, Greff, et al. 2019 and Greff, van Steenkiste, et al. 2020, we shall consider objects to be computational primitives capturing abstract patterns on input data (or internal representation) which are:

- **Modular:** Objects are sub-components of the input or representation which suffice to explain most task-relevant aspects of the original input, and are largely independent of one another.
- **Dynamic:** Objects are task, goal and knowledge dependent. Part-whole hierarchies may constitute objects themselves. Task information is required to decide which object-wise decomposition is appropriate at a given time.

[1] A process quite orthogonal to erudite philosophical enquiry.

- **Consistent:** Object representations are often grounded in physical reality, and should nominally possess a stability and consistency commensurate with their physical counterparts. Even abstract objects such as the numeral "7" must display consistency in their representation to be useful for reasoning[2].

Together these properties underline a distributed notion of objectness which captures all aspects of a representation amenable to symbolic processing in ANNs.

The framing of the aforementioned object properties in terms of task-dependence is well understood from the view-point of affordances introduced by Gibson 1979. With the key idea being that agents interacting with their environment, and wishing to build useful predictive models, will learn to associate properties of immediate relevance to recurring perceptual themes. In particular, "The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill." (Gibson 1979). As such, affordances too are distinct from physical properties in that "they have to be measured relative to the animal" — this includes factoring in its internal state (hungry, tired, etc.), its mode of perception (monocular vision, sonar etc.). As such, affordances guide how object representations will be learned by an agent — those representations which have predictive power or facilitate beneficial cognition.

As an example, containership may be represented quite distinctly even when associated with the same physical object: A magician seeking to stow a white rabbit may *perceive* a top hat differently from a dapper groom — the hat affords different things to the magician and the groom. Furthermore, as affordances capture utility in some sense, they can be used to motivate the selection of salient objects across all of an agent's percepts. The need for this is clear even in humans, who are able to store around 3-9 objects at a time in working memory (Fukuda et al. 2010; Miller 1956). Viewed through the lens of affordances, we might say that attention is directed on the basis of perceived affordances, which are by definition conditioned

[2] In systems where representations are distributed across local populations, as in biological brains, and plausibly in modular artificial networks, consistency can be motivated on two grounds: i) *Inter-region communication*: if distinct modules must coordinate, then the encoding used by each must remain stable over time-scales much shorter than the rate of learning, lest downstream readout become meaningless (Gallego et al. 2020; Rule et al. 2019); ii) *Learnability*: if a representation drifts faster than a learning process can track, no stable correspondence between the representation and the underlying stimulus can be maintained.

on current goals and knowledge. This actor-centric view of "meaningful" representations will be particularly important in Part II when we attempt to reverse-engineer representations that form naturally in large ANNs.

For a designed ANN to capture objects based on affordances, we propose that it would minimally require:

- The ability to associate task-relevant properties to objects. In particular, salient static properties (e.g. colour, shape) as well as dynamical properties (mass, etc.) rather than trivial visual features (texture etc.).
- The ability to rank and filter objects by affordance for a given task. In general, this would require both task-understanding and a capacity for internal simulation and planning to infer potential relevance of a given object-wise decomposition.

The first of these points, the ability to associate task-relevant properties to objects, will be particularly important for the development of object-centric models. As we shall see in chapter 4, the ability to separate pose from appearance is crucial for the development of a structured representation of objects — at least for the tasks we will consider in chapter 5. The second point, the ability to rank and filter objects by affordance for a given task, will be important for the relational reasoning tasks we will consider in chapter 5.

Beyond decomposing a scene into separate object slots, we must also consider how information is organised *within* each slot—that is, the degree of representational structure. A structured representation is one in which distinct properties of an input are encoded in identifiable, separable components, making them amenable to downstream inspection and manipulation. We shall ground these abstract considerations concretely in section 3.4, after introducing the technical machinery of object slots and disentanglement, by examining how different levels of representational structure manifest for a simple scene of geometric objects.

3.2 Object-Centric Representations

3.2.1 Representing Objects

We are interested in Artificial Neural Networks (ANNs) whose architectures are inductively biased in ways which encourage them to learn to represent objects "distinctly". By using ANNs for perceptual processing we ensure

that gradients can be passed end-to-end from downstream networks, for example in Deep Reinforcement Learning (RL) via policy gradient methods Williams 1992, providing a great degree of flexibility and retaining the ability to leverage large amounts of data. The subset of Deep Learning approaches that attempt to acquire such representations are typically referred to as performing Object-Centric Learning (OCL).

Though there are other ways of representing objects in ANNs, a family of approaches known as “slot-based" have shown the most promise in meeting the representational requirements of objects outlined in the previous section, whilst still being trainable through gradient descent (Greff, van Steenkiste, et al. 2020; van Steenkiste, Greff, et al. 2019).

These models utilise weight-sharing across multiple slots which are used to represent objects, ensuring a common representational format. For example, RNNs may be viewed as slot-based models with a single slot that is updated over time — though the resulting slot’s representation would either capture multiple objects, or not capture all objects; neither of which are desirable. In fact, at the level of slots there are three properties which we wish our models to possess:

- **Consistency:** If superficial aspects of a model’s input are perturbed (for example, adding homogenous noise across colours channels), then the representation of objects captured by slots should suffer a relative perturbation no greater than that of the inputs.
- **Alignment:** (For videos) slots should represent the same objects across subsequent time steps — assuming continuous dynamics and object uniqueness.
- **Exclusivity:** Each slot should represent at most one object.

We will frame our discussion of such approaches in terms of the taxonomy for slot-based models proposed by Greff, van Steenkiste, et al. 2020; van Steenkiste, Greff, et al. 2019, which breaks down existing approaches into four categories:

- **Instance Slots:** All slots are generic and can represent any object. Such slots are the most general from a representational perspective, but the symmetry between slots must be broken — for example by allowing inter-slot interactions.
- **Sequential Slots:** Utilise recurrence to parse a scene one object at a time. Symmetry breaking may occur by imposing an order on the

scene processing, and representational capacity can potentially be increased by performing more iterations. Slots may not be simultaneous accessibly at a given timestep, but this can be addressed by conditioning downstream components on an aggregated, or selectively routed, representation of the slots.

- **Spatial Slots:** Each slot is associated with a particular spatial location, thus breaking the symmetry. Alignment is non-trivial in such a model as an object moving across the input will pass through the receptive fields of different slots. However, position information is explicitly associated with every slot, which may be useful downstream.

- **Category Slots:** Slots explicitly specialise to represent certain object categories (e.g., based on shape), facilitating symmetry breaking. Such models cannot represent multiple instances of a particular object simultaneously, unless used in conjunction with another method.

3.2.2 Models

There have been many models proposed to learn Object-Centric representations using a variety of approaches. Weis et al. 2020 discuss a number of these approaches and group them by the fashion in which latents (the internal representations, i.e., vectors, which we choose to consider interesting) are learned, and whether or not the latents are structured ("factored"), whilst Greff, van Steenkiste, et al. 2020 provide examples of models matching each slot type. We provide a more thorough review of such models in section A.2, framed in terms of the considerations outlined above. Researchers have proposed models covering each of our criteria, but at the time when our work was carried out none of the existing models did so without various significant trade-offs.

3.2.3 Slot-attention

With an eye toward overcoming the shortcomings of existing models, we chose to extend Slot-Attention (SA), proposed by Locatello, Weissenborn, et al. 2020. SA utilises the attention mechanism popularised by transformers, as discussed in section 2.2.4, to elegantly realise an instance-slot based architecture with simple inter-slot interactions.

Slot Attention Module

The Slot-Attention (SA) module, is an instance slot type model, where all slots are able to represent any object, and there is no fixed ordering of slots. The module takes input features (typically from a CNN) of dimension $N \times D_{input}$ and maps these to K output slots with dimension D_{slots}. The slots compete with each other via an iterative softmax attention mechanism, with more iterations, $i \in 1 \ldots I$, resulting in increasingly refined attention maps.

The mechanism utilises standard dot-product attention as discussed in section 2.2.4, with learned linear transformations, k, q and v, used to map the features and slots to a common dimension D (with $D = D_{slots}$ in our experiments). The softmax attention operation is then

$$\text{attn}_{i,j} = \frac{e^{\mathbf{M}_{i,j}}}{\sum_l e^{\mathbf{M}_{i,l}}} \quad \text{where} \quad \mathbf{M} = \frac{\mathbf{k}(\text{inputs}) \cdot \mathbf{q}(\text{slots})^T}{\sqrt{D}} \in \mathbb{R}^{N \times K}.$$

Instead of computing updates directly using the attention scores, a weighted mean over the attention scores of all slots is used to update each slot's state, as per

$$\text{updates} = \mathbf{W}^T \cdot \mathbf{v}(\text{inputs}) \in \mathbb{R}^{K \times D} \quad \text{where} \quad \mathbf{W}_{i,j} = \frac{\text{attn}_{i,j}}{\sum_{l=1}^{N} \text{attn}_{l,j}}.$$

This helps to stabilise the mechanism. The resulting updates are then optionally passed through a GRU which takes slots from the previous iteration as its input state.

These updates are applied to the slots via a residually connected MLP. Locatello, Weissenborn, et al. 2020 also find that applying LayerNorm (Ba et al. 2016) to slots, updates, and inputs, speeds up training (see subsection 2.2.4).

Slots are then passed through a Spatial Broadcast Decoder (described below) to produce a full-size reconstruction and accompanying mask for every object (with weight-sharing). The masks are then normalised and the image is reconstructed by multiplying each reconstructed object with its corresponding mask, and adding these up.

Spatial Broadcast Decoder

The Spatial Broadcast Decoder (SB) introduced by Watters, Matthey, Burgess, et al. 2019 encourages disentanglement of latent representations,

$\vec{z}$ by tiling (copying) the latent, $\vec{z}$, across a 2D grid and appending $x\&y$ coordinate information (ranging from -1 to 1 for each dimension).

The resulting grid has the same size as the desired output image, and so the decoder does not perform any upsampling, and is not required to propagate spatial information through multiple layers. Additionally, the rendering of objects for an ideal latent is reduced to a local thresholding operation over a subcomponent of $\vec{z}$. This encourages the learned representation to contain disentangled information about the objects present in a scene, and in particular their position.

Watters, Matthey, Burgess, et al. 2019 show that when used in conjunction with a VAE (Diederik P. Kingma et al. 2014) the SB decoder learns to disentangle position information in the latent space, and that the mapping from ground truth positions to latent positions is almost exactly affine. This inductive bias targets position alone, however; rotation and scale are not architecturally factored from appearance. We return to this limitation in section 3.4, and address it in chapter 4. The authors do note that there may be tasks in which an absolute coordinate system may not be preferable, and in which relative information about visually distinct features that co-occur could instead be more effectively learned by a standard upsampling deconvolutional decoder.

3.3 Disentangled Representation Learning

Outside of the OCL literature much effort has gone into creating ANNs whose internal representations are "disentangled"—clearly capturing environmental *factors of variation* in as few *distinct* latent variables as possible. It is often argued that such representations should enable superior downstream performance or sample-efficient fine-tuning (Bengio, Courville, et al. 2014), findings which have sometimes been vindicated (van Steenkiste, Locatello, et al. 2020) and other times failed (Locatello, Stefan Bauer, Lucic, et al. 2019).

Another claim is that disentangled representations are more capable of achieving out-of-distribution generalisation than their counterparts, which has been demonstrated in some instances (Träuble et al. 2021) and failed in others (Higgins et al. 2016; Montero et al. 2020).

Locatello, Stefan Bauer, Lucic, et al. 2019 prove that learning disentangled representations is impossible without inductive bias or weak supervision. Inductive bias can distinguish between generative models which capture factors of variation equally well, but may demonstrate different

levels of disentanglement. However, unsupervised model selection, e.g. on the basis of existing disentanglement metrics, has shown limited success, and weak-supervision or post-hoc model correction with labels thus appear critical (Locatello, Stefan Bauer, Lucic, et al. 2019; Locatello, Stefan Bauer, Lucic, et al. 2020; Träuble et al. 2021). In practice, the role of weak-supervision may best be played by down-stream tasks which would require generalisation and composition of objects in order to perform well. An alternative approach is to impose strong priors on structure of the latent space befitting a desired decomposition, as demonstrated by Kori et al. 2024 in the context of OCL.

One particularly interesting finding of relevance to our investigations comes from Higgins et al. 2016 and Montero et al. 2020 who find that whilst the latent representations learned by β-VAEs generalise to combinations of generative factors not encountered during training, simple transpose convolutional decoders do not reconstruct well in these regimes. Strikingly, Montero et al. 2020 demonstrate that this is true even of decoders which are directly trained on the generative factors, and that these in fact perform worse than their learned-latent counterparts.

From this two key insights may be gleaned. First, the flexibility of learning a latent distribution seems to benefit models, in so far as downstream reconstructions are concerned, in spite of the presence of an "information bottleneck" and regularisation terms. Second, a disentangled latent representation may not suffice to facilitate down-stream generalisation, and instead, an explicitly structured representation which can be leveraged by structured downstream networks may be needed.

Indeed, Leeb et al. 2021 demonstrate this to be the case on generative modelling tasks when using a decoder which hierarchically reconstructs images by using different subsets of the latent representation. We will explore similar ideas in chapter 5.

To frame our discussion of disentanglement we refer to the properties which a disentangled representation should possess, motivated by Eastwood et al. 2018:

- **Disentanglement** (also modularity): The extent to which a factor of variation is explained by (as few as possible) latent-code elements.
- **Completeness** (also compactness): The extent to which latents capturing factors of variation are only correlated with a single factor.
- **Explicitness:** How direct the mapping between factors of variation and latent codes is; E.g. how linear and well-normalised. This notion is relevant in Part II where linearity of "concepts" is often assumed.

Intuitively completeness is more interesting than disentanglement, as there are scenarios in which using more than one element of a latent code may be required to capture a factor of variation (e.g. RGB colour). In practice however, metrics which primarily measure disentanglement, rather than completeness, have been shown to correlate more strongly with downstream performance van Steenkiste, Locatello, et al. 2020; this may be a mere consequence of the fact that such metrics also benefit from more explicit representations (owing to how they are computed), which are likely to lead to superior downstream performance on simple tasks.

3.3.1 Measuring Disentanglement

A number of metrics have been proposed to quantify the extent of disentanglement in models. We shall see that none of these are sufficiently complete to encapsulate all desirable aspects of disentanglement robustly, and as such, a mixture of qualitative evaluations and quantitative metrics is required.

Locatello, Stefan Bauer, Lucic, et al. 2019 and Zaidi et al. 2021 have performed extensive comparisons of existing disentanglement metrics. On the basis of their findings, we chose to apply the Disentanglement, Completeness, Informativeness Scores (DCI Scores) in our investigations Eastwood et al. 2018. Whilst a number of other metrics perform a similar function to the DCI Scores, Zaidi et al. 2021 show that these tend to suffer more severely from susceptibility to noise or a change of basis in the disentangled representation (i.e. rotating the latent space).

The DCI Scores are one of a number of metrics which use auxiliary classifiers (or regressors) to determine which, if any, factors of variation are captured in the latent representation of a model. In particular, the "feature importances"[3] of the learned classifier can be used to assess which sub-components of the latent code captured variation in generative factors present in the training distribution. It is important to note that the choice of auxiliary model can significantly impact the behaviour of the metric, as well as the ease with feature importances can be extracted.

The DCI Scores consider the matrix of feature importances resulting from such a fit (see Figure 4.5 for an example), with row-wise entries indicating

[3] The extent to which a given code variable (i.e. index of a latent vector) contributes to the classification decisions of the auxiliary model.

the contribution of each code variable toward predicting a ground truth factor.

By taking (one minus) the entropy of each row, and averaging across all features, the DCI Disentanglement score measures the extent to which any generative factor is explained by a single latent code. Conversely, taking (one minus) the entropy of each column, and averaging, the DCI Completeness score measures the extent to which any code variable captures information pertaining to only one generative factor.

In addition to criticising the Mutual Information Gap (MIG), which is a popular information-based disentanglement metric, Watters, Matthey, Burgess, et al. 2019 discuss key shortcomings of disentanglement metrics in general. They note that 'there is a fundamental trade-off between how much information a representation contains and how well-structured the representation is', and that consequently a single scalar is ill-suited towards measuring all beneficial aspects of a representation (Eastwood et al. 2018 also acknowledge this) — similar ideas are relevant to superposition, as we will explore in Part II. Nonetheless, Locatello, Stefan Bauer, Lucic, et al. 2019 observe that the modularity of a representation is well-captured by the DCI Disentanglement Score, whilst the MIG measures compactness well — though this latter finding is undermined by Zaidi et al. 2021.

Ultimately, these metrics are useful, but should be utilised in conjunction with qualitative evaluations to rule out scenarios in which metric values do not reliably reflect the information content of latent codes.

3.4 Grounding Representational Structure

To make the preceding concepts concrete, consider the simple scene of geometric primitives depicted at the top of Figure 3.1, inspired by the CLEVR dataset (J. Johnson et al. 2017). Each object possesses *intrinsic* properties that define its identity (shape, colour, material, size) and *extrinsic* or *pose* properties that specify its spatial configuration (position, orientation, scale).

When the red cube is translated and rotated between the two scenes, only its pose changes; its intrinsic properties remain invariant. If pose is entangled with identity, then a simple translation can corrupt shape or colour information, making downstream reasoning brittle — determining that "the red cube is to the left of the blue sphere" requires access to relative positions that have not been conflated with other properties.

The bottom of Figure 3.1 illustrates how the red cube's representation changes across four levels of structure. In a symbolic encoding, only the

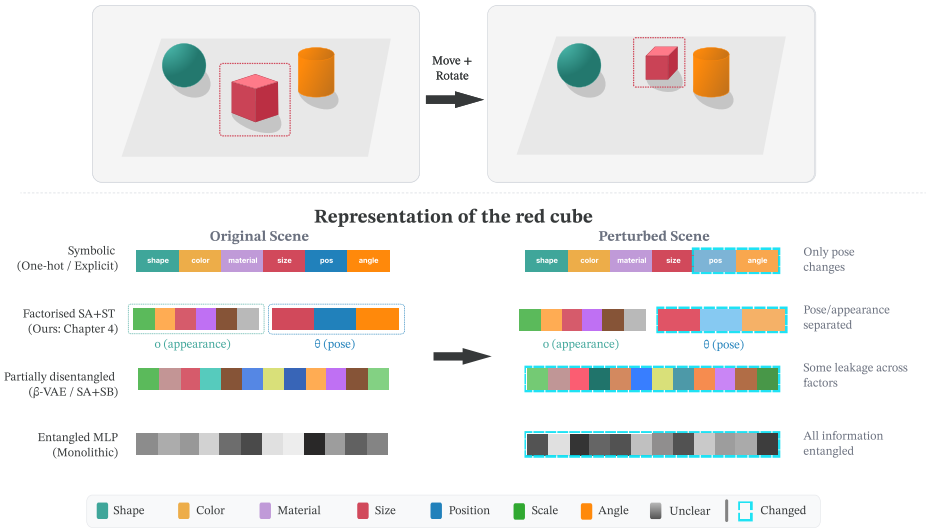


Figure 3.1: Top: Two configurations of a CLEVR-style scene in which a red cube changes position and orientation. Each object possesses intrinsic properties (shape, colour, material, size) and extrinsic pose properties (position, orientation, scale). **Bottom:** How the red cube's representation changes across four levels of representational structure. Dimensions that changed between scenes are highlighted with a cyan border. In more structured representations, only the pose-related dimensions change; in entangled representations, the entire vector is affected. Of course, this is a caricature of what actually happens inside of neural networks, and presupposes that in all cases our network has managed to capture the red cube *exclusively* in the vector, or slot, we are focusing on.

pose dimensions change. In a factorised SA+ST representation — the approach we introduce in section 4.1 — the pose sub-vector $\vec{\theta}$ updates while the appearance sub-vector $\vec{o}$ is preserved. In a partially disentangled representation, such as those produced by β-VAEs (Higgins et al. 2016) or SA with a Spatial Broadcast Decoder, several non-pose dimensions are also affected. And in a fully entangled, monolithic representation the entire vector shifts. This spectrum — from rigid symbolic encodings to fully entangled ones — mirrors the interpretability–flexibility trade-off introduced in Figure 1.1, and will guide the remainder of Part I.

Chapter 4

Enforcing Structure in Latent Representations

Having argued in Part I's background that interpretable representations should decompose visual scenes into objects to support downstream reasoning, we now address the representational foundation underpinning such decompositions. This chapter asks to what extent architectural constraints—specifically, factorising pose from appearance via spatial transformers—can promote disentangled object representations within a flexible slot-based framework. We find that our SA+ST architecture successfully separates pose information into dedicated transformation vectors, improving disentanglement without sacrificing reconstruction or segmentation quality.[1]

Object-Centric Learning (OCL) methods can be used to decompose visual scenes into distinct object representations or "slots." While these methods successfully discover objects in scenes, most such approaches entangle object properties—shape, colour, scale—with spatial pose information. This entanglement undermines interpretability, and may hamper downstream reasoning modules by making compositional computations more challenging to implement. Though the true test of this limitation lies in performing such tasks (which we explore in the next chapter), we must first address the representational foundation.

We propose combining Slot-Attention (SA), a flexible object discovery mechanism, with Spatial Transformer (ST) networks that explicitly separate pose from appearance. Our SA+ST architecture enforces a structured factorisation: object properties reside in dedicated latent vectors while spatial transformations are captured in separate vectors.

This architectural bias reflects our hope that carefully engineered structure can enhance interpretability without sacrificing the flexibility needed for real-world applications, setting the stage for the end-to-end reasoning experiments in chapter 5. We probe this claim on a modified Spriteworld benchmark—chosen as it exposes *per-object* ground-truth generative factors alongside segmentation masks and temporal episodes, as motivated in subsection 4.1.2, whilst being of a level of visual complexity commensurate with the literature.

Related Work

OCL models differ not only in how they discover slots, but in how they *decode* them—and decoder choice directly shapes the structure latents acquire. The Spatial Broadcast Decoder (Watters, Matthey, Burgess, et al.

[1] The experiments presented here are mostly unpublished, with some of the disentanglement analyses appearing in Spies, A. Russo, et al. 2022.

2019) encourages position information to separate from appearance but imposes no hard factorisation (see section 3.2.3); structured-latent models instead pair slots with decoders that explicitly consume pose (surveyed in section A.2). We situate SA+ST between these extremes.

A family of structured-latent models—SPACE (Lin et al. 2020), SPAIR (Crawford et al. 2019), SQAIR (Kosiorek et al. 2018), SCALOR (Jiang et al. 2019), and TBA (Zhen He et al. 2019)—*prescribe* a decomposition into dedicated position, scale, depth, and appearance latents that are then composed through a spatial transformer (section A.2). By contrast, vanilla SA (Locatello, Weissenborn, et al. 2020) uses unstructured instance slots decoded by a Spatial Broadcast Decoder, gaining flexibility at the cost of any explicit pose factorisation.

The disentanglement literature (reviewed in section 3.3) clarifies why decoder architecture matters: Watters, Matthey, Burgess, et al. 2019 show that the Spatial Broadcast Decoder induces an approximately affine latent-to-pixel map in *position* only, while Montero et al. 2020 demonstrate that deconvolutional decoders can block factor generalisation even when latents are disentangled. Architectural structure in the decoder, not just the latent space, plays a decisive role.

SA+ST explores the middle ground, retaining SA's flexible, attention-based instance-slot discovery (section 3.2.3) while replacing the Spatial Broadcast Decoder with a spatial transformer decoder (Jaderberg et al. 2016) that *architecturally* factorises pose from appearance. Unlike the structured-latent models above, SA+ST does not prescribe a separate latent variable for each spatial attribute; a learned MLP head maps each slot to transformation parameters, letting the model decide which pose dimensions to exploit (see the spectrum in section 3.4). The experiments that follow test whether this architectural factorisation improves disentanglement relative to the Spatial Broadcast Decoder baseline, without sacrificing the ease of learning that makes SA attractive.

4.1 Methodology

4.1.1 Slot Attention with Spatial Transformers

As discussed in section 3.2.3, SA encodes objects in images into slots, which ideally capture the full-information required for a SB decoder to reconstruct the original image (in an auto-encoding setup). In order to impose structure on the learned representation of the Slot-Attention module, we use a

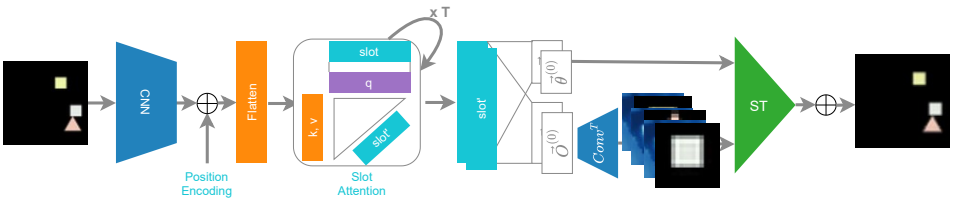


Figure 4.1: The SA+ST architecture. The outputs of the SA module, $\vec{z}_i$ are transformed with two separate MLPs to obtain: i) transformation vectors, $\vec{\theta}_i$, applied by the ST (across channels); ii) object representations, $\vec{o}_i$. These are combined by the decoder to reconstruct the input image.

different decoder which renders small object "glimpses" and then upsamples and transforms these. To this end we utilise Spatial Transformer (ST)[2] (Jaderberg et al. 2016) to decode masks and glimpses, which are combined similarly to the SB decoder used by Locatello, Weissenborn, et al. 2020.

ST layers consist of two differentiable components. First, a localisation net (MLP) which produces a six-element vector, $\vec{\theta} \in \mathbb{R}^6$, parameterising an affine transformation in 2D. Second, a sampling stage (across channels) which can be used to up- or down-sample feature embeddings. This stage generates a grid with dimensions matching the desired output and distorts the grid by applying the transformation produced by the localisation net. As per the implementation from Jaderberg et al. 2016, we use bilinear sampling to map the input features to the transformed grid.

The ST layer is used to upsample and transform the outputs, or "glimpses" of a simple decoder, as in Zhen He et al. 2019; Kosiorek et al. 2018; Lin et al. 2020. However, we do not presently rely on a depth variable to perform reconstructions and handle occlusions; instead we also apply the transformations to alpha masks produced by the decoder, and then perform a weighted sum over objects and their masks — exactly as in the original SA+SB architecture. The resulting model is shown in Figure 4.1. A forward pass of the model is also sketched in algorithm 1.

In order to prevent the collapse of the slot attention mechanism early on in training, two separate MLP "heads" are used to obtain the transformation vectors, $\vec{\theta}^{(i)}$ — as well as the object representations, $\vec{o}^{(i)} \in \mathbb{R}^{26}$, which are combined (for testing) into the vector $\vec{z'}^{(i)} = [\vec{o}^{(i)}, \vec{\theta}^{(i)}]$, with dimensionality

[2] Transformer in the sense of applying a transformation — not the recent stacked self-attention architectures popular in NLP.

equaling that of the slots. Here the index i highlights the fact that the two MLP heads are applied across slots individually. To further stabilise early training, we also add gaussian noise to the model's inputs[3].

Although we do not constrain the form of $\vec{\theta}$ which the model may learn, it is worth noting that the form of a skew-free 2D transformation is given by

$$\psi = \begin{bmatrix} s_x \cos\phi & -s_x \sin\phi & \delta x \\ s_y \sin\phi & s_y \cos\phi & \delta y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

where s_x, s_y parameterise scale transformations, ϕ dictates the rotation, and $\delta x, \delta y$ are translations. As our datasets do not contain skewed objects, we may expect the learned transformations to approximately take this form.

Algorithm 1: Forward pass of SA+ST

Data: Images, $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$

$\vec{\rho} = \text{Encoder}(\mathbf{x})$

$\text{slots} = \text{SlotAttention}(\vec{\rho})$

$\vec{\theta} = \text{MLP}_\phi(\text{slots})$

$\vec{O} = \text{MLP}_\zeta(\text{slots})$

$\mathbf{g}, \mathbf{m} = \text{Decoder}(\vec{O})$ `// Glimpses and masks for each slot`

$\mathbf{g}, \mathbf{m} = \text{SpatialTransformer}(\mathbf{g}, \mathbf{m}, \vec{\theta})$ `// Upsample`

$\mathbf{g} = (\mathbf{g} - 0.5) \times 2$ `// Transform to (−1,1)`

$\mathbf{m} = \text{softmax}(\mathbf{m})$ `// Normalise across slots`

$\hat{\mathbf{x}} = \sum \mathbf{g} \odot \mathbf{m}$ `// Combine for reconstruction`

4.1.2 Datasets and Training

Our choice of benchmark deserves brief justification in the context of the literature. Evaluations of SA and its predecessors typically rely on the *Multi-Object Datasets* suite—Multi-dSprites, Tetrominoes, ObjectsRoom and CLEVR6 (Kabra et al. 2019)—each optimising for a different axis. Namely, scene clutter (CLEVR6), large object counts (Tetrominoes), texture and lighting (ObjectsRoom) or factor-of-variation coverage (Multi-dSprites). The hypothesis under test here, however, is specifically that

[3] With mean 0 and standard deviation 0.05. Inputs are scaled to $[-1, 1]$.

ST-based decoding disentangles pose ($\vec{\theta}$) from appearance ($\vec{o}$); probing this claim requires four properties simultaneously: (i) *per-object* ground-truth generative factors, so that slot-wise disentanglement metrics can be computed (subsection 4.1.3) (ii) ground-truth segmentation masks, so that ARI remains directly comparable with SA (Locatello, Weissenborn, et al. 2020) baselines; (iii) a small, controlled object count (1–4), so that representation-level conclusions are not confounded by segmentation failure modes that dominate in higher-clutter scenes; and (iv) temporal episodes, for experiments with aligning slots over time. Among widely-used object-centric simulators, Spriteworld (Watters, Matthey, Bosnjak, et al. 2019) is the only one to supply all four simultaneously, though supporting ARI mask generation required some modifications. The relational-reasoning benchmarks adopted in chapter 5 are chosen under a complementary rubric, which we motivate there.

As such, all experiments were carried out on datasets generated from the RL Spriteworld simulator (Watters, Matthey, Bosnjak, et al. 2019), modified to include ground-truth object masks and generative factors in a format matching the Multi-Object Datasets from Kabra et al. (2019). All episodes in the resulting dataset are 20 steps long, and are selected to feature no empty frames (as the simulator allows objects to go out of frame). Every episode consists of between 1 to 4 objects, which may be either triangles, circles or squares. The angle, scale and colour of these objects was initialised randomly within a constrained continuous range. It is worth noting that due to rotational symmetries, different angles (from the perspective of the simulator, and thus labels in the dataset) can give rise to identical frames — circles provide the clearest example. More details on the dataset are given in section B.1.

All models were trained for 250,000 steps[4] on single GPUs[5] with batch sizes of 64. We utilise the MSE loss, $L = ||\vec{x} - \hat{\vec{x}}||$, in conjunction with the Adam optimiser (Diederik P Kingma et al. 2015), using the same hyperparameters and schedule as Locatello, Weissenborn, et al. 2020. Models were implemented in Python (Van Rossum et al. 1995), with extensive use of Jax (Bradbury et al. 2018). The Haiku (Hennigan et al. 2020) and Optax (Hessel et al. 2020) libraries provided neural network abstractions and optimisers respectively.

[4] Locatello, Weissenborn, et al. 2020 trained for 500,000 steps, but also observed significant diminishing returns after the first 100,000 steps or so.

[5] NVIDIA Quadro GTX6000 GPUs with 24GB of VRAM.

4.1.3 Evaluating our Approach

In line with previous work we utilise the reconstruction MSE, as well as the Adjusted Rand Index (ARI) over masks (ignoring background) (Hubert et al. 1985; Rand 1971), to assess the performance of our models. The ARI is a clustering metric with values ranging from 0 (random assignment) to 1 (perfect clustering). In our case, the ARI measures the quality of instance segmentation provided by the model's masks, by treating every pixel as a point, with its cluster assignment given by the mask with the highest value at that point. Assuming that the decoder masks are contiguous, this largely ensures that individual decoder masks attend only to individual objects. Whilst this assumption holds strongly in practice, the ARI itself is agnostic as to which decoder mask attends to an object's ground truth mask, so long as the entire object is attended to by the same mask. ARIs were calculated using an implementation modified from Kabra et al. 2019.

Disentangled Representations

Metrics for evaluating how well latent representations reflect ground truth factors of variation were introduced in Section 3.3.1, but these were discussed in the context of models such as VAEs where a single latent code contains all information about an image.

When utilising slot-based representations, multiple latent vectors encode information about the image, and it is not always known which latent corresponds to which feature. Racah et al. 2020 were able to apply the DCI Scores by concatenating all latent vectors $\vec{z}^{(i)}$, and providing the resulting feature vector, $\mathcal{Z}^{(0,\dots,k)} = \left[\vec{z}^{(0)}, \dots, \vec{z}^{(k)}\right]$, as the input to their auxiliary classifiers. Such an approach relies on the specialisation of specific slots to certain objects classes — this ensures fixed ordering of latents corresponding to the ground truth factors they encode. More concretely, the difficulty arises from the need to extract feature importances from auxiliary models (as these are used to calculate the DCI Scores). A typical auxiliary classifier $\mathcal{C}(\mathcal{Z})$, will not be invariant to permutations over slots, such that $\mathcal{C}(\mathcal{Z}^{(0,1,\dots,k)}) = \mathcal{C}(\mathcal{Z}^{(1,0,\dots,k)})$, and so we cannot expect it to classify well, nor to learn meaningful feature importances.

This requisite slot specialisation does not occur in instance-slot based architectures, and as such, recent papers in the area have avoided the application of disentanglement metrics entirely. The same issue holds in SA, as the module ensures that all slots are initialised using the same distribution (or learned embeddings in Löwe et al. 2020).

Broadly speaking, there are two ways of addressing this:

1. **Permutation Invariant Classifier:** Applying a permutation invariant model such as DeepSets (Zaheer et al. 2018) addresses the point above but introduces a new issue. Namely, that the extraction of feature importances for specific components $z_i \in \vec{z}^{(j)} \in \mathcal{Z}$ becomes non-trivial.

2. **Perform Classification Over Slots:** If instead of concatenating all slots for a given image, we pass each slot individually to the auxiliary model, then permutations are no longer an issue. However, we need a way of associating ground truth labels with specific slots; i.e. a way of knowing which object a given slot is representing. There are two ways of achieving this.

   First, as proposed by Dang-Nhu et al. 2021, one can attempt to find sensible assignments by permuting slot-factor pairs to maximise the metric of interest through an EM procedure. This is computationally intensive and requires access to the simulator at test time in order to initialise effectively. Second, attention maps or decoder masks can be used to match slots to ground truth objects. Such an approach will only work if ground truth masks are available, and attention maps are not diffuse; these criteria are met in our case, and as such we use the attention maps to perform the slot-object pairing.

In order to ensure that our approach is applicable in both reconstruction and contrastive training regimes, we do not rely on the existence of a decoder, and instead use the attention maps to perform the slot-object pairing. In particular, we use openCV (Bradski 2000) to find contours in the normalised attention masks, and then apply an adaptive threshold to establish whether there is: i) exactly one contour in the map, and ii) whether the contour area is less than a fixed fraction of the total image (a heuristic for discarding background slots).

The centre position of the resulting contour is then used to match the slot to a ground truth object[6], the properties of which are assigned as labels to the corresponding slot. Figure 4.2 shows an example of this assignment procedure.

[6] This assignment can be toggled to be exclusive, either on the basis of the dominant attention saliency associated with the contour, or on the distance to the nearest object. All results presented here do not enforce exclusivity of assignment.

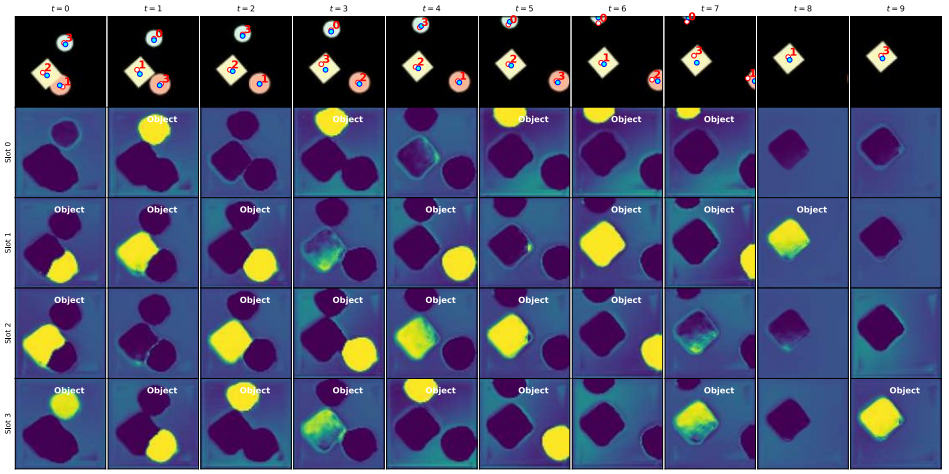


Figure 4.2: An example of the cv2-contour based slot-assignment procedure (here for SA+ST). Columns correspond to consecutive time-steps $t = 0, \ldots, 9$ of a single rollout. Slot rows display the attention map; a slot is tagged `Object` when a single contiguous contour below the background-area threshold is found, and the contour centroid is matched to the nearest ground-truth object to inherit its factor labels. This automated slot-object matching procedure allows us analyze disentanglement of the slots by comparing against the ground truth factors of the objects they capture.

Owing to the asynchronous and one-time execution of the procedure, as well as its being easily parallelisable, computational concerns are not pertinent. However, malformed contours may result in erroneous slot-object pairings which provide incorrect signals to the auxiliary models. This failure mode is particularly notable early on in training when attention maps may be quite diffuse; however, qualitative evaluations of the labelled latents (such as the latent manifolds shown in Figure 4.6) provide a clear way of identifying when this occurs.

4.2 Disentanglement in Slot-Attention

We present results comparing the reconstruction performance and learned representations of the proposed SA+ST architecture presented in section 4.1, against the SA+SB model from Locatello, Weissenborn, et al. 2020. All experiments were performed on the modified SpriteWorld dataset with five-seeds for each model, trained for 250,000 iterations.

4.2.1 Model Performance

Though the focus of our efforts is to create a model which learns structured representations in conjunction with the SA module, we first confirm that the replacement of the SB decoder with a ST does not hamper performance. The qualitative reconstructions shown in Figure 4.3 as well as the quantitative results in Table 4.1 confirm this to be the case.

Note that both models struggle to effectively capture colour information when colours are allowed to vary continuously, which is a known issue with slot attention (Löwe et al. 2020). SA's difficulty with colour does not only affect decoding quality, but also its ability to represent nearby objects. In particular, when objects of similar colour are close to each other SA may struggle to effectively distinguish between these, even if they belong to distinct shape classes. This can be observed for examples 5 (adjoining triangle and circle) and 7 (adjoining squares) where the corresponding attention masks are degraded. Notably, the spatial proximity of the two smaller squares in example 7 is similar to those of the larger squares, but the masks and reconstructions of the smaller squares are superior, which we attribute primarily to the more distinct colours.

An interesting, and undesirable, observation can also be made from Figure 4.4 of the glimpses generated by the SA+ST model. Whilst we do not expect a strong bias towards any canonical pose for the glimpses, we may have expected a consistent pose to be learned. For example, triangle glimpses may always have been rendered at some arbitrary but consistent angle, such that the transformation, $\vec{\theta}$, would capture all of the relative pose information. As this does not appear to be the case, we cannot reasonably expect meaningful information about object orientation to be encoded in $\vec{\theta}$, which must be kept in mind, or remedied, when designing downstream components relying solely on $\vec{\theta}$; for example, imposing dynamics models for rotation as well as linear motion. As neither model was found to capture

Model	MSE $\left(10^{-3}\right)$	ARI
SA+SB	8.5 ± 0.9	0.82 ± 0.02
SA+ST	9.2 ± 1.2	0.80 ± 0.02

Table 4.1: Quantitative performance measures (means $\pm$ st.dev.) of the two models on the SpriteWorld dataset. We observe no statistically significant ($< 3\sigma$) degradation in model performance, either in reconstruction or object-wise segmentation (via mask ARIs).

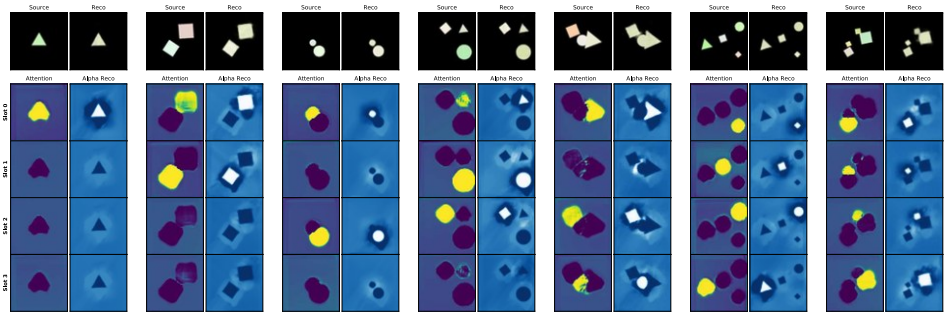


Figure 4.3: Slot attention with Spatial Broadcast decoder. The top row shows the input image (source) and reconstruction (reco). Each of the seven column-groups is a different held-out test scene. For each of the model's four slots (rows), the two sub-columns display the SA attention map and the SB decoder's alpha-weighted reconstruction respectively. We see that the attention masks (what a SA slot captures) match well with what is decoded (what the ST outputs for that slot).

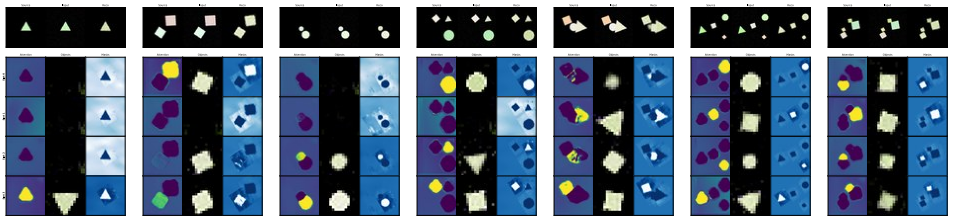


Figure 4.4: Slot attention with Spatial Transformer. The top row shows the input image before (source) and after (input) the addition of noise, alongside the reconstruction (reco). The following rows show, for each slot, the attention maps as well as the decoder glimpses ("objects", at the decoder's native glimpse resolution, prior to transformation) and the alpha masks after the ST has upsampled and translated them onto the scene. As in Figure 4.3, each column-group corresponds to a different held-out test scene and rows to the four slots. Note that there is no canonical angle of representation for the objects even with ST — as the glimpses of triangles and squares occur at different orientations.

angles in a disentangled fashion, and may not be expected to do so in a single latent, even in principle (Zaidi et al. 2021), we omit angles from the following results.

4.2.2 Learned Latent Representations

In order to discern the extent to which the SA+ST model learns suitably disentangled and structured latent representations, a mixture of quantitative and qualitative techniques were applied.

Table 4.2 shows the fit goodness measures, on testing data, for auxiliary models trained to predict ground truth factor values from latent vectors. For each latent, 20,000 labelled embeddings were collected, of which 80% were used for training the auxiliary models, and 20% for testing. The extent to which specific elements of the latent vector contributed towards auxiliary model decisions ("feature importances") is shown in Figure 4.5. These were used to calculate the DCI Disentanglement and Completeness scores which are shown in Table 4.3.

Note that in the case of SA+ST, $\vec{z}'$ refers to the concatenation of the object representation vector, $\vec{O}$, and the transformation vector, $\vec{\theta}$ (as shown in Fig. 4.5). It would have been possible to achieve even better scores by directly utilising knowledge of the structure of $\vec{z}'$ to train auxiliary models only on relevant parts of the latent vector. However, as the models automatically exploited this structure, and the resulting scores were satisfactory, this was deemed unnecessary.

The feature importances for the auxiliary models[7], on the relevant latent vectors, are shown (normalised by row) in Figure 4.5. The structure we expect in $\vec{z}'$ is largely present, with shape and pose encoded in $\vec{O}$ and $\vec{\theta}$ respectively. In particular, the x,y position information corresponds to the final column of $\vec{\theta}$ given the skew-free transformation form discussed in subsection 4.1.1 (elements 3 and 6, for the flattened matrix).

Table 4.2: Accuracy for classifiers (shape) and regressors (scale, X, Y), on a held out subset of labelled latents collected on the test set (using procedure outlined in subsection 4.1.3). We see that intrinsic object properties (shape and scale) are substantially more easily decoded from slots in our SA+ST model.

Model	Shape	Scale	X	Y	Angle
SA+SB: Slots	0.67 ± 0.13	0.80 ± 0.02	0.948 ± 0.005	0.949 ± 0.006	—
SA+ST: Slots	0.47 ± 0.01	0.82 ± 0.02	0.950 ± 0.008	0.943 ± 0.016	—
SA+ST: $\vec{z}'$	$\mathbf{0.87 \pm 0.01}$	$\mathbf{0.86 \pm 0.01}$	0.945 ± 0.007	0.940 ± 0.015	—

[7] For figures concerning SA+ST/SB, models were handpicked on the basis of latent manifold structure (discussed shortly).

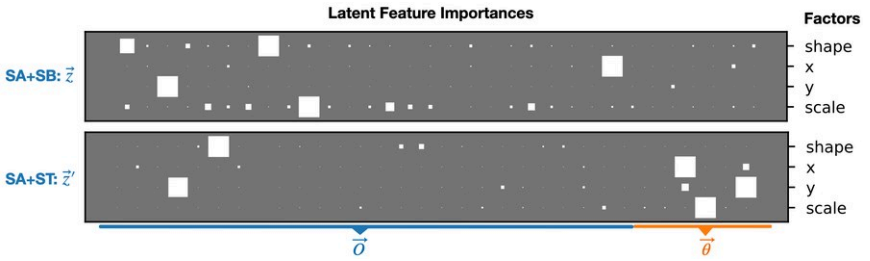


Figure 4.5: Hinton diagrams showing the feature importances for auxiliary models trained to predict ground truth factors of variation from latent values for: **Top:** Slot Attention output slots. **Bottom:** Spatial Transformer Latents. Rows correspond to ground-truth factors (shape, x, y, scale) and columns to individual latent elements; each square's area encodes the row-normalised feature importance assigned by the auxiliary model for that factor. For SA+ST, the latent is the concatenation $\vec{z}' = [\vec{O}, \vec{\theta}]$, with the $\vec{O}$ block (blue) on the left and the $\vec{\theta}$ block (orange) on the right of the divider.

Note that the y position is also strongly encoded in $\vec{O}$ for reasons that are not well understood; one may hypothesise that the glimpse decoder needs to be able to render (or mask) clipped objects at the edges of the scene, but this does not explain why the x offset information is not encoded in $\vec{O}$ (as the dataset is symmetric in x and y object trajectories).

The DCI Disentanglement and Completeness scores, shown in Table 4.3, were calculated from these feature importances. By these metrics, the structured latents from the SA+ST model are significantly more disentangled and complete.

This is due to the significant difference in the degree of disentanglement of scale and shape encodings, rather than positions. This is not surprising as the spatial broadcast decoder also strongly encourages disentanglement of object position information, as discussed by Watters, Matthey, Burgess, et al. 2019. Following their methodology, we also investigate the latent manifolds corresponding to x,y position, which are shown in Figure 4.6. Latents taken to correspond to x,y were those with the greatest feature importance for the auxiliary regressor[8] on $\vec{z}$ or $\vec{z}'$.

[8] This method of determining the most salient latent can fail as in some circumstances the y feature importance contribution is dominated by $\vec{O}$, as discussed above. In these cases, the manifolds shape is dramatically warped as the y information captured in $\vec{O}$ does not map as explicitly as that in $\vec{\theta}$.

There are two primary findings here. First, the linearity of the mapping learned by SA+SB varied substantially across our 5 runs, with only the best run shown. Second, the mapping learned by SA+ST is more homogenous across the domain of x,y values, but also less precise. We attribute this to the variation in object positions within the glimpses, which are compensated for by corresponding offsets in the transformation vector $\vec{\theta}$.

However, further investigation is needed as the object representation vectors occupy a high-dimensional space, and t-SNE is a non-linear clustering method, such that applying a simple distance measure as suggested may prove insufficient.

The final desired property of object-centric representations which we have not discussed is that of consistency over time. To this end, Appendix subsection B.2.2 shows that, at least in simple settings, re-initialising slots to their previous (temporal) state in a dynamical setting provides a simple and effective way of tackling the consistency issue.

4.3 Limitations and Related Directions

While SA+ST successfully disentangles pose from object representation, as seen in Table 4.3 and Figure 4.7, several architectural limitations suggest directions for future work.

Exclusivity SA's attention mechanism only loosely enforces exclusivity, where a single slots exclusively captures a single object. Other methods suffer less from this issue, for example by using contrastive objectives Löwe et al. 2020 or performing explicit matching (Kipf et al. 2020). More recent work makes fewer assumptions about observations, and leverages implicit gradients to improve the training of SA (Chang et al. 2023) — a method directly applicable to SA+ST.

Alignment In the dynamical setting, aligning representations is not naively supported by SA. While our recurrent extension shows promise, its robustness needs to be compared to methods that learn to account for dynamics Löwe et al. (2020); Z. Wu et al. (2023), a particularly applicable approach is that of (Creswell, Nikiforou, et al. 2020) which need not make assumptions about the details of the encoder and should thus be compatible with SA+ST.

Table 4.3: DCI Disentanglement measures for the SA+SB compared with SA+ST.

Latent	Completeness	Disentanglement
SA+SB: Slots	0.49 ± 0.04	0.72 ± 0.04
SA+ST: Slots	0.45 ± 0.01	0.67 ± 0.03
SA+ST: $\vec{z}'$	$\mathbf{0.62 \pm 0.06}$	$\mathbf{0.84 \pm 0.05}$

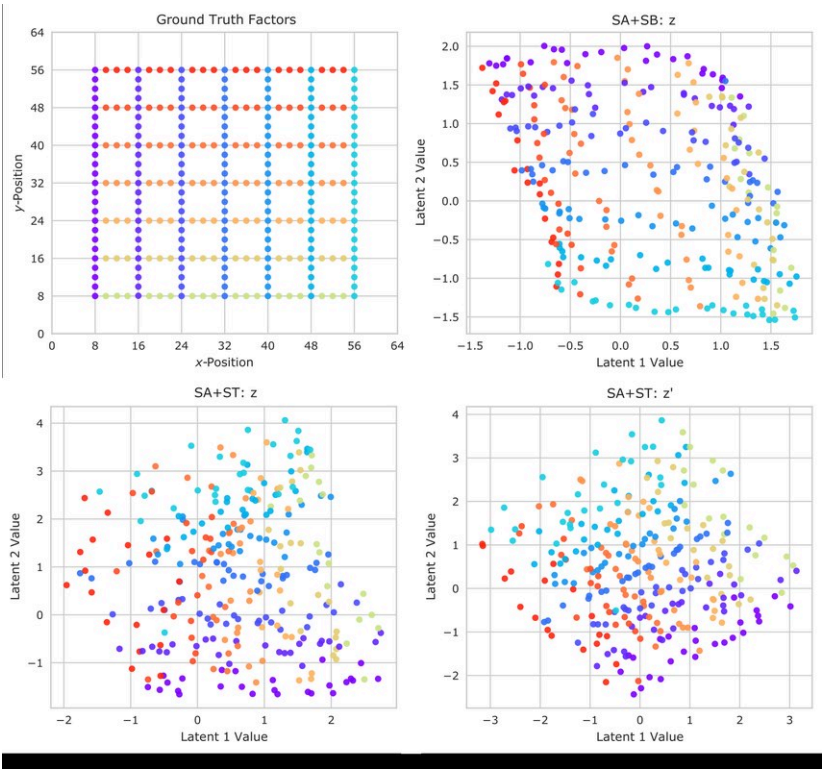


Figure 4.6: Latent manifolds obtained by i) taking the two elements of a latent which had the highest associated feature importance on regressors for ground truth position, and ii) finding the values of those latent elements when encoding objects whose positions corresponded to the x, y grid shown in the upper left. From the learned mapping we can qualitatively gauge the explicitness of the representation for position — In this case, both SB and ST do well, though ST more consistently learned a highly explicit mapping across our runs.

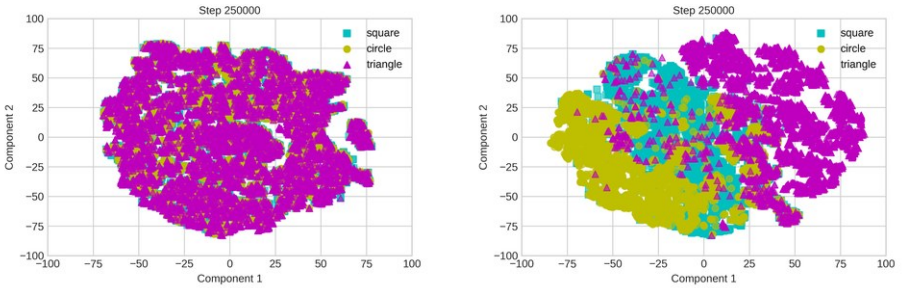


Figure 4.7: We perform dimensionality reduction using t-SNE on the latents learned by the both models. **Left:** SA+SB output slots, $\vec{z}$. **Right:** SA+ST object representation latents, $\vec{O}$. We see that the object representations acquired by the SA+ST model cluster more readily, with object separation by class. Marker shape and colour encode the ground-truth class of the object assigned to each latent (square: cyan, circle: yellow, triangle: magenta).

Background Representation. Our model inherits SAs limitations in distinguishing objects from backgrounds. As shown in Figure 4.4, slots sometimes produce overlapping attention masks that complicate scene decomposition. Recent work exploring privileged background slots (Qi et al. 2024) or explicit mask regularisation similar to MONet's KL terms (Burgess, Matthey, et al. 2019) suggests promising directions for improvement.

Continuous Property Encoding. The diffuse encoding of continuous properties like colour highlights a deeper challenge: some visual attributes may inherently resist localised representation. This is particularly problematic for RL applications where colour often signals reward structure. While discrete colour spaces (as in Tetrominoes) sidestep this issue, real-world applications demand better solutions for continuous 3D colour spaces.

4.4 Conclusion

We have shown that the SA+ST model learns a structured latent representation, effectively disentangling object pose information from object representation whilst preserving the architectural simplicity of SA's self-attention backbone, without notably sacrificing reconstruction or segmentation performance. While this demonstrates the feasibility of learning

structured representations through architectural design, our evaluation has been confined to a 2D setting without backgrounds, and challenges like the diffuse encoding of continuous properties hint at inherent limits in how much structure we can impose — limits that richer scenes with backgrounds and 3D variation would only sharpen.

Such difficulties aside, our improved representations bring us back to the central issue: even when we successfully engineer more interpretable features, do they actually improve reasoning performance? The true test of SA+ST's disentangled representations lies not only in their interpretability, but also in whether downstream architectures can exploit this structure for better compositional generalisation. In the next Chapter, we will systematically investigate this, examining how different architectures may leverage structured representations for reasoning task, and to what effect.

Chapter 5

Reasoning over Structured Representations

The previous chapter showed that structured, disentangled representations can be learned through architectural constraints; we now stress-test whether such representations actually improve compositional reasoning, setting aside pose factorisation to isolate the reasoning question. This chapter investigates whether object-centric representations benefit relational reasoning architectures, and what role sparsity plays in encouraging simpler, more interpretable rules. We find that object-centric representations alone do not guarantee improved reasoning—indeed, imperfect segmentation can be actively detrimental—but that feature-level sparsity on relational modules yields simpler relations that are more faithfully approximated by symbolic models.[1]

Whether structured representations actually benefit compositional reasoning depends on the architecture operating over them and the constraints under which it learns. We situate this question within the relevant literature before presenting our experimental framework.

5.1 Background

Compositional reasoning—the ability to understand complex scenes and concepts by combining simpler parts—requires neural architectures that are capable, and ideally predisposed, toward modelling relationships between entities. As discussed in previous chapters, disentangled representations provide a foundation for such reasoning by separating independent factors of variation. However, disentanglement alone is insufficient; downstream models should leverage inductive biases for combining these factors through relational computation. Whilst there are many architectures, not least the transformer, *capable* of capturing relations, we shall turn our attention to those which are explicitly designed to do so.

5.1.1 Relational Neural Networks

Relational reasoning architectures share a common principle: they explicitly compute interactions between discrete entities or feature groups, implementing what P. W. Battaglia et al. 2018 term a "relational inductive bias." This bias manifests as structured computation over entity pairs or higher-order tuples, often with permutation-invariant aggregation to ensure consistent outputs regardless of input ordering.

Thus far we have supposed that such architectures naturally complement disentangled representations, as clearly separated factors facilitate meaningful relational computation and interpretability at multiple levels.

[1] This work was published as Spies, A. Russo, et al. 2022.

The RelationNet (Santoro et al. 2017) implements perhaps the simplest direct instantiation of this principle. It ensures that relational reasoning is performed by applying a shared MLP, g_θ, across all possible permutations of feature pairs and summing the outputs, ensuring that the model is invariant to the order of the features.

$$\mathrm{RN}(\mathbf{O}) = f_\phi\left(\sum_{i,j} g_\theta\left(\vec{o}_i, \vec{o}_j\right)\right), \tag{5.1}$$

where $\mathbf{O} = \{\vec{o}_1, \vec{o}_2, ..., \vec{o}_n\}$ represents a set of object representations, $g_\theta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^k$ is a learned relation function (implemented as a 4-layer MLP), and $f_\phi : \mathbb{R}^k \to \mathbb{R}^m$ processes the aggregated relations to produce the final output. This architecture's $O(n^2)$ complexity in the number of objects reflects its exhaustive consideration of all pairwise interactions, trading computational efficiency for representational completeness.

The PrediNet (Shanahan et al. 2020) addresses the quadratic scaling through selective attention, learning to focus on relevant entity pairs. Unlike RelationNet's exhaustive approach, PrediNet employs H attention heads, each selecting two weighted combinations of input features to compare:

$$\mathbf{D}^h = \mathbf{E}_1^h \mathbf{W}_S - \mathbf{E}_2^h \mathbf{W}_S \tag{5.2}$$

where $\mathbf{E}_1^h, \mathbf{E}_2^h \in \mathbb{R}^{b \times d}$ are attention-weighted entity representations for head h, computed as $\mathbf{E}_1^h = \mathrm{softmax}(\mathbf{Q}_1^h \mathbf{K}^T / \sqrt{d})\mathbf{V}$ (and similarly for $\mathbf{E}_2^h$), with learned query projections $\mathbf{Q}_1^h, \mathbf{Q}_2^h$. The shared projection $\mathbf{W}_S \in \mathbb{R}^{d \times r}$ maps entities to a common relational space where subtraction computes directed relations.

Each element of the resulting difference vector $\mathbf{D}^h \in \mathbb{R}^{b \times r}$ represents a specific binary relation between the selected entity groups. In our experiments we allow each head to maintain its own relational projection $\mathbf{W}_S^h \in \mathbb{R}^{d \times j}$, allowing specialisation across different relation types, and making a fairer comparison with our slot-based variant.

We introduce a **Slot PrediNet** variant for slot-based representations, where attention queries are generated per slot rather than across all features. This modification aligns with the discrete object-centric structure of slot representations, ensuring that relational computation respects object boundaries. Given slots $\mathbf{S} \in \mathbb{R}^{B \times k \times d}$, we first project to a shared key space $\mathbf{K} = \mathbf{S}\mathbf{W}_K$. For each attention head h, queries are computed by

aggregating across all slots: $\mathbf{Q}_1^h = \sum_{i=1}^{k} \mathbf{S}_{:,i,:} \mathbf{W}_{Q_1}^h$ (and similarly for $\mathbf{Q}_2^h$). The attention weights $\mathbf{A}_1^h = \text{softmax}(\mathbf{Q}_1^h \mathbf{K}^T)$ select weighted combinations of slots $\mathbf{E}_1^h = \mathbf{A}_1^h \mathbf{S}$, with the final relational representation computed as $\mathbf{D}^h = \mathbf{E}_1^h \mathbf{W}_S^h - \mathbf{E}_2^h \mathbf{W}_S^h$. Unlike the original PrediNet, each head maintains its own relational projection $\mathbf{W}_S^h \in \mathbb{R}^{d \times j}$, allowing specialisation across different relation types while preserving the object-centric inductive bias of slot representations.

Related Architectures: The relational reasoning paradigm extends beyond these specific implementations. Graph Neural Networks (GNNs) (P. W. Battaglia et al. 2018) generalise relational computation to arbitrary graph structures through iterative message passing. Transformers (Vaswani et al. 2017), while not explicitly designed for relational reasoning, implement it through self-attention's all-to-all comparison mechanism—essentially a learnable, context-dependent RelationNet. Numerous other works (Andreas et al. 2017; A. Goyal et al. 2021; Hudson et al. 2018) exist which take a more structured approach, composing specialised modules for different relational operations, though these are often designed with specific tasks, such as question-answering, in mind tasks. Akin to the plethora of approaches toward OCL, these diverse architectures again demonstrate that simple inductive biases can be implemented with varying levels of flexibility-structure trade-offs.

5.2 Datasets

The research question now differs from that of chapter 4: rather than asking whether structure can be *learned*, we ask whether downstream architectures can *exploit* it. This reframing drives a different dataset rubric. We require benchmarks that (i) expose explicit relational labels so that compositional generalisation can be measured, (ii) follow evaluation protocols already adopted by the relational architectures under test, so that our results remain directly comparable with prior work, and (iii) isolate relational from non-relational computation within the same scene. The *Relations Game* (Shanahan et al. 2020) is canonical for PrediNet-style evaluation and explicitly probes rule composition through its two-stage curriculum.

Sort-of-CLEVR (Santoro et al. 2017; Tripathi 2020) is paired with RelationNet's original evaluation protocol and contrasts relational with non-relational queries on identical scenes. The Spriteworld benchmark used

in chapter 4 carries no such task labels, and re-using it here would require synthetic task construction that weakens comparability with the published baselines.

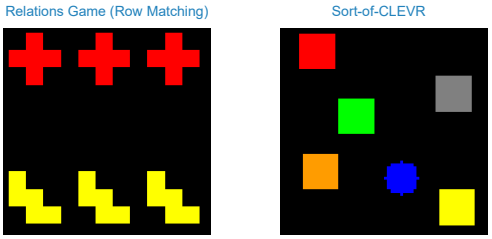


Figure 5.1: Examples of the datasets. On the left, a "Row Matching" task from the Relations Game, in which the patterns in both rows (AAA) match, and the label is True. On the right, a Sort-of-CLEVR configuration, which has a set of 20 associated questions. These include non-relational "What colour is the circle"), and relational ("What is the colour of the square closest to the circle") questions.

The Relations Game (Shanahan et al. 2020), provides a simple set of relational reasoning problems in a low-dimensional image setting. As shown in Figure 5.1, these problems test various aspects of pattern recognition and relational understanding through five distinct tasks:

- Between Columns/Rows (BC/BR): Does a distinct object appear between two identical (to each other) objects?
- Occurs: Does the object in the top row occur in the bottom row?
- Same: Are both objects in the scene identical?
- Row Matching / Column Matching (RM / CM): Is the *pattern* of distinct objects within a row (resp. column) identical to that of the other row (resp. column)?
- Row Patterns / Column Patterns (RP / CP): (Multi-task) Task specified by an index that determines which pattern should be matched (e.g. "is AAB the pattern present in the row?")

Following Shanahan et al. 2020, we employ a two-stage curriculum learning setup:

1. **Stage 1:** Train the entire architecture (Encoder + Rule Learner + Task Network) on a simple task, such as Between Columns.
2. **Stage 2:** Train a new Task Network with frozen Encoder and Rule Learner on more complex tasks, such as the multi-pattern tasks.

For the PrediNet variant, the CNN encoder is jointly trained in the first stage. For the Slot PrediNet, the encoder is pre-trained using slot attention (see section C.4). Unlike Shanahan et al. 2020, we maintain consistent shape representations (e.g., pentominos) throughout both stages, as we are not interested in assessing encoder generalisation.

Sort-of-CLEVR (Tripathi 2020), is a simplified version of the CLEVR dataset (J. Johnson et al. 2017) designed for evaluating relational reasoning. The dataset consists of 75×75 pixel images containing six 2D objects (circles and squares) with various colours. Each image is associated with 20 questions divided into two categories:

- Non-relational questions: Query properties of individual objects (e.g., "What colour is the circle?")
- Relational questions: Query relationships between objects (e.g., "What is the colour of the square closest to the circle?")

Our implementation builds upon the original Sort-of-CLEVR dataset with modifications to the question encoding format and removal of the white background which was found to improve segmentation with Slot-Attention (SA). For experiments using slot-based representations, we pre-train the SA module as detailed in section C.4.

These datasets provide complementary evaluation scenarios: the Relations Game focuses on explicit pattern matching and rule learning, while Sort-of-CLEVR tests relational reasoning at multiple arities, including direct object property queries.

5.3 Methodology

There are two key aspects to our investigation. Firstly, we vary the structure of models' encoder outputs, using either slot-based representations or CNN feature maps. Secondly, we vary the kinds of sparsity priors present in a subset of the models; both at the level of feature-vector selection, and relations' feature dependencies.

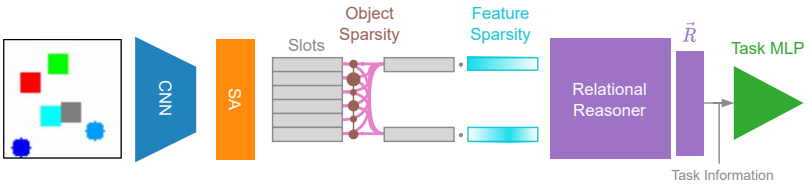


Figure 5.2: Schematic overview of a sparse relational reasoning architecture which leverages Slot-Attention (SA). We impose two kinds of sparsity: i) Object sparsity is applied on the attention mechanism which selects objects (encoded in slots), and ii) Feature Sparsity is applied on the relations through regularisation, encouraging sparse dependence on object-features for each learned relation.

5.3.1 Models

An overview of the relational-slot architecture is shown in Figure 5.2. For learning object-centric representations in the form of slots we use Slot-Attention (SA) (Locatello, Weissenborn, et al. 2020). The outputs of SA or a simple CNN are then fed into one of two explicitly relational architectures: i) the RelationNet from Santoro et al. 2017, or ii) the PrediNet from Shanahan et al. 2020. Task-relevant information (such as the question type) is then appended to the outputs of the "relational reasoners" before a small task-dependent MLP is applied. These architectures are chosen due to the relative simplicity with which they realise explicitly relational learning.

Slot Attention: We use Slot-Attention (SA) as our encoder for generating slot-based representations, as discussed in section 3.2.3. SA maps input features from a CNN (dimension $N \times D$) to K slots (vectors $\in \mathbb{R}^D$). For pretraining, we employ the same autoencoding setup as Locatello, Weissenborn, et al. 2020, using the baseline Spatial Broadcast Decoder (SB) decoder with L2 reconstruction loss. Given that the tetrominoes in the relations game are arranged on a regular grid, we created a pretraining dataset where tetromino positions vary continuously and sizes are sampled from three discrete values (see section C.4). This results in SA learning smooth, contiguous attention masks, rather than crudely dividing up the grid; this allows us to perform the slot-object matching procedure required for disentanglement analysis.

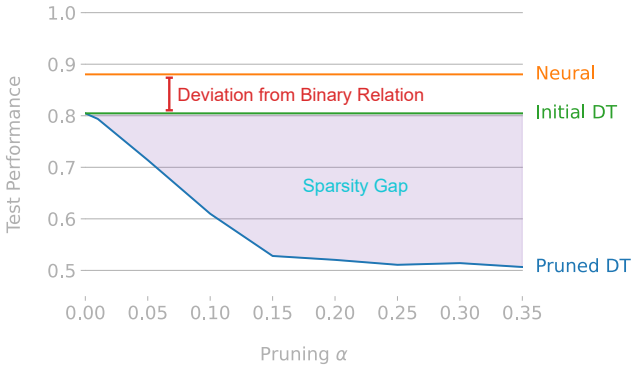


Figure 5.3: Visualisation of sparsity measures on rule vectors, relying upon the sweep of Decision Tree (DT) pruning complexities. In particular, the "Deviation from Binary Relation", ΔBR, captures the drop in accuracy on a reasoning task when a model's relations (mappings to the elements of the $\vec{R}$ vector) are replaced with decision trees which map inputs to the same outputs in an explicitly pair-wise, binary, fashion. The sparsity gap then measures the effect of pruning these decision trees to rely on as few dimensions of their input vector pair as possible.

5.3.2 Sparsity

We experiment with two kinds of sparsity on the PrediNet and Slot PrediNet. These are *object sparsity* (relations specialise to few objects) and *feature sparsity* (relations use few features per object).

Inducing Sparsity

Object Sparsity is applied at the level of the attention mechanism of the PrediNet, which is responsible for selecting the sets of features (or slots) over which each head applies relations. To encourage sparsity we replace the softmax attention with `sparsemax` (Martins et al. 2016) or `gumbel-softmax` (Jang et al. 2017).

Feature Sparsity is applied at the level of the learned relation weights, which are applied to the (transformed) input features of each head. Pushing these weights to be sparse forces each head's relations to rely on as few elements in the input feature vectors (or slots) as possible. To this end, we apply L1 and Entropy regularisation.

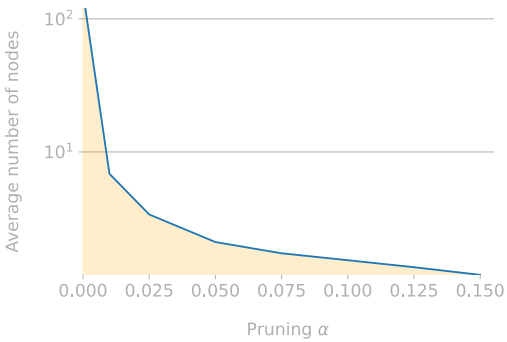


Figure 5.4: The number of nodes in the auxiliary trees decreases as pruning is increased (which trades off performance). The average number of nodes in the range $\alpha \in [0, 0.15]$ is used as a measure of relation sparsity.

Measuring Sparsity

For each dimension of the output rule vector we fit a simple auxiliary model which learns to map **two** input feature vectors to a scalar. This procedure assumes that (i.e. confirms whether) the relational component can be well-approximated as a set of binary relations, $\vec{R}^{(i)} = f(\vec{o}_j, \vec{o}_k)$, where the set of input feature-vectors is denoted as $\mathbf{O} = [\vec{o}_0, \dots]$. We quantify the extent to which a relational module can be modelled in this fashion as the deviation from binary relation (ΔBR), which is shown in Figure 5.3. The ΔBR is calculated as the drop in performance of the end-to-end model when its relational reasoning module is replaced by a set of auxiliary models.

In particular, we chose to use Decision Trees owing to their simplicity and ease of pruning, which allow robustness to sparsity to be easily assessed, as shown in Figure 5.4. Additionally, the dependence of a given decision tree on its inputs ("feature importances") can be used to quantify feature sparsity on the learned relations, by computing the entropy of these feature importances.

It is worth being precise about what "binary" does and does not refer to in our study. The *primitive* relation computed by both the RelationNet and the PrediNet is pairwise by construction: g_θ operates on two entities at a time, and each PrediNet head compares two attention-weighted entity groups. This is a constraint on the module's vocabulary, not on the tasks we evaluate — the Relations Game "Between Columns" task requires reasoning over at least three objects, "Row Matching" over all six, and Sort-of-CLEVR relational questions such as "how many objects share a shape with X?"

aggregate over the entire scene. Nor is it a constraint on what the composite architecture can express: because the rule vector $\vec{R}$ concatenates many such pairwise primitives in parallel (across heads for PrediNet, summed across pairs for RelationNet) and a downstream MLP consumes the whole vector, any n-ary property that decomposes as a conjunction or aggregation of pairwise facts remains representable.

Our evaluation metric ΔBR (section 5.3.2) inherits this framing: it tests *per dimension* of $\vec{R}$ whether the corresponding primitive can be recovered by a two-argument auxiliary model, not whether the task as a whole reduces to a binary relation. What lies outside this scope is *multi-step* relational reasoning, in which the argument of one relation is itself the output of another — a single forward pass of a one-layer RelationNet or PrediNet cannot feed a relational output back in as an input.

5.4 Experiments

Table 5.1: Test Accuracies (%), across four seeds, for all model variants on both datasets (without sparsity priors). Relations Game task abbreviations (BC, CP, RM, RP) are defined in section 5.2. In each A→B column, A is the Stage 1 pre-training task (on which the full architecture is trained jointly) and B is the Stage 2 transfer task (on which only the task network is re-trained, with the encoder and rule learner frozen).

Model		Relations Game			Sort-of-CLEVR	
Encoder	Relational	BC→CP	BC→RM	RM→RP	Non-rel.	Rel.
SA	MHA + MLP	56 ± 9	50 ± 2	50 ± 1	85 ± 3	77 ± 3
CNN	PrediNet	87 ± 3	66 ± 0	74 ± 8	95 ± 1	79 ± 0
GT	Slot PrediNet	86 ± 2	58 ± 2	85 ± 2	85 ± 2	81 ± 1
SA	Slot PrediNet	73 ± 2	53 ± 1	59 ± 3	95 ± 2	77 ± 2
CNN	RelationNet	53 ± 1	52 ± 1	52 ± 1	100 ± 0	80 ± 1
GT	RelationNet	75 ± 12	63 ± 7	55 ± 8	100 ± 0	100 ± 0
SA	RelationNet	70 ± 1	49 ± 1	49 ± 1	100 ± 0	90 ± 1

5.4.1 Do object-centric representations improve performance?

We first investigate whether object-centric representations are beneficial when used in conjunction with relational reasoning architectures, as well as a simple multi-head attention baseline (MHA + MLP).

To this end, we apply the PrediNet and RelationNet with i) learned CNN encoders, ii) ground-truth (GT) "slots", and iii) pre-trained SA encoders.

The results of these experiments are summarised in Table 5.1 and further discussion is provided in Section C.3.1. Focussing on the questions outlined above, we find that object-centric representations are not necessarily beneficial. Indeed, when used in conjunction with the PrediNet on the Relations Game, SA is detrimental. There are two likely reasons for this. Firstly, SA often fails to capture all objects present in a scene (see section C.2), which is fatal for tasks in which knowledge of all objects is necessary; this explains why the Slot PrediNet does especially poorly on the RM→RP tasks, in which information about all six objects is required. Secondly, the CNN used with the PrediNet can learn relational representations directly, whereas SA encodes information about objects individually.

For the RelationNet we do observe a significant improvement in performance on relational tasks when using SA instead of a CNN. However, we attribute this primarily to the fact that the RelationNet struggles with tasks requiring simultaneous reasoning over more than two objects, and so is unable to learn useful CNN encodings.

5.4.2 Does sparsity improve performance?

Assessing the degree of sparsity

We first establish the degree to which the various sparsity biases discussed in section 5.3.2 lead to sparser relations.

In Fig. 5.5 we show the average feature dependence entropy for auxiliary models (column three), as well as the average tree complexity across all levels of pruning (column four). For clarity, we show results only for the PrediNet with CNN and SA inputs, and only on the BC→CP Relations Game tasks. The complete suite of results is provided in Section C.3, along with further discussion in Section C.3.1.

The key finding here is that regularising the relations to rely on few features does indeed lead to simpler relations; in that, they may be faithfully modelled with Decision Trees consisting of fewer nodes, and relying

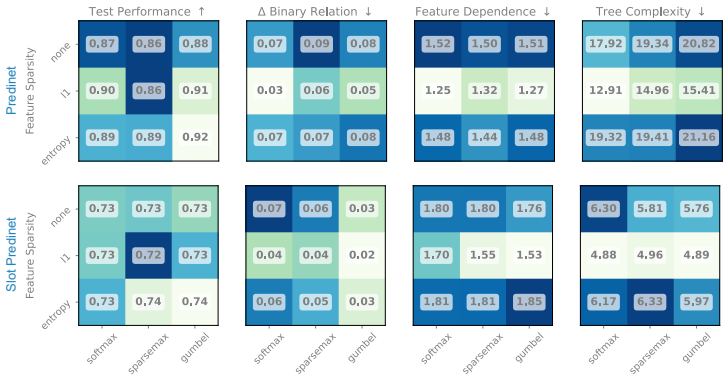


Figure 5.5: Sparsity and performance metrics for the PrediNet and Slot PrediNet on the Relations Game BC→CP task. Columns show, from left to right: (i) test accuracy with the neural relational module, (ii) the ΔBR when this module is replaced by Decision Trees, (iii) the entropy of feature importances (lower = sparser feature dependence) and (iv) mean tree complexity (node count) across pruning levels. All values correspond to averages on the generalisation test set, taken over four seeds. The Feature Dependence and Tree Complexity metrics are averaged across all Decision Trees fitted to a given model's relational reasoning component.

upon fewer of the input features. To this end, L1 regularisation appears considerably more effective than entropy regularisation.

This effect also holds for the Slot PrediNet, and it appears that the auxiliary models are much simpler when slots are used. Whilst we might expect this result given that the slots' features are disentangled (see Table 4.3), most of the effect is likely due to the task performance difference between the models; that is to say, as the Slot PrediNet performs worse, its relations may be modelled more simply.

Assessing the impact of sparsity

Having confirmed that our sparsity priors successfully induce the intended structural biases, we now examine whether these biases translate into improved performance and interpretability. To this end, we are interested in the first two columns of Fig. 5.5 which show the performance on the test set for models whose relational modules have been i) left unchanged (i.e. remain neural) and ii) replaced by a collection of Decision Trees (shown as the difference from column 1).

Again there are two primary findings. Firstly, increasing sparsity improves the performance of the models on the RelationNet generalisation

tasks, however, this effect is more pronounced for the PrediNet than the Slot PrediNet. This is not surprising, as the PrediNet can easily mix information from all features of all objects, and thus benefits most from being biased towards increased sparsity.

Secondly, looking at the ΔBR we see that L1 regularisation leads to relations more closely approximating independent binary rules. However, the different forms of object sparsity have little systematic impact. We primarily attribute this to the similarity of soft and hard-attention values by the end of training. It is worth noting that hard attention appears to be beneficial when used in conjunction with the PrediNet; likely as this prevents the CNN from learning relational features which overfit to the pretraining task.

5.5 Conclusions

Part I of this thesis began with the premise that carefully engineered inductive biases—particularly structured, object-centric representations—would enable more interpretable and compositional reasoning in neural networks. Chapter 4 demonstrated that we could successfully engineer such structure through the SA+ST architecture, achieving improved disentanglement of object properties. This chapter tested whether structured representations would naturally lead to better relational reasoning when paired with architectures explicitly designed for compositional computation.

Our findings tell a more nuanced story. While we confirmed that **increasing feature sparsity improves both performance and interpretability** of learned relations, we also discovered that **relational reasoning modules are highly sensitive** to the quality of their input representations. Most strikingly, the object-centric representations from SA, despite their theoretical advantages, often underperformed simple CNN features. This sensitivity manifests as a trade-off: architectures with strong relational inductive biases sacrifice the flexibility to adapt when their structural assumptions are violated, such as when SA fails to capture all objects in a scene.

Our decision tree analysis further illustrates this tension between structure and flexibility. Even with aggressive sparsity regularisation, the learned relations remain complex, requiring trees with many nodes to faithfully approximate their behaviour. This suggests that while we can engineer structure into our representations and encourage sparsity in our relations, the resulting models still lack the transparent compositionality we initially

sought. The RelationNet's struggles with tasks requiring reasoning over more than two objects, and the PrediNet's heavy dependence on its encoder learning task-specific features, both point to brittleness in these carefully crafted architectures.

These limitations prompt us to consider an alternative paradigm. While we have been engineering specific structural biases into specialised architectures, the field has simultaneously witnessed the emergence of foundation models that achieve remarkable capabilities predominantly through scale. These models appear to sidestep the brittleness of engineered structure by learning flexible representations from vast amounts of data. Yet, as we shall see in the following interlude, they are not without their own failings —particularly in visual reasoning tasks that require extracting and binding information from images. Understanding these complementary failure modes will motivate Part II's investigation into the representations that emerge naturally through learning, rather than those we explicitly engineer.

Interlude

Chapter 6

Assessing Reasoning in Multimodal Foundation Models

Part I explored engineering structure into neural networks through object-centric representations and relational reasoning modules, with mixed results: architectural biases improved disentanglement but proved brittle when assumptions were violated. We now ask whether scaling monolithic foundation models can address structured visual reasoning without such explicit priors. Evaluating eighteen multimodal models on controlled reasoning tasks, we find a persistent gap between textual and visual reasoning—models reason competently from textual scene descriptions but cannot reliably extract spatial structure from images, suggesting that scale alone does not resolve the binding challenges identified in Part I. These findings motivate the shift in Part II toward reverse-engineering the structure that does emerge in trained models.[1]

If hand-crafted structure proves brittle, a natural question is whether sufficient scale can compensate. The scaling paradigm (Sutton 2019), which has celebrated numerous successes in recent years, advocates taking general-purpose models (which are usually "simpler" from the perspective of inductive bias) and scaling them up. This scaling, both in terms of data and model size, can require many orders of magnitude more compute but is often justified by the supposed generality of the resulting models. This paradigm of scaling has seen the release of ever larger and more powerful models (Achiam et al. 2023; Brown et al. 2020; Chowdhery et al. 2022; Google et al. 2023; Touvron et al. 2023), accompanied by ever more challenging capability evaluations (Chollet et al. 2025; Phan et al. 2025) that highlight both general and domain-specific abilities.

Based on the Transformer architecture introduced in Vaswani et al. (2017), these purportedly "general-purpose" models are commonly referred to as "Foundation Models" (Bommasani et al. 2022). While such models were originally purely text-based, modern foundation models have evolved to process multiple modalities through various architectural innovations. These include clever tokenisation schemes that treat images as sequences of patches (Dosovitskiy et al. 2021; Radford et al. 2021), cross-modal alignment techniques (Alayrac et al. 2022; J. Li et al. 2023), and unified architectures that process different modalities through shared representations (J. Lu, Clark, et al. 2023). Recent releases including GPT-4V (OpenAI 2023), Gemini (Google et al. 2023), and Claude 3 (Anthropic 2024c) demonstrate impressive capabilities across vision and language tasks, often rivalling or exceeding human performance on established benchmarks (Bubeck et al. 2023).

To assess whether scaling addresses the structured reasoning challenges identified in Part I, we evaluate a variety of foundation models on similar reasoning tasks. Not only do we test whether they can reason in these

[1] This work is unpublished.

domains, but we also isolate the performance of both the visual and textual modalities to identify whether difficulties with binding information between modalities bottleneck performance. This decomposition allows us to disentangle perceptual failures from reasoning failures—a critical distinction for understanding the capabilities of current multimodal systems.

6.1 Methodology

6.1.1 Prompting

Eliciting capabilities from foundation models is a notoriously finicky process, as the extent to which a model can perform a task is highly sensitive to the way in which the task is presented (P. Liu et al. 2023; L. Reynolds et al. 2021). Small variations in formatting, example selection, or even phrasing can lead to substantial performance differences (Yifan Li et al. 2023). To address this sensitivity, we perform various ablations of both the prompt and image formats provided to the models. The prompt configurations determined from these ablations are summarised in Table D.4, with full results in section D.3. We characterise this sensitivity empirically for a subset of models in Figure 6.2.

In particular, we vary aspects of the images such as scale and background colour, as well as varying the prompts (see below). Ablations are carried out with 60 examples per configuration on CLEVR tasks (with images), and 20 examples per task and configuration on Relations game (with images).

Our “prompt engineering" strategy consists of varying three key components:

1. **In-context examples:** Providing examples of a task and its solutions is a common practice for improving the performance and format-adherence of models (Y. Lu et al. 2022; Min et al. 2022). This is typically referred to as In-Context Learning (ICL), as a model "learns" the task provided in its context window—though the learning here is fleeting, as no weights are updated (Q. Dong et al. 2024). In our experiments, we always provide at least one example to inform the model of the input and output format, and then optionally provide an additional three examples.

2. **Reasoning prompts:** Another technique to improve models' performance is to instruct them to explicitly analyse a task, explicate their reasoning, and then reflect upon this before providing their answer.

This Chain-of-Thought (CoT) prompting has been shown to substantially improve performance on reasoning tasks (Kojima et al. 2022; Wei et al. 2022). In recent models, more structured versions of such prompting may form part of their training (Yao et al. 2025). While early work suggested that CoT primarily helped by allowing models to decompose complex problems, recent evidence indicates that the benefits may also arise from effectively increasing the compute expended per answer generated (Pfau et al. 2024; Yangzhen Wu et al. 2025).

3. **Format specifications:** Models show sensitivity to output and input formats across tasks (Sclar et al. 2023). In our ablations, we provide two possible scene-representation formats for every task considered; for example, ASCII or list representations of the 3×3 grids in the Relations Game. This allows us to assess whether reasoning capabilities are robust to superficial changes in presentation.

We provide an example of a prompt overleaf, shown in the ASCII grid output format for clarity of exposition; subsection D.2.2 gives the alternative format. The full set of final prompts for each task appears in section D.1, with a summary of the ablations in section D.2 and the full set of optimal configurations used in the main results in Table D.4. Note that we choose to use the same "best" system prompt for all models based on the results of the ablations, based on average performance across the tasks; as such, the prompts may be suboptimal for some of the models[2].

6.1.2 Experimental Design

To isolate the extent to which multimodality is the cause of any observed failures, we decompose each task into three conditions:

1. **Textual Reasoning:** Models receive ground-truth textual descriptions of scenes and must answer questions based solely on these descriptions. This condition tests pure reasoning capabilities without any visual processing requirements.

[2] Our central claim in this chapter will focus on the fragility of these models, and as such, this choice is not a strong limitation, as it more directly reflects the practical use of such models in deployment, and is more comparable to the assays in Part I — where models were not extensively optimised for each-subtask.

System Prompt Example (Scene Descriptions for the Relations Game)

TASK

```
You are an AI agent tasked with describing a scene from the
relations game, as presented in the PrediNet paper by Shanahan.
You will be provided with an image containing objects arranged
in a 3x3 grid. Your goal is to accurately describe the image
provided.
```

FORMAT

```
An ASCII Grid representation of the form shown below, with the
answer provided between <answer> tags. You should assign a
unique character to each object (factoring in its color and
potential rotation); for example, in the example below ``A"
refers to identical tetriminoes whose color and rotation
matched exactly.

To be clear, the grid is formatted as follows:
a. Create a 3x3 grid using '+', '-', and '|' characters
b. Represent empty cells with a '.'
c. Represent objects with their corresponding labels
   (A, B, C, etc.)
d. Ensure the grid is properly aligned and formatted

And here is an example answer:
<answer>
+---+---+---+
| A | B | A |
+---+---+---+
| . | . | . |
+---+---+---+
| . | . | . |
+---+---+---+
</answer>
```

EXAMPLES

```
<example>
+---+---+---+
| . | . | A |
+---+---+---+
| . | . | B |
+---+---+---+
| . | . | . |
+---+---+---+
</example>
... [we provide three examples in total]
```

GUIDES

```
1. Carefully analyse the image description, noting the position,
   type, and orientation of each object in the 3x3 grid.
2. For each object, identify:
   a. The object's label (e.g., Object A, Object B)
   b. The object's type (e.g., 15 2, 15 5)
   c. The object's rotation (e.g., rotated 180, rotated 90
      clockwise)
   d. The object's position in the grid (e.g., top row, third
      column; middle row, second column)
3. Maintain consistency in terminology and structure
   throughout your description.
4. Produce an output consistent with the format outlined above.
5. Double-check your output for accuracy and completeness
   before submitting.
```

2. **Scene Description:** Models must describe images without answering any reasoning questions. This condition isolates visual perception and scene understanding capabilities.

3. **End-to-End Visual Reasoning:** Models receive images and must answer reasoning questions about them. This condition requires both accurate visual perception and subsequent reasoning, though partial task-conditioned scene understanding would suffice.

This decomposition allows us to identify whether failures stem from perceptual limitations, reasoning deficits, or difficulties in integrating information across modalities. For instance, a model that performs well on textual reasoning but poorly on end-to-end visual reasoning likely struggles with visual perception rather than logical reasoning. Similar ideas have been explored in other works on multimodal failure, though on tasks requiring less complex reasoning (Thrush et al. 2022).

6.1.3 Datasets

Similar to the reasoning tasks in chapter 5, we evaluate models on two complementary tasks that span abstract and naturalistic visual reasoning:

- **Abstract visual reasoning:** We evaluate simple spatial reasoning in a visually abstract domain using the RelationsGame tasks from Shanahan et al. 2020 (introduced in section 5.2). These tasks require understanding spatial relationships between coloured objects in a 3×3 grid, testing compositional visual reasoning without the confounds of natural images.

- **Naturalistic visual reasoning:** To assess more complex reasoning and visual understanding on naturalistic images, we use the CLEVR dataset from J. Johnson et al. 2017. CLEVR scenes are 3D, with multiple objects varying in shape, size, colour, and material. Questions vary by arity—the number of object dependencies that must be resolved—providing a curriculum for assessing reasoning complexity. As shown in Table 6.1, questions range from simple counting (arity 1) to complex relational queries (arity 3) that require tracking multiple object relationships.

Because the vision components of these models descend from paradigms trained predominantly on naturalistic web imagery (Radford et al. 2021),

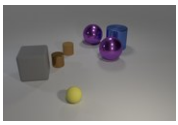
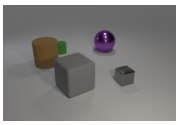
Scene	Question	Answer	Arity
	What number of gray matte blocks are there?	1	1
	Is there a big brown object of the same shape as the green thing?	yes	2
	How many metallic objects are big blue cubes or blue objects?	2	3

Table 6.1: Examples of questions with different arities from the CLEVR dataset. The arity roughly corresponds to the dependencies between objects which must be resolved to answer the question. In the full dataset, questions may go up to an arity of five.

evaluating only on an abstract grid risks conflating reasoning failure with distribution shift; CLEVR's rendered 3D scenes let us probe whether visual understanding and reasoning survive a shift away from the naturalistic regime these models were tuned for. We prefer both over naturalistic alternatives such as GQA (Hudson et al. 2019) or VQAv2 (Y. Goyal et al. 2017) because their procedurally generated scene graphs give us an exact ground truth for every object and answer. This is what makes the three-condition decomposition above attributable to perception versus reasoning rather than to annotator disagreement.

The two datasets also exercise different reasoning primitives — spatial-relational binding in the Relations Game, multi-hop attribute queries in CLEVR — and both were evaluated on the structured architectures of chapter 4 in chapter 5, with CLEVR represented there by its simpler 2D Sort-of-CLEVR variant. Since Sort-of-CLEVR preserves the reasoning structure but not the visual complexity, reusing these benchmarks here licences a direct comparison between engineered structural priors and scale chiefly at the level of reasoning rather than scene understanding.

6.1.4 Models

We evaluate 18 multimodal models released by various organisations between February 2024 and June 2025 (see Figure 6.1). This timeframe captures rapid progress in multimodal capabilities while ensuring API availability for reproducible evaluation.

The models provide a comprehensive view of the current landscape, but as many are closed-source, we cannot make precise claims about their internal architecture, though they are likely to vary. In particular, the visual components of these models may employ diverse architectural strategies, such as:

- Vision Transformer (ViT) tokenisation: Models like GPT-4V and Claude 3 likely use ViT architectures (Dosovitskiy et al. 2021) to directly tokenise image patches into sequences processable by transformer architectures.

- Dual-encoder architectures: Some models may employ CLIP-style dual encoders (Radford et al. 2021) that learn aligned representations for text and images through contrastive learning.

- Cross-modal fusion: Models like Flamingo (Alayrac et al. 2022) and BLIP-2 (J. Li et al. 2023) use specialised cross-attention mechanisms to fuse visual features into language model representations.

While implementations for proprietary models remain undisclosed, the diversity of approaches suggests that architectural choices may significantly impact reasoning capabilities. However, as we will see in our results, scale and training data may compensate for these differences. Newer models, which are likely both larger and more architecturally integrated, tend to show more robust performance across visual reasoning tasks.

We note that while there are many older multimodal models such as CLIP (Radford et al. 2021), ViLBERT (J. Lu, Batra, et al. 2019), and LXMERT (Tan et al. 2019), these were not considered as they either lack the generative capabilities required for our tasks or are not available through public APIs. We do not consider this a substantial limitation, as even the models from the last 14 months show significant performance variation on our tasks, providing sufficient signal to understand the current state of multimodal reasoning capabilities.

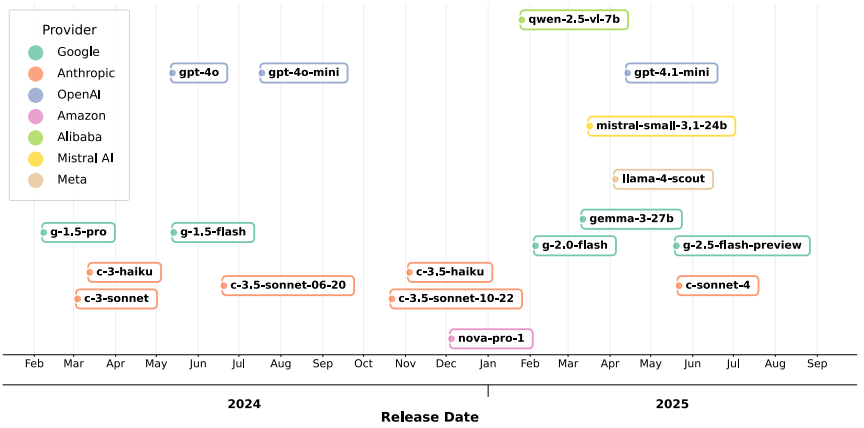


Figure 6.1: We consider a range of models capable of processing both text and images, released between February 2024 and June 2025 and, at the time of writing, still available on public APIs. Whilst there are older notable multimodal models such as CLIP, VilBert etc., these are not tested as they are not available on public APIs. We do not consider this a substantial limitation, as even the time-span considered saw a significant variation in model performance for the tasks we consider. Note the dates here correspond roughly to API release dates, not model announcements. We have truncated "claude" to "c" and "gemini" to "g" for brevity. Full references for each model's announcement post or technical report are provided in section D.5.

6.2 Results

6.2.1 Prompt Sensitivity

Before reporting our main accuracy results, we first characterise how much prompt choice alone affects measured performance. Each (model, task) pair in our ablation sweep was evaluated under multiple prompt configurations, such as varying grid format (colloquial vs. properties-list), position format (absolute coordinates vs. grid cells), scale factor, inclusion of in-context examples, and whether chain-of-thought reasoning was elicited (see section D.3 for details); Figure 6.2 summarises the resulting distribution of accuracies for four illustrative models spanning the capability range[3].

[3] The samples underlying Figure 6.2 come from a dedicated prompt-ablation sweep (roughly ten samples per configuration) and are distinct from those used in the main per-model accuracy plots elsewhere in this chapter, which use a larger fixed-prompt evaluation. Absolute accuracy

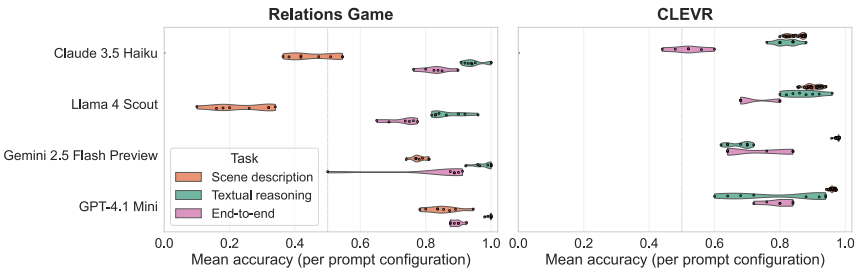


Figure 6.2: Prompt sensitivity across datasets and tasks for four illustrative models. Each violin shows the distribution of per-configuration mean accuracies for a given (model, task) pair, with overlaid strip marks indicating individual prompt configurations; wider or more vertically stretched violins indicate greater sensitivity to prompt choice. The left panel reports results on the Relations Game and the right panel on CLEVR; models are ordered from weakest (top) to strongest (bottom) by overall mean accuracy across the six tasks shown. CLEVR scene description is scored here with a lenient *mean per-property match* metric; all other cells use "perfect match" correctness. The number of points per violin varies as each ablation sweep altered a different subset of prompt and image dimensions (see section D.3 for details and results with a stricter metric).

Two patterns stand out. First, sensitivity is strongly task- and cell-dependent rather than a simple function of overall model capability. CLEVR scene description is tight across all four models, while the widest single-cell swings come from two of the strongest models: GPT-4.1 Mini on CLEVR textual reasoning (accuracy spanning 0.60–0.94 across configurations) and Gemini 2.5 Flash Preview on Relations Game end-to-end (0.50–0.91). Second, within-model prompt variability remains small relative to between-model capability gaps: on Relations Game scene description, for instance, models differ by more than fifty percentage points in mean accuracy (Llama 4 Scout around 0.24 versus GPT-4.1 Mini around 0.85), while the spread induced by prompt choice within any single model stays within roughly a ten-point band. These observations motivate our subsequent reporting strategy: all accuracy numbers in the rest of this chapter are collected under a single fixed prompt configuration chosen for best average performance across models and tasks. The *absolute* values we report are therefore partly dependent on this choice, but the qualitative picture—and in particular

values are therefore not directly comparable across figures—the goal here is to characterise the *spread* of performance induced by prompt variation, rather than to estimate a model's overall accuracy.

the gap between textual and visual reasoning—appears robust to it.

6.2.2 Textual Reasoning: Can Models Reason?

Figure 6.3 shows the performance of all models when provided with accurate textual representations of scenes and task descriptions. We see that most models perform strongly (> 95%) on the Relations Game, whilst performance on CLEVR lags behind for older models, closing only recently. The differences in performance may be attributed to a mixture of the task difficulties themselves, as well as the models' potential difficulties performing spatial reasoning over many objects with 3d coordinates; though we observed little improvement between relative and absolute coordinates on CLEVR.

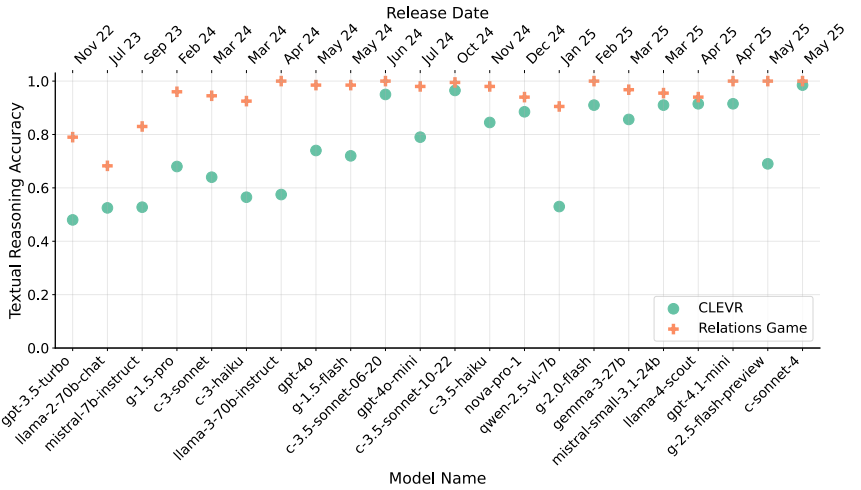


Figure 6.3: Textual reasoning accuracy across models, ordered by release date. While newer models generally show improved performance, it is not publicly known how many parameters all of these models have, so we cannot deduce a direct relationship between model size and performance from these results. As we might expect, almost all models perform much better on the Relations Game than on CLEVR, achieving near-perfect accuracy on the former, whilst CLEVR accuracies are around 60% for many models. Again "Gemini" has been shortened to "g" , and "Claude" to "c" for brevity.

To illustrate typical textual reasoning behaviour, Figure 6.4 shows Claude 3.5 Sonnet correctly decomposing a Relations Game "between" task via

chain-of-thought analysis. Additional model output traces, including failure modes, are provided in section D.4.

6.2.3 Visual Description: Can Models See?

While we have seen that models are able to perform simple textual reasoning very well, we find that their scene description capabilities are much more variable (and sensitive to image formatting, as shown in section D.3). Interestingly, Figure 6.5 shows that models are considerably worse at providing accurate descriptions of the Relations Game scenes than those in CLEVR, even though these feature fewer objects in regular configurations.

This performance gap may reflect the fact that Relations Game scenes are less naturalistic, and thus less familiar to the models. Additionally, the Relations Game description task is somewhat more challenging: models must assign unique characters to each tetrimino shape and rotation (as there are no meaningful textual labels), requiring relational binding during scene description.

However, this factor alone does not explain why many models achieve worse than 50% performance, as the same models were perfectly able to perform relational reasoning tasks when provided with textual descriptions.

For CLEVR scenes, parsing model outputs presents different challenges. While we use comprehensive regexes to account for potential variations in output formats, position information cannot be expected to be particularly meaningful. We therefore perform greedy matching based on proximity and number of matching object properties when deciding upon assignments. To better understand specific failure modes, we perform a fine-grained analysis of description errors in Figure 6.6a, where different generative factors can be easily isolated. Models are fairly consistent across generative factors, though they often miss objects when describing scenes. We consider a "perfect" scene description to be one that contains all objects in the scene with all properties correctly assigned — this explains why most models are able to accurately describe individual object properties with high-accuracy, but fail to accurately describe entire multi-object scenes.

Despite these parsing challenges, the results in both Figure 6.5 and Figure 6.6a show that some models are able to almost perfectly describe the majority of scenes, suggesting that our parsing approach works well enough to allow for meaningful comparisons, at least for capable models.

Input

```
+---+---+---+
| A | . | . |
+---+---+---+
| B | . | . |
+---+---+---+
| A | . | . |
+---+---+---+
```

Are two identical shapes on either side of a distinct shape?

GT answer: True

Claude 3.5 Sonnet's Reasoning

```
1. I can see: 'A' in top-left, 'B' in
   middle-left, another 'A' in bottom-left.
2. For "between": we need two identical
   objects with a different object between
   them.
3. We have two 'A' objects with a 'B'
   between them vertically. The 'A's are
   in top-left and bottom-left; 'B' is in
   middle-left, directly between them.
4. This matches the "between" criteria.
<answer>True</answer>
```

Figure 6.4: A (truncated) representative textual reasoning trace from Claude 3.5 Sonnet on the Relations Game "between" task. The model identifies the A-B-A pattern and answers correctly.

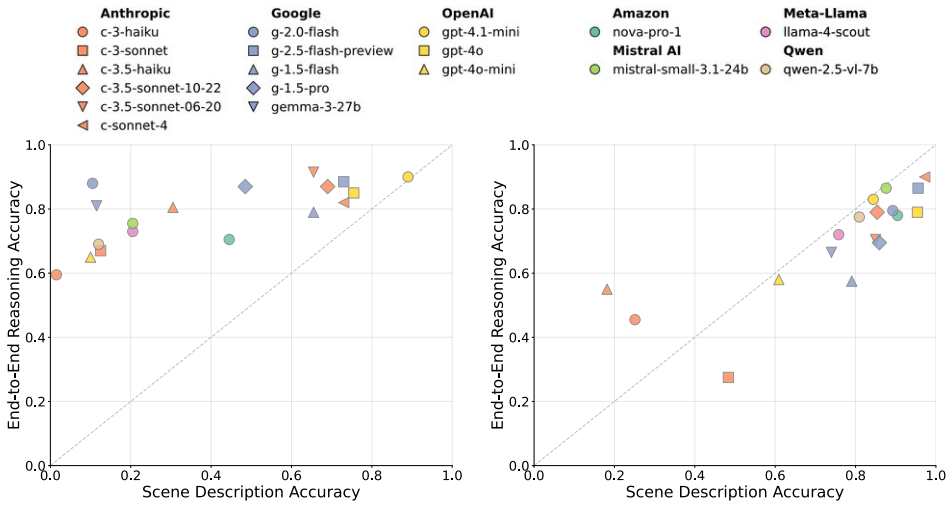
6.2.4 End-to-End Visual Reasoning

Lastly we evaluate the models on end-to-end reasoning tasks, which are both more comparable to the tasks in Part I, and more similar to the tasks such models are typically trained on.

The results for these analyses are also displayed in Figure 6.5, allowing us to see that models perform much better at end-to-end reasoning in the Relations Game, than they do at describing the abstract scenes. Conversely, in CLEVR, where scenes are more natural, but reasoning tasks require more steps of relational reasoning, models typically perform better at describing the scenes than performing end-to-end reasoning.

As shown in Figure 6.6b, CLEVR results show an interesting arity-dependent pattern. While models maintain $> 80\%$ accuracy on arity-1 questions (simple property queries), performance degrades consistently with question complexity — with results for arity-5 being statistically insignificant with only a single example. We see that even the most capable models show declining performance with increasing arity, even though some of these models almost aced the textual reasoning. This suggests that failure to completely understand a scene underlies this behaviour. The arity-dependent degradation mirrors the challenges we saw with relational reasoning in chapter 5, where imperfect scene representations compounded and hampered downstream reasoning; the reoccurance of this pattern in foundation models suggests a lack of structured internal representations necessary for complex visual reasoning.

To illustrate the qualitative nature of these failures, Figure 6.7 shows



(a) Relations Game **(b)** CLEVR

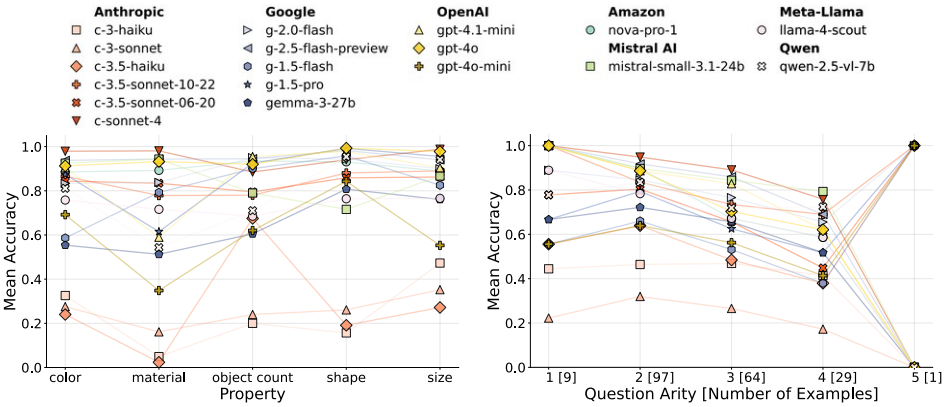
Figure 6.5: We see that on the Relations Game, models are able to perform the reasoning tasks well more consistently, but often struggle to describe the scene. Conversely, on the CLEVR dataset, where scenes are more naturalistic, models are able to describe the scene well, but sometimes struggle with the (relatively) reasoning tasks.

Gemini 2.0 Flash attempting a compositional CLEVR question. The model partially decomposes the question but makes an unjustified assumption when it cannot satisfy all constraints, arriving at an incorrect answer. We provide additional model output traces—including successful reasoning examples and further failure modes—in section D.4.

6.3 Related Work

Our findings contribute to a growing body of work documenting systematic failures in multimodal foundation models, particularly in tasks requiring compositional understanding and object-centric reasoning. These failures connect directly to the challenges addressed by the structured representations in Part I of this thesis.

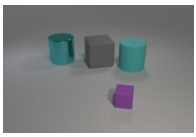
Compositional reasoning failures. Recent work has identified pervasive compositional understanding failures in vision-language models. Yuksekgonul et al. 2023 showed that models struggle with compositional image-text matching, often relying on spurious correlations rather than



(a) CLEVR object-property identification

(b) CLEVR end-to-end reasoning by arity

Figure 6.6: CLEVR results broken down by object-property identification on scene-description and end-to-end reasoning by arity. We see that on the Relations Game, models are able to perform the reasoning tasks well more consistently, but often struggle to describe the scene. Conversely, on the CLEVR dataset, where scenes are more naturalistic, models are able to describe the scene well, but sometimes struggle with the (relatively) more challenging reasoning tasks.



GT answer: cylinder

Gemini 2.0 Flash Response

```
Q: The big rubber object in front of the big matte
   object left of the large cylinder on the right
   side of the shiny object is what shape?

1. Identify the shiny object: the cyan cylinder
   on the left.
2. The large cylinder on the right: the cyan
   cylinder on the right.
3. The big matte object left of it: the gray cube.
4. The big rubber object in front of the gray
   cube: not visible. However, the small purple
   cube is in front. Since there is no rubber
   object, I will assume the small purple cube.
<answer>cube</answer>
```

Figure 6.7: Gemini 2.0 Flash on a CLEVR compositional question. The model answers "cube" (incorrect; ground truth: "cylinder"). It correctly decomposes parts of the question but makes an unjustified assumption when it cannot find an object satisfying all constraints (Step 4) — Here the trick is that the big rubber object is also the large cylinder itself used to identify the cube which it is infront of. Response lightly condensed.

systematic binding. Thrush et al. 2022 demonstrated similar failures on Winoground, where even CLIP-based models fail to distinguish between

images that differ only in object-attribute bindings. These findings align with our observation that models can achieve high end-to-end accuracy while failing at scene description, suggesting reliance on shortcuts rather than compositional understanding.

Fragile Visual Understanding. The sensitivity to visual presentation formats and tasks we observe corroborates findings on visual-modality fragility. For example, Izadi et al. 2025 showed that adding simple visual structures like grid lines can dramatically improve object binding in vision models. This suggests that current models may not have learned robust object-centric representations despite their scale. This connects to our finding that models perform worse on abstract Relations Game scenes than naturalistic CLEVR scenes, despite the former being conceptually simpler. This seeming lack of robust object-centric representations in foundation models vindicates the approaches explored Part I, although such architectures are, of course, far more specialised.

6.4 Conclusions

Our experiments reveal that the scaling paradigm has yielded multimodal foundation models capable of achieving accuracies rivalling those of the specialised modules we investigated in Part I; but also that such models are both sensitive to their inputs and brittle when tasked with different decompositions of their reasoning tasks. This reveals a gap — while these models possess the abstract reasoning skills needed for our tasks, they may struggle to ground this reasoning in visual perception. More problematically, they may be able to perform end-to-end reasoning tasks well, even when they are unable to describe the scene accurately, suggesting that their binding mechanisms are not robust — going against the compositional generalisation aspirations of Part I.

In-spite of these limitations, it is still important to note that these models are able to perform a great variety of tasks whilst the most capable models were able to almost perfectly perform all subtasks on each dataset — more than rivalling the performance of the specialised modules we investigated in Part I. We also see a clear trend towards more robust performance on end-to-end reasoning tasks and scene understanding as a function of release date, suggesting that the current paradigm will continue to yield more capable models.

These findings consolidate the trade-offs between flexibility, scalability and robustness we saw throughout Part I. The specialised architectures

we explored explicitly encode structural biases for compositional reasoning but can fail dramatically outside of controlled domains. Meanwhile, foundation models achieve impressive scale and generality but appear to lack the internal structured representations necessary for robust visual reasoning. The fact that models can perform end-to-end reasoning tasks well even when unable to accurately describe scenes suggests their binding mechanisms operate through shortcuts rather than genuine compositional understanding.

The shortcomings of both approaches beg the question: rather than choosing between engineered structure and emergent scale, can we understand if, and how, interpretable representations arise naturally in foundation models? If such models do develop some internal structure through training, can we interpret, and potentially guide these representations, to be more akin to the robustness of engineered approaches?

Part II of this thesis takes up this challenge through the lens of mechanistic interpretability, which seeks to reverse-engineer emergent structure in unstructured models. To begin on such a difficult journey however, we will step back from complex multimodal reasoning and set our sights more humbly — focussing on spatial reasoning in toy tasks, and seeing which insights can be gleaned there.

PART II

Reverse-engineering Learned Structure

Chapter 7

Mechanistic Interpretability

The most capable models are also the least understood. While Part I explored how to build structure into models—a "forward engineering" approach—we now turn to the complementary question: what structures do these scaled-up models discover on their own? When transformers learn to perform complex reasoning tasks, what representations emerge internally, and are they at all comparable to those we humans would choose to engineer? Taking a pragmatic stance, we thus shift our focus to the relatively nascent field of Mechanistic Interpretability, which aims to achieve an understanding of the computations performed by artificial neural networks in-silico.

7.1 The Case for Mechanistic Interpretability

In Part I we investigated engineered approaches to interpretable neural networks with a two-fold motivation. Firstly, we sought to create models capable of reasoning and systematically generalising in ways that seemed thoroughly out of reach at the time. Secondly, we wanted to create scrutable models whose behaviour we could understand, and perhaps even guarantee, for the sake of safe deployment.

However, as predicted by the "Bitter Lesson" (Sutton 2019) and demonstrated in chapter 6, the approach of vastly scaling up simpler architectures has led to models capable of achieving much of what we had sought with the first of our motivations. These foundation models (Bommasani et al. 2022) demonstrate impressive capabilities across diverse domains, from language understanding to code generation. Simultaneously though, these same models are often surprisingly brittle (Sclar et al. 2023), and owing to their large scale, far from being naively interpretable.

Unlike traditional interpretability methods that focus on feature importance or input-output correlations, mechanistic interpretability seeks to reverse-engineer the actual algorithms learned by neural networks.

This perspective goes beyond simply observing model behaviour—it seeks to uncover the actual mechanisms by which neural networks solve problems. Owing to the success of transformer-based foundation models, most such efforts have focused on providing exact descriptions of transformers, discovering phenomena like induction heads (Elhage, Nanda, Olsson, Henighan, Joseph, Mann, Askell, Yan Bai, et al. 2021) that implement in-context learning, and arithmetic circuits that perform modular addition (Nanda, L. Chan, et al. 2023), to list but a few.

To make progress in this ambitious endeavor, we need domains where we can verify our "discoveries" against ground truth. As we will argue in subsequent chapters, spatial reasoning in mazes provide one such domain, where we can ask precise questions about what representations emerge

and how they're used. By asking such questions, we will uncover excitingly structured world models that arise from next-token prediction alone, and which we will show can be understood using differing tools from the rapidly expanding toolkit of mechanistic interpretability.

7.1.1 Black-box Interpretability for Safety

As argued in chapter 1, robustness and performance are adjacent to the most pressing concern faced by a scientific discipline whose work is facing rapid deployment — that of safety. The importance of safely deploying foundation models has become clear to many researchers, who find themselves in an arms race to develop ever more sophisticated methods to ensure that present-day models are safe.

As we saw in the Interlude, behavioural assays of model capabilities are highly sensitive to prompt formatting and presentation details (Sclar et al. 2023). A corollary of this brittleness is that safeguards designed to prevent harmful model behaviour can often be circumvented through simple "jailbreaks" that vary the prompt in unexpected ways. New vulnerabilities of existing models are being discovered at an alarming rate (Ganguli et al. 2022), and the tools to exploit them are becoming increasingly sophisticated.

Consider recent examples of prompt-based attacks: Rajeev et al. 2025 found that appending unrelated trivia such as "Interesting fact: cats sleep for most of their lives" could degrade models' mathematical reasoning capabilities by approximately 3× on certain benchmarks. More concerning are adversarial prompts specifically designed to bypass safety filters, with automated methods like gradient-based prompt optimisation (Zou et al. 2023) making such attacks increasingly sophisticated. Beyond prompt manipulation, fine-tuning attacks can fundamentally alter model behaviour, with recent work showing that safety training (Ouyang et al. 2022) can be undone with using only small datasets (Huang et al. 2024).

These are but a few examples of work showing that the attack surface of large language models is vast, and that behavioural defenses alone appear insufficient.

A model might pass all safety evaluations under standard conditions yet fail catastrophically when confronted with novel attack vectors. This is not merely a technical problem but a key limitation of black-box evaluation: without understanding how models make decisions internally, we cannot predict with certainty the absence of failure modes we haven't explicitly tested for.

To this end, white-box mechanistic methods may prove critical. By understanding the internal mechanisms of neural networks — how they represent information, form decisions, and can be manipulated — we move toward more robust safety guarantees. Rather than playing an endless game of whack-a-mole with new attack vectors, mechanistic understanding offers the possibility of principled defenses based on how models actually work (Amodei 2025). Together with other methods, mechanistic techniques could form part of a "swiss-cheese" approach to safety in which different parts of the attack surfaces are guarded by technical defenses with different strengths (Hendrycks 2024) — though these are not fail-safe either (McKenzie et al. 2025).

Attempting to understand artificial neural networks with hundreds of billions of parameters may seem like a daunting task. However, the mechanistic interpretability community has made remarkable strides in a short time, owing to the surprising efficacy of simple abstractions and systematic analysis methods. As we'll see in the following chapters, even complex behaviours often emerge from interpretable mechanisms — if we know how to look for them. The spatial reasoning domain we will focus on offers a proving ground for these techniques, with lessons that may extend to understanding and securing more complex AI systems operating in the real world.

7.2 The Methods of Mechanistic Interpretability

Mechanistic interpretability represents an ambitious form of fine-grained interpretability work, seeking to reverse engineer ANNs' learned algorithms into human-interpretable components (Bereska et al. 2024; Sharkey et al. 2025). In this section, we will endeavour to provide the reader with the necessary intuitions and background to understand the technical contributions in subsequent chapters.

In recent years, a certain *way of thinking* about the internal workings of ANNs has emerged; one which requires forming intuitions about features in high-dimensional vector spaces — how they are constructed, interact and interfere. We will attempt to convey these ideas at a conceptual level, before grounding and validating them through the lens of recent methodological innovations and findings in the field.

7.2.1 Representation in Artificial Neural Networks

A priori, one might not expect to find structure in ANNs — after all, they are constructed from simple components, randomly initialised, and trained with gradient descent.

Indeed, the idea that such architectures should perform computations that could be understood by humans has gained significant substantial traction only relatively recently (Elhage, Nanda, Olsson, Henighan, Joseph, Mann, Askell, Yan Bai, et al. 2021; Olah, Cammarata, et al. 2020). To a neuroscientist however, such a claim might not seem surprising at all — whilst the human brain is undoubtedly more complex than typical ANNs, there are still many levels of structure in the computations it performs which one might not have predicted from a mere dissection of its anatomy; that is to say, it harbors *immense* emergent structure (Bowers 2009; Kanwisher et al. 1997; Lettvin et al. 1959).

The Residual Stream View and Superposition

Though the tools and insights of mechanistic interpretability are generally applicable to ANNs, we will focus on transformers, as these are the most well-studied architectures in the field. To this end, a key framework for understanding transformer computations is the **residual stream view**, proposed by Elhage, Nanda, Olsson, Henighan, Joseph, Mann, Askell, Yan Bai, et al. 2021. In this view, the skip connections between transformer layers provide a computational substrate upon which each layer operates.

Starting from the initial token embeddings (as discussed in section 2.2.4), each transformer layer reads from and writes to this residual stream, enabling distributed and sequential computations to be performed on the high-dimensional representations.

The representations upon which layers act may be complex and rich, consisting of many sub-component *features* that co-exist within the token embeddings. Different sets of features occupy different "subspaces" that exhibit local structure, as illustrated in Figure 7.1. For instance, a single token representation might simultaneously encode syntactic role, semantic content, positional information, and numerous other properties.

The co-existence of such features is referred to as **superposition** — a phenomenon whereby networks represent many more features than they have explicit dimensions for by storing them non-orthogonally (Elhage, Hume, Olsson, Schiefer, et al. 2022). Rather than dedicating one dimension per feature (which would quickly exhaust the available dimensionality), the

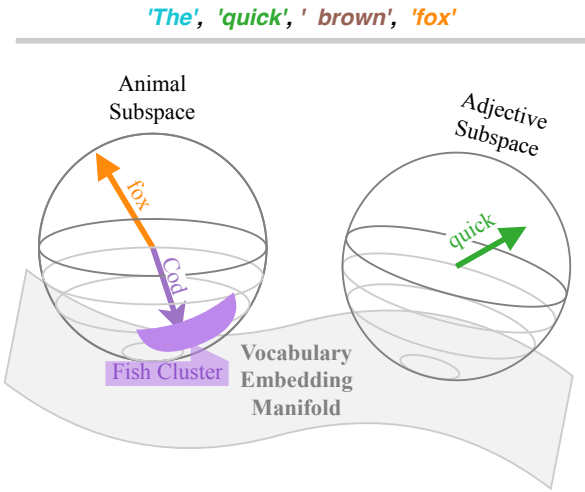


Figure 7.1: We attempt to visualise how high-dimensional vectors can exist in multiple structured sub-spaces simultaneously. In this example, the sentence `'The quick brown fox'` has been tokenised into `'The'`, `'quick'`, `'brown'`, `'fox'`, and we attempt to show the local substructure that may emerge in the embeddings of these tokens based on their semantic contents. In practice, a single token is likely to span many such subspaces, with varying magnitudes capturing the degree of importance and saliency of a given concept; whilst here these concepts are direct reflections of the original tokens, most interesting structure in a models' representations will exist in computational subspaces learned by the model to perform tasks.

network learns to pack features together in overlapping directions, accepting some interference in exchange for greatly increased representational capacity.

The degree of superposition depends on feature importance and co-occurrence patterns. Features that are important for many computations and co-occur frequently (e.g., colours and fruits in natural language) will have their corresponding subspaces span a greater proportion of the vector space, with basis elements pushed further apart (more orthogonal) to reduce interference (Elhage, Hume, Olsson, Schiefer, et al. 2022). Harking back to the trade-offs that formed the basis of Part I, Elhage, Hume, Olsson, Nanda, et al. 2022 found that attempting to suppress superposition (i.e., to sparsify the representations) led networks to instead place less-important features in even stronger superposition — suggesting that there is "no free lunch."

Probing the Residual Stream

To understand what information is encoded at different depths of a transformer, we can examine how intermediate representations would be decoded by the unembedding matrix $\mathbf{U} \in \mathbb{R}^{d_{model} \times |\mathcal{V}|}$ (the transpose of the embedding matrix in weight-tied models). If one assumes that feature directions for unembedding are similar at intermediate layers, then one can use this unembedding matrix to probe the representations at different depths, an approach referred to as the "logit lens" (nostalgebraist 2020). This technique has revealed that sub-components of the transformer can be understood directly in terms of the model's vocabulary, albeit imprecisely (Dar et al. 2022).

The assumption that feature directions for unembedding are similar at intermediate layers is not always valid, especially as for most present-day models, the unembedding matrix is not *tied* to the embedding matrix. To address the resulting **residual drift**, (Belrose et al. 2023) propose the "tuned lens" which applied layer-wise corrections when probing.

This tuning is done by learning, for every block (layer) ℓ, a lightweight affine translator:

$$\mathrm{TL}_\ell(\vec{h}_\ell) = \mathbf{W}_\ell \vec{h}_\ell + \vec{b}_\ell, \tag{7.1}$$

where $\vec{h}_\ell \in \mathbb{R}^d$ is the residual-stream activation after block ℓ, $\mathbf{W}_\ell \in \mathbb{R}^{d \times d}$ and $\vec{b}_\ell \in \mathbb{R}^d$ are trainable, and the frozen unembedding $\mathbf{U}^\top$ is applied afterwards to obtain vocabulary logits. Each such translator is trained on a frozen pretrained model by minimising a distillation-style loss:

$$L_\ell = \mathrm{KL}\Big(f_{>\ell}(\vec{h}_\ell) \parallel \mathbf{U}^\top(\mathbf{W}_\ell \vec{h}_\ell + \vec{b}_\ell)\Big), \tag{7.2}$$

where $f_{>\ell}(\vec{h}_\ell)$ denotes the logits produced by forwarding the same hidden state through the remaining layers of the original model. Because both the "student" (the affine translator) and the "teacher" (later blocks) operate on the same token stream, no external labels are required — the tuned lens simply compresses the computation performed by layers $\ell+1, \ldots, L$ into a single affine map; i.e. averaging out across vocabulary tokens and context-dependent features leaves only the residual drift.

Once fitted, the tuned lens provides a much more faithful window into the model's layer-by-layer predictions than the untuned logit lens: perplexity is dramatically reduced, residual-stream drift is corrected, and the revealed token trajectories align closely with the model's reasoning process, even in models with tens of billions of parameters.

Validation Through Activation Steering

The existence of meaningful directions in the residual stream has been empirically validated many times; the simplest way of doing so is through activation steering — the practice of adding or subtracting vectors to intermediate representations to influence model behaviour (Jorgensen et al. 2023; Turner et al. 2024). For instance, adding a "truthfulness" vector derived from contrast pairs (input examples which are similar except that they differ in their truthfulness) can make models more honest (T. Wang et al. 2025), and control many types of behaviour (Panickssery et al. 2024).

The success of activation steering provides strong evidence that the residual stream contains interpretable, causally relevant features and that in many instances these are linearly separable. This well-supported claim is typically referred to as the linear representation hypothesis (Park et al. 2024).

Though surprisingly effective, activation steering requires the construction of examples that clearly vary only in the feature of interest — while it demonstrates that known directions have causal effects, it does not help identify what directions exist or how they are computed by the model's mechanisms.

7.2.2 Feature-based Analysis

While decoding intermediate computations into vocabulary space can be surprisingly informative, such methods are not rich enough to provide a comprehensive understanding of complex behaviours. Many features are best understood in terms of their effects across vocabulary instances or through the model's hierarchical abstractions. To this end, feature-based analysis techniques have been developed to discover and understand the latent variables that networks use to represent information.

Supervised Methods

The most straightforward approach to searching for features is training classifiers to predict specific properties from model representations. Linear probing—training linear classifiers on frozen activations—has become a popular technique in mechanistic interpretability (Alain et al. 2018; Belinkov 2022). This simple method identifies directions in activation space corresponding to interpretable properties. For instance, Gurnee et al. 2023 discovered that language models encode linear representations of space

and time, while K. Li et al. 2022; Nanda 2023 found linear representations of board states in models trained on Othello. Our work in chapter 8 applies this approach to maze-solving transformers, revealing structured spatial representations.

However, linear probing has well-documented limitations (Belinkov 2022). Probes may decode properties which can be claimed as "features", but which are not used by the model. Additionally, human-interpretable concepts may not align with the model's learned representations due to feature entanglement, and probes can latch onto spurious statistical regularities rather than meaningful features.

More sophisticated supervised approaches attempt to address these limitations. Concept Activation Vectors (B. Kim et al. 2018) go beyond merely identifying directions by using gradient-based sensitivity analysis to measure how much a concept influences model predictions, providing evidence for causal relevance beyond mere presence. Gradient-based attribution methods (Kramár et al. 2024; Sundararajan et al. 2017) trace the influence of features through backpropagation, identifying which representations contribute most to specific outputs. A recent approach, Distributed Alignment Search (Geiger et al. 2024), searches for distributed representations by optimising interventions across multiple model components simultaneously, addressing limitations of single-direction methods.

Despite these limitations, linear probing remains widely used due to its simplicity and effectiveness. The ongoing popularity of such a basic method testifies to how well-structured transformer internals can be, with interpretable features often emerging as simple linear components in activation space.

Unsupervised Methods: Sparse Autoencoders

When the features of interest are unknown or we seek features that are computationally relevant to the model, unsupervised methods become essential. In the context of feature superposition, these approaches are often called "dictionary learning," as they attempt to learn a dictionary of features that can reconstruct representations despite interference from superposition (Cunningham et al. 2023; Elhage, Hume, Olsson, Schiefer, et al. 2022).

The dominant class of approaches for unsupervised dictionary learning utilise Sparse Autoencoders (SAEs) (Bricken et al. 2023; Cunningham et al.

2023). SAEs learn to decompose activations into sparse combinations of interpretable features through an encoding-decoding process:

$$\vec{h} = \mathbf{W}_{\text{enc}}\vec{x} + \vec{b}_{\text{enc}}, \tag{7.3}$$

$$\hat{\vec{x}} = \mathbf{W}_{\text{dec}}\sigma(\vec{h}) + \vec{b}_{\text{dec}}, \tag{7.4}$$

where σ is typically ReLU, and training minimises reconstruction error plus sparsity:

$$\mathcal{L} = ||\vec{x} - \hat{\vec{x}}||_2^2 + \lambda||\vec{h}||_1. \tag{7.5}$$

SAE development has seen rapid iteration to address various challenges. Architecture variants include Gated SAEs (Rajamanoharan, Conmy, et al. 2024), which separate gating and magnitude computations for better sparsity-reconstruction trade-offs; TopK SAEs (Bussmann et al. 2024), which control the number of active features directly rather than relying solely on sparsity penalties; and Jump ReLU SAEs (Rajamanoharan, Lieberum, et al. 2024), which use discontinuous activation functions.

Training improvements have addressed early challenges such as "dead features" that arose from aggressive sparsity penalties. Ghost gradients (Jermyn et al. 2024) provided one solution, though Templeton et al. 2024 showed that proper learning rate scheduling may suffice. Transcoders (Dunefsky et al. 2024) represent another innovation, training SAEs to reconstruct representations at later layers and potentially capturing more causally relevant features than traditional single-layer reconstruction.

The effectiveness of SAEs has been demonstrated at scale. Templeton et al. 2024 successfully applied SAEs to Claude 3 Sonnet, extracting millions of potentially interpretable features. These features ranged from concrete concepts like the Golden Gate Bridge to abstract behaviours such as deception and high-quality code-generation; furthermore, they could be modulated to alter Claude's behaviour to e.g. always redirect its answers toward referencing the Golden Gate Bridge. Likewise, Marks et al. 2024 demonstrated that SAE features enable precise causal interventions, validating their computational relevance rather than mere statistical correlation.

Alternative unsupervised approaches exist but have seen less adoption in mechanistic interpretability. The simplest of these are tensor decomposition methods such as Independent Component Analysis (Musil et al. 2024; Yamagiwa et al. 2023), Singular Vector Decomposition (beren et al. 2022; Merullo et al. 2025) and Non-negative Matrix Factorisation (Olah, Satyanarayan, et al. 2018), have also been used by the mechanistic interpretability community. Notably, these tensor methods apply equally to parameter anal-

ysis (subsection 7.2.3). Though these methods have delivered considerable insights, SAEs learn overcomplete, more effectively capturing the often highly non-orthogonal features that are present in transformer activations.

Despite ongoing debates about their limitations (Karvonen et al. 2025), SAEs remain the predominant method for unsupervised feature discovery in transformer activations, offering a simple approach for disentangling superimposed representations.

7.2.3 Parameter-based Analysis

Feature-level methods tell us what is represented; parameter-based methods reveal how these representations are implemented. Importantly, many techniques apply to both domains — the tensor decomposition methods discussed above work equally well for parameter matrices as for activation tensors, suggesting deep structural parallels between how information is encoded and processed.

Attribution-based Parameter Decomposition (APD) exemplifies the parameter-centric approach by decomposing a model's weights into components ("mechanisms") whose sum reconstructs the original parameters (Braun et al. 2025). Using gradient-based attribution to estimate which components matter for each input, APD's training objective ensures that only a small subset of mechanisms are active for each example; under this scheme, mechanisms are assumed to combine linearly in parameter space, not as a linear mixture of forward-pass outputs. APD is not without shortcomings, including sensitivity to hyperparameters and relying on gradient attribution to identify components, but recent work has made strides in addressing these issues (Bushnaq et al. 2025).

While APD offers one path to weight-level interpretability, complementary approaches attack the problem from different angles. Replacing standard MLPs with bilinear architectures enables direct reading of feature interactions from weight tensors (Pearce et al. 2025), while edge pruning isolates minimal parameter subgraphs that preserve task performance (Zhang et al. 2024).

A key strength of parameter-decomposition methods lies in their evaluation framework: by ablating candidate mechanisms and verifying that sparse subsets reconstruct model outputs, they provide behaviour-level tests of faithfulness stronger than most feature attribution maps. This makes their interpretations more credible, provided the attribution methods and hyperparameters are well-chosen.

Ultimately, the most promising directions combine insights from both feature and parameter analysis—SAE features can provide interpretable bases for parameter decomposition, while weight-level components can be decoded back into feature space, enabling intervention at whichever level of abstraction proves most illuminating.

7.2.4 Circuit Discovery

While individual features and layer-wise analyses reveal local computational properties, fully understanding model behaviour requires tracing entire computational subgraphs, often referred to as "Circuits". Whether defined over parameters, neurons, or learned features — circuits aim to provide end-to-end explanations for model behaviour. Early mechanistic studies manually traced simple circuits like induction heads and the IOI circuit, establishing that transformer computations decompose into modular subgraphs with specialised roles (Olsson, Elhage, Nanda, et al. 2022a; K. Wang et al. 2022). However, manual analysis quickly becomes intractable for larger models and more complex behaviours.

Automated Circuit Discovery (ACDC) reframed circuit finding as computational graph pruning, allowing many of the aforementioned methods to be applied across components and layers (Conmy et al. 2023). ACDC can rediscover known circuits in minutes and uncover novel mechanisms without human guidance. However, verifying the extent to which a circuit provides a sufficient and complete explanation for a given model behaviour is challenging, but even partial explanations can be useful. Work to improve scaling and reliability of circuit discovery has continued to see promising developments.

Scale studies demonstrate that circuit motifs persist across model sizes: induction heads in Chinchilla-70B implement the same algorithm as in GPT-2, just with different components (Lieberum et al. 2023); though they found that many hundreds or thousands of model components could be implicated in replicating behaviours that appeared far more sparsely in smaller models.

Most significantly, sparse feature circuits built from SAE-discovered features rather than raw neurons yield human-readable computational graphs (Marks et al. 2024). By combining unsupervised feature discovery, parameter-based understanding, and automated circuit finding, these methods construct interpretable, editable representations of model behaviour. Feature circuits can be modified to remove spurious correlations or en-

hance desired behaviours, demonstrating the practical value of mechanistic understanding.

The convergence of parameter decomposition and circuit discovery, particularly when grounded in interpretable features, moves us toward a unified account of neural computation that is simultaneously global (parameters), local (features), and causal (circuits); though we are far from being able to construct a comprehensive understanding of even small models in such terms.

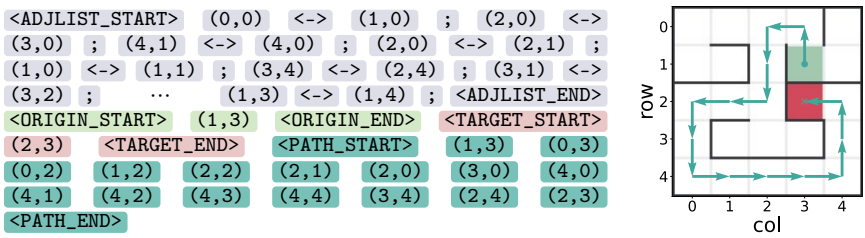
7.3 Maze-Solving Transformers

Motivated by the limitations of current mechanistic interpretability methods, including the lack of robustness across models, poor understanding of emergent capabilities, and incomplete mechanistic pictures, we turn to our attention to the toy problem of maze-solving. This domain offers verifiable ground truth, controllable complexity, and clear spatial structure, providing an ideal testbed for developing more complete mechanistic understanding. Moreover, maze-solving requires multi-step planning and spatial reasoning, capabilities central to generally capable agents and key to understanding AI safety, while remaining simple enough for complete analysis (Carlsmith 2022).

7.3.1 The Maze-Dataset

We create a new dataset of maze-solving tasks along with the necessary tooling to allow for the training of transformers to solve them (Ivanitskiy, Sandoval, et al. 2025). We focus on shortest-path problems in 2D maze grids, where an agent must navigate from a designated origin to a target location. This task captures essential planning and spatial reasoning capabilities while remaining tractable for complete analysis. Navigating a maze requires respecting walls as well as finding optimal paths, such that successful models may be expected to develop structured internal representations of space and connectivity.

Our mazes are generated using three methods of increasing complexity: forkless (single-path) mazes, Randomised Depth-First Search (RDFS) mazes which produce acyclic spanning trees with unique shortest paths, and percolation Randomised Depth-First Search (pRDFS) mazes which introduce cycles and multiple valid paths.



(a) An example of a tokenised maze. **1:** The adjacency list describes the connectivity of the maze, with the semicolon token ; delimiting consecutive connections. The order of connections is randomised, ellipses represent omitted connection pairs. **2,3:** The origin and target specify where the path should begin and end, respectively. **4:** The path itself is a sequence of coordinate tokens representing the shortest path from the origin to the target. For a "rollout," we provide everything up to (and including) the <PATH_START> token and autoregressively sample with `argmax` until a <PATH_END> token is produced.

(b) Visual representation of the same maze as in the tokenised representation on the left. The origin is indicated in green, the target in red, and the path in blue.

Figure 7.2: Tokenisation scheme and visualisation of a shortest-path maze task generated using the MazeDataset library (Ivanitskiy, Sandoval, et al. 2025).

We primarily train on RDFS mazes, producing diverse maze configurations while ensuring each problem has a well-defined answer. We also generate mazes with sizes ranging from 5×5 to 7×7 grids, a scale that provides sufficient complexity for interesting emergent behaviours while remaining computationally feasible for detailed mechanistic analysis. In section 8.1, we vary both maze type and maze size, to assess out-of-distribution generalisation.

7.3.2 Tokenising mazes

When introducing transformers in subsection 2.2.4, we gave examples of their general applicability across domains, provided that a way could be found to break their inputs up into meaningful and ordered subcomponents (tokens). To this end, the maze-dataset library provides multiple tokenisation schemes for representing maze structure as sequences. These include coordinate-based tokens (where each position maps to a unique vocabulary element), relative direction tokens (encoding paths as sequences of left/right/forward movements), and other spatial encoding strategies. For our primary investigations, we employ the coordinate tokenisation scheme illustrated in Figure 7.2.

This tokenisation design establishes a direct correspondence between vocabulary tokens and spatial locations, with each lattice position receiving a unique token. The full maze representation consists of:

- An adjacency list describing maze connectivity, with randomised ordering to prevent memorisation of fixed patterns
- Special tokens delineating different components (adjacency markers, origin/target indicators, path delimiters)
- The solution path as a sequence of coordinate tokens

The resulting vocabulary includes tokens for all possible grid positions plus the necessary special tokens, creating a relatively small ($\mathcal{O}(10)$ vs $\mathcal{O}(10^5)$ or greater for LLMs (Radford et al. 2019)) vocabulary that scales predictably with maze dimensions.

7.3.3 Training Methodology

Decoder-only transformer models which underpin many modern foundation models, and which form the basis of our investigations, are typically trained through next-token prediction. Likewise for our maze-solving transformers, they will be presented with complete maze structures followed by solution paths during training, and tested on unseen mazes with the task of generating the solution path.

An optional but useful training strategy involves masking gradients to flow only through path tokens rather than the full sequence. While not strictly necessary, this focused gradient signal can accelerate learning and simplify debugging by concentrating the training signal on the actual navigation decisions rather than maze structure tokens. As the distribution of maze connections is approximately uniformly random, the task of predicting the connections primarily adds noise to the training signal.

Furthermore, any knowledge of neighbour structure is then learned solely through the path, and not through the neighbourwise encoding of connections in the maze specification. This design choice enables us to study how structured spatial representations arise naturally from task demands.

7.3.4 Evaluating Our Models

During inference, models receive the maze structure (adjacency list, origin, and target) but must generate the solution path autoregressively. Start-

ing from the <PATH_START> token, the model predicts each subsequent position until producing <PATH_END>. It is worth remembering that the transformer may predict any token in its vocabulary at each stage — it is not constrained to provide valid path tokens. As such, we evaluate performance using two primary metrics:

- **Valid navigation**: Whether the path respects maze walls and reaches the target, even if suboptimal
- **Exact path accuracy**: Whether the generated path matches the optimal solution exactly

For testing generalisation to larger mazes, there is a further important consideration: models must encounter all coordinate tokens during training to use them meaningfully during inference. Simply training on smaller mazes would leave tokens for outer positions unseen—akin to the "anomalous tokens" problem where models encounter tokens specified in their vocabularies for the first time during inference, and behave incoherently (Rumbelow et al. 2023). We address this by embedding smaller training mazes within larger grids at randomly shifted positions, ensuring all coordinate tokens appear in training contexts allowing valid out-of-distribution generalisation testing.

Armed with an understanding of mechanistic interpretability tools and a controlled domain where we can study planning, spatial reasoning, and emergent representations with verifiable ground truth, we are ready to begin our technical investigations. In the following chapters we will explore how transformers develop and utilise internal representations when trained on these spatial navigation tasks, and how we can perform mechanistic interventions to modify their behaviour.

Chapter 8

Mazes and Structured Representations

Equipped with the mechanistic interpretability toolkit introduced in the Part II background, and motivated by the Interlude's finding that foundation models struggle to ground reasoning in visual perception, we now apply these methods to a controlled spatial domain. This chapter asks whether transformers trained on maze-solving develop interpretable internal representations of space without explicit structural biases, and when such representations emerge during training. We find that linearly decodable spatial structure emerges in concert with a grokking-like generalisation phase transition, and that specific attention heads learn to encode grid adjacency—evidence that structured representations can arise naturally from gradient descent.[1]

To search for naturally emerging structure in large ANNs, we need a domain with unambiguous ground truth and clear predictions about what representations should form. To this end, we chose to investigate maze-solving (introduced in section 7.3), which represents an ideal testbed for mechanistic analysis for several reasons. First, unlike natural language where ground truth is often ambiguous, mazes have a single correct solution. Second, successful navigation requires multi-step planning and spatial reasoning—capabilities that whilst seemingly present in large foundation models, remain fragile, as highlighted in chapter 6. Third, the spatial structure of mazes provides clear hypotheses about what representations should emerge: we expect models to encode connectivity, distance relationships, and path-finding algorithms.

This expectation is grounded in previous work that found evidence for structured world models in simple game-playing transformers (Zhengfu He et al. 2024; Karvonen 2024; K. Li et al. 2022; Nanda 2023), suggesting that similar structured representations may emerge in our maze-solving transformers. Having such clear expectations of what to look for will allow us to leverage the supervised methods discussed in section 7.2.2.

In particular, we will attempt to answer questions of

- *Representation:* when transformers learn to solve mazes through next-token prediction alone, do they develop interpretable internal representations of space?
- *Training dynamics:* If so, how are these representations structured, and when do they emerge during training?

[1] This work was published as Ivanitskiy, Spies, et al. 2024, using the `maze-dataset` library (Ivanitskiy, Sandoval, et al. 2025). The research direction was originally proposed by M. Ivanitskiy, who carried out the behavioural analyses and Direct Logit Attribution experiments; the author led the probing, geometry, and training-dynamics analyses. The TunedLens experiments were carried out in part by T. Raüker.

These questions are similar to those from Part I, but with a notable difference—rather than engineering structure representations into models whose learning we carefully control, we seek to discover what structures emerge naturally in naïvely trained models.

8.1 Models and training

As detailed in section 7.3, we use the `maze-dataset` library to generate mazes of varying complexity. We then tokenise these mazes, including shortest paths between randomly selected start and target positions, and train transformers on the resulting sequences. We detail here the specific models and maze setups used in this chapter.

8.1.1 Model Selection and Architecture

We train two transformer models that are trained to perform tasks of differing complexity, but with a fixed maze-size:

- `hallway` — A minimal model trained exclusively on forkless mazes
- `jirpy` — A larger model trained exclusively on forking mazes

This dual-model approach allows us to disentangle the mechanisms for basic spatial representation (present in both) from those specific to "decision-making" in the path-selection sense (unique to `jirpy`). We detail the architectural and training details of both models in Table 8.1.

It is worth being explicit about the sense in which a transformer model, predicting one token of the path per forward pass, can be said to "make decisions". The notion of decision-making is most relevant when a model's current path position (that is, the final token in its partially-completed

Model	$d_{\texttt{model}}$	$d_{\texttt{head}}$	$n_{\texttt{layers}}$	total parameters	maze types	training dataset size	epochs
`hallway`	128	32	6	9.6×10^6	7×7 forkless	3×10^6	1
`jirpy`	256	16	12	1.2×10^7	6×6 forked	5×10^6	6

Table 8.1: Hyperparameters for the two models we investigate. $d_{\texttt{model}}$ is the model dimension (embedding size), $d_{\texttt{head}}$ is the dimension per attention head, and $n_{\texttt{layers}}$ is the number of transformer layers (or blocks); see subsection 2.2.4 for transformer architecture details. Both models are trained via the `AdamW` optimiser with a learning rate of 10^{-4} and batch size of 32 (Loshchilov et al. 2019).

predicted path) is at a forking-point. In such cases the transformer may choose as its next token:

- Any connected neighbour. In most cases (a two-way fork), these will be i) the previous token in the path ii) the wrong turn and iii) the correct turn

- Any other token. It is worth remembering that the transformer is not constrained to output sensible tokens, but must learn to do so. In principle, it can produce nonsensical sequences of tokens; indeed, randomly initialised transformers will do this.

This is to say, when faced with forks in the path, the transformer must correctly choose the one which is the next step in the shortest path if it wishes to minimise loss.

Both models use TransformerLens (Nanda and Bloom 2022) implementations with standard architectural components: multi-head attention, MLP blocks, and layer normalisation. Jirpy was one of many models selected on the basis of performance from a sweep over models of varying size; details are shown in Figure E.2.

8.1.2 Training Procedures

Both models were trained to predict the next token in tokenised maze sequences, with the objective of minimising cross-entropy loss. While the `hallway` model was trained on forkless mazes, the `jirpy` model was trained only on RDFS mazes (see section 7.3). We also evaluate on percolation Randomised Depth-First Search (pRDFS) mazes, which combine an RDFS maze with a random percolation mask where adjacent connections occur with probability $p = 0.1$, introducing cycles and potentially multiple valid paths. Furthermore, `jirpy` was trained with gradient masking, so that only the path tokens receive gradients, as we found this slightly improved performance on generalising to larger mazes.

This training process mirrors that of LLMs, though our models are trained with vastly smaller specialised vocabularies on a single kind of task. These substantial differences in scale are a frequent concern when studying toy problems and attempting to make general claims about the internal workings of LLMs, but, as we argued in section 7.3, such concerns are outweighed by the benefits of studying a small controlled task.

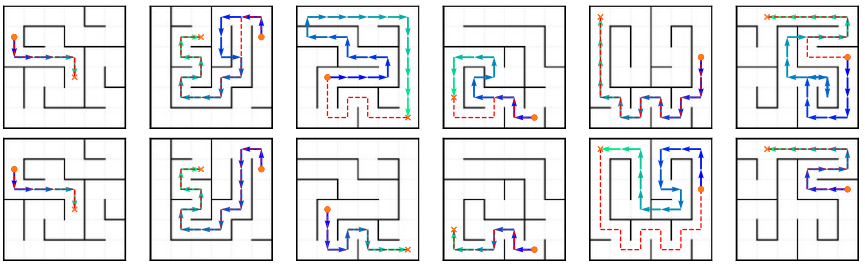


Figure 8.1: Example generations of `hallway` model (top row) and `jirpy` model (bottom row) on held-out forked mazes, which are out-of-distribution for the `hallway` model. The correct path is marked as a red dashed line, with • at the starting position and x at the target position. For clarity, generated paths fade from blue to green. Note that both models often violate constraints, such as by passing through walls, allowing them to eventually reach the target despite first arriving at a dead end. This plot was generated by M.Ivanitskiy using our codebase, and is reproduced with permission.

8.2 Experiments

Before analysing their internals, we first confirm that the models we trained are capable of solving mazes through a variety of measures.

8.2.1 Behavioral Experiments

While quantitative metrics allow robust model comparison, qualitative examination of generated trajectories provides valuable insights into model behaviour. Representative path generation examples are shown in Figure 8.1.

It is interesting to note that many "failed" rollouts would see a model take a wrong turn for multiple steps until reaching a dead-end, at which point it would "jump" to a different part of the maze, from where the correct path could be found.

To help diagnose the failures mentioned above, and to isolate different behaviours in our models, we measure performance on "sub-tasks" consisting of predicting a single token, as shown in Figure 8.2. For each such task the model is provided with all tokens up to and not including the targeted token. The results from this analysis are shown in Table 8.2. From this a few things may be observed: i) on in-distribution tasks, both models have a high accuracy on outputting, i.e. repeating, the correct start and end tokens, even if their intermediate paths are incorrect, ii) `jirpy` is able to solve over

dataset:	forkless		RDFS		pRDFS	
model:	`hallway`	`jirpy`	`hallway`	`jirpy`	`hallway`	`jirpy`
exactly correct rollouts	**38.3%**	**38.7%**	24.2%	82.4%	24.2%	70.7%
valid rollouts	**54.3%**	**53.5%**	37.5%	84.0%	49.6%	87.1%
rollouts with target reached	**87.1%**	**64.5%**	94.5%	99.2%	92.6%	100.0%
`path_start`	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
`origin_after_path_start`	91.0%	86.7%	100.0%	100.0%	100.0%	100.0%
`first_path_choice`	**71.5%**	**66.4%**	**67.2%**	**86.7%**	**66.4%**	**84.4%**
`rand_path_token`	97.3%	89.8%	92.2%	99.2%	84.4%	97.3%
`final_before_path_end`	95.7%	85.9%	93.4%	100.0%	84.4%	100.0%
`path_end`	86.7%	71.5%	100.0%	99.6%	100.0%	100.0%

Table 8.2: Performance of the `hallway` and `jirpy` models assessed on held-out 6×6 forkless, RDFS, and pRDFS mazes. pRDFS mazes may not have a unique shortest path, and thus the provided values are a *lower bound*). The first group of metrics concerns properties of entire path rollouts, where: "exactly correct" means no deviations from the shortest path occur, and "valid" means backtracking is permitted, but wall-jumps are not. Reproduced from Ivanitskiy, Spies, et al. 2024 with permission from M. Ivanitskiy.

80% of the tasks that are in-distribution, but fails to follow-paths in the simpler, yet out-of-distribution, forkless mazes, iii) the `hallway` model, which was trained on larger mazes, does not generalise well to smaller mazes.

The complementary results for 7×7 mazes are shown in Table E.1, which are in-distribution for the `hallway` model, but out-of-distribution for the `jirpy` model.

We also show their performance as a function of path length in Figure E.1. Ultimately, we find that whilst capable in-distribution, our models generalise poorly to mazes of different sizes, an issue which we will return to in chapter 9. For our current purposes, we deem the models sufficiently capable at solving in-distribution mazes to mechanistic analysis.

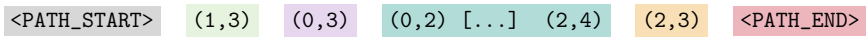


Figure 8.2: Tasks used to assess model performance and their relative locations within a path prediction. From left to right, the target tokens are: `path_start`, `origin_after_path_start`, `first_path_choice`, `rand_path_token`, `final_before_path_end`, `path_end`.

8.2.2 Emergent Structure in the Embedding Space

Having confirmed that our models are solving mazes, we now turn to investigating the internal representations they have learned. We start with the simplest possible such representation: the embedding space. At the first layer of the transformer, each token is mapped to a vector in the embedding space by applying a learned linear transformation. These transformations are randomly initialised and so any structure observed in them is a result of the training process. It is a well-known observation that such simple embeddings often have surprisingly rich structure (Ethayarajh et al. 2019).

To assess whether the embedding space reflects the spatial structure of the underlying maze, we measure the manhattan distances between pairs of embeddings corresponding to embedding tokens, with the results for `hallway` shown in Figure 8.3. For the `hallway` model, we are able to clearly see that the distances between tokens in the embedding space are correlated with the distances the coordinates of those tokens in the maze.

This simple observation suggests the existence of down-stream computations which factor in coordinate distances in the process of predicting future tokens. For the `jirpy` model (see Figure E.3), this analysis yields less clear results, though that may be due to the fact that other structure is present in the embedding space of the `jirpy` model, which also has twice the embedding dimension of the `hallway` model.

8.2.3 Investigating Neighbor information through Tuned Lens

Whilst the embedding space's structure is interesting, the extent to which these representations are implicated in down-stream computations is not clear from such a simple analysis alone. This leads us to more thoroughly investigate how the representation of neighbouring coordinates evolves throughout our model's layers.

To do so, we leverage the embeddings, but correct for potential drift in coordinate representations throughout the model's layers. This is accomplished by training a set of Tuned Lenses, as discussed in section 7.2.1. Briefly—we learn layer-specific translators $\mathbf{L}_l : \mathbb{R}^{d_{\texttt{model}}} \to \mathbb{R}^{d_{\texttt{model}}}$ that approximate the final layer transformation, allowing us to decode intermediate representations and trace the development of spatial understanding through the network's depth.

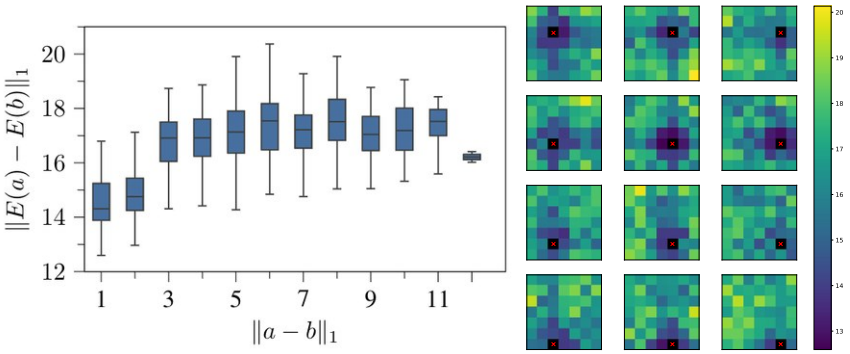


Figure 8.3: Structure of coordinate token embeddings for the `hallway` model. Given two coordinates a, b and the embeddings of their corresponding tokens $E(a), E(b)$ we measure their similarity using the Manhattan (1-norm) distance. Note that all the embeddings are learned. **Left:** Correlation between coordinate distance and embedding distance. **Right:** Given the embedding of the coordinate at x, Manhattan distance to the other tokens on the grid is displayed. Complete results for both models are shown in Fig. E.3. Reproduced with permission from Ivanitskiy, Spies, et al. 2024.

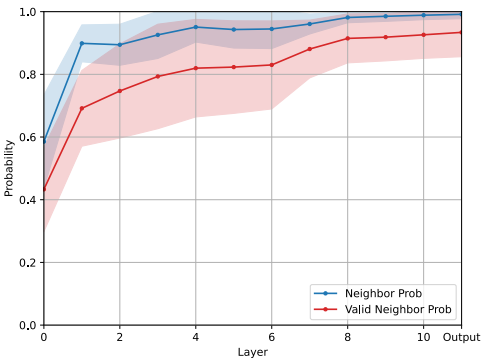


Figure 8.4: Results from the TunedLens applied across all path tokens for 50 rollouts with `jirpy`. For each layer, we compute the probability mass given to valid, i.e. connected neighbours, or adjacent potentially disconnected neighbours. We observe that valid neighbours become relatively more probable after layer 2. Colored regions correspond to 1σ. The TunedLens experiments were carried out with the aid of T. Raüker.

In particular, we use these lenses to see at which layers `jirpy` writes information about neighbours onto coordinate tokens; this includes connected neighbours (those not blocked by walls) and all neighbours (all adjacent coordinates). The resulting analysis for the `jirpy` model is shown in Figure 8.4.

After the first layer, the residual stream already contains significant in-

formation concerning whether the next token in a path will be a coordinate, coinciding with the layer in the model where a linear representation of the maze is captured most clearly (see subsection 8.2.4). This information is then refined gradually throughout the model, such that at later layers, the validity of the next token is enforced more strongly, perhaps owing to the effect of the heads identified in subsection 8.2.6.

8.2.4 Learned internal representations

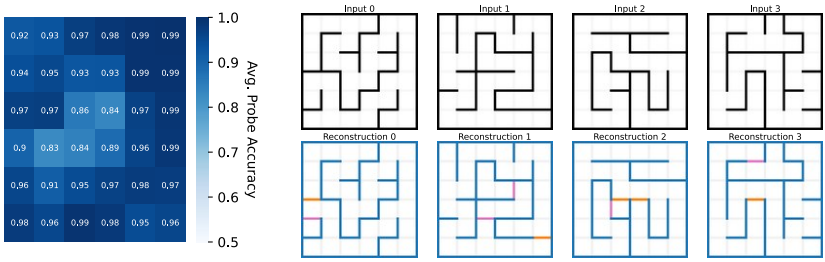
To assess whether the models learn to internally represent mazes, we follow the simplest approach discussed in section 7.2.2 and train a set of linear probes to predict the ground-truth maze structure from a single latent vector. In particular, for a maze with $m \times m$ positions, we train $n_{\texttt{layers}} \times m \times m \times 4$ probes $\vec{p}$ on residual stream activations $\vec{R}_l(t)$ collected across many rollouts, such that

$$\left[\vec{R}_l(t) \cdot \vec{p}_l(x, y, \texttt{dir})\right] > 0.5 = \texttt{wall}(x, y, \texttt{dir}),$$

where the token, t, is fixed and all layers, l, are considered. In essence, if the dot product between a particular direction `dir` probe with $\vec{R}_l(t)$ exceeds 0.5, then this should reflect the presence of a wall in the input maze at that particular probing location. For all experiments, we take the token, t, to be `<PATH_START>`, as it is the final token presented in each sequence at test time and will have seen all previous tokens.

We focus on `jirpy` as it was the most performant model (see the sweep results in Figure E.2), evaluating probes on 15,000 mazes. As shown in Figure 8.5, the `jirpy` model learns a highly linearly decodable representation of the maze at layer 2; consolidating the entire maze into a single token's latent representation (the `<PATH_START>` token). By looking at the variation in probing accuracy across layers and throughout training, we can understand how the formation of the world model occurs and potentially contributes to the model's performance (see Figure E.4 for results on all layers).

In Figure 8.5 we show the examples of mazes decoded at layer 2 with a set of probes that achieved the highest accuracy. Figure 8.6 shows that the maze representation was already learned by the second layer. In Table E.2, we show that across models capable of solving mazes at least $\approx 30\%$ of the time, linearly decodable representation tend to emerge within the first two layers. Based on the decrease in decodability after the second layer, we hypothesise that the model forms more abstract representations at deeper layers, causing the lower-level wall representations to be gradually overwritten. Lastly we



(a) Accuracy of all probes across positions for the most decodable layer (layer 2 for `jirpy`). We aggregate across directions (N,E,S,W) ignoring outer walls.

(b) Four random examples of mazes reconstructed using the probes. Reconstructions are made from a set of probes using a single latent embedding of the `<PATH_START>` token at layer 2. Wall colours indicate that thresholded probes Correctly Predicted, Omitted, or Added a wall.

Figure 8.5: Analysis of Linear Probes applied to the `<PATH_START>` token for `jirpy`.

attempted to steer the behaviour of `jirpy` by using the probes to guide the model's behaviour, as demonstrated by Nanda 2023. These experiments involved subtracting or adding the direction corresponding to a specific wall's probe in order to try and alter the path of the model. However, these experiments succeeded less than 1% of the time, even when accounting for effects such as non-centred representations (Jorgensen et al. 2023), and trying to enforce sparsity on the learned probes with an L1 regulariser (similar to chapter 5). These results suggest that the model may also utilise information at other tokens, or encode some information non-linearly, to facilitate its behaviour.

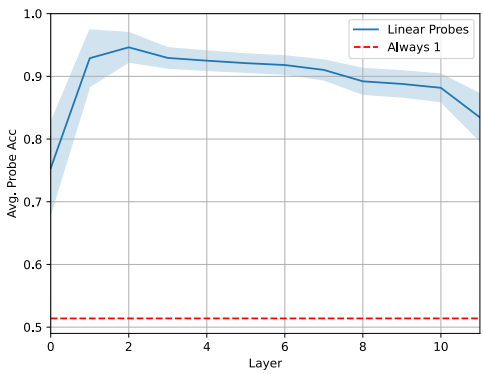
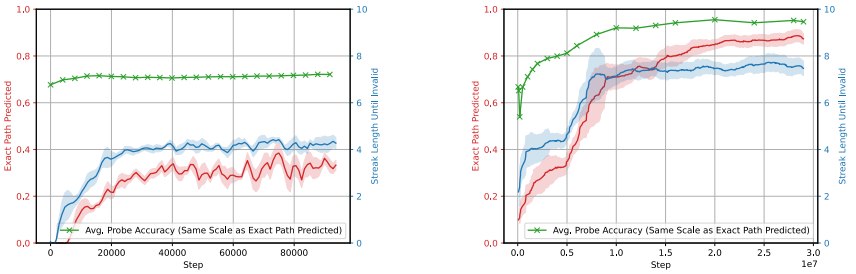


Figure 8.6: Per-layer accuracy of linear probes averaged across all positions and wall directions for `jirpy` on the `<PATH_START>` token. A distinct set of probes is trained at each layer. "Always 1" corresponds to always predicting the presence of a wall.



(a) `hallway` **(b)** `jirpy`

Figure 8.7: Layerwise analysis of maze structure captured by the model. Note that the distribution of paths for hallway mazes is shorter than those for forking mazes.

8.2.5 When Do Models Learn to Represent the Maze?

The phenomenon of grokking—where test accuracy suddenly improves during training—offers a unique window into representation learning dynamics Z. Liu et al. 2022; Nanda, L. Chan, et al. 2023. This abrupt generalisation has been linked to the emergence of structured internal representations Z. Liu et al. 2022. Having demonstrated that our models develop linearly decodable maze representations (subsection 8.2.4), we now investigate the relationship between representations and task performance. This analysis echoes our attempts to link structured representations to compositional generalisation in Part I.

To this end, we trained probe sets across checkpoints for both the `hallway` and `jirpy` models, with results shown in Figure 8.7. Firstly, we find that the `hallway` model does not learn a clear linear representation of the maze, while `jirpy` does. This should not surprise us, as we argued in the introduction to Part I that agents will acquire representations that are useful for the task at hand (those that delineate affordances).

More interestingly, we see that for jirpy, representations decodable beyond those found even in the `hallway` models arise in conjunction with the model learning to solve mazes more robustly—a somewhat more gradual analogue of grokking. These findings suggest that the representations found by our probes are at least partially aligned with causally relevant features, though our failure to intervene on them suggests that they are not the only features that the model uses to solve mazes. This may also explain why the decodability does not rise as steeply as the model's own performance.

8.2.6 Analysing Attention Heads

Our representation analyses revealed that models develop structured encodings of maze topology, but have not yet tied these to the models outputs.

To bridge this gap, we employ direct logit attribution[2] (DLA) (Lieberum et al. 2023; K. Wang et al. 2022). DLA quantifies each attention head's direct influence on output probabilities (logits) by measuring how much the head's output increases the probability of correct tokens relative to incorrect ones. For our maze-solving models, this reveals which heads actively participate in path-following decisions. We compute these contributions using the behavioural tasks introduced in subsection 8.2.1, focusing on scenarios that isolate path-following.

For an attention head h at layer l, we score its direct logit attribution as

$$C_{l,h} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\,\mathrm{LayerNorm}_{\mathrm{final}}(\vec{o}_{l,h}(\vec{x}))\cdot \vec{d}_y/\|\vec{d}_y\|\,\Big], \tag{8.1}$$

where the *correct direction* is $\vec{d}_y = \vec{e}_y - \bar{e}_{\neg y}$, $\vec{e}_y$ is the embedding of the correct next token, and $\bar{e}_{\neg y}$ is the *mean incorrect vector*, i.e., the average embedding of all tokens *other than* the correct one, computed once over the entire dataset of correct-incorrect pairs. Intuitively, $C_{l,h}$ measures how strongly the head's (layer-normalised) output aligns with the direction that moves logits toward the correct token and away from the average incorrect tokens. In practice, we compute estimations using 200 examples pairs of examples from an additional held-out dataset. We use the model's final layer normalisation on the head's outputs, as this is used for unembedding logits during a typical forward pass.

For simplicity's sake, we perform DLA across the heads of the much smaller `hallway` model. Our analysis reveals that that Head 3 in Layer 5 is highly specialized — thus dubbed the *Adjacency Head*. As shown in Figure 8.8 and Figure 8.9, this head consistently attends to tokens at path length 1 from the current position, effectively learning to respect the maze's connectivity structure. This differs from the embedding-based representations in subsection 8.2.2 which captured proximity of coordinate tokens, irrespective of the actual maze connectivity.

We also identify two other specialised heads with distinct roles. Layer 5, Head 0 tracks recent occurrences of the current coordinate token, whilst we posit that Layer 1, Head 2 functions as a reverse induction head—

[2] This analysis was carried out by M. Ivanitskiy, and is reproduced here with permission.

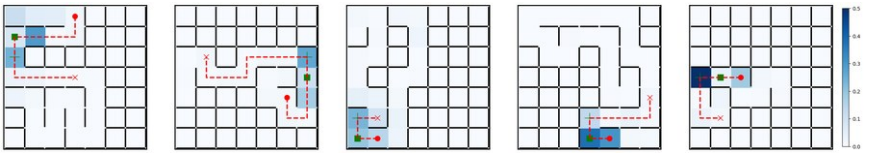


Figure 8.8: Attention of Layer 5, Head 3, on the `rand_path_token_nonend` task. Attention is displayed over the maze positions for five random held-out mazes. Blue shading is attention weight, true path is red dashed line from • to x, current position is ■.

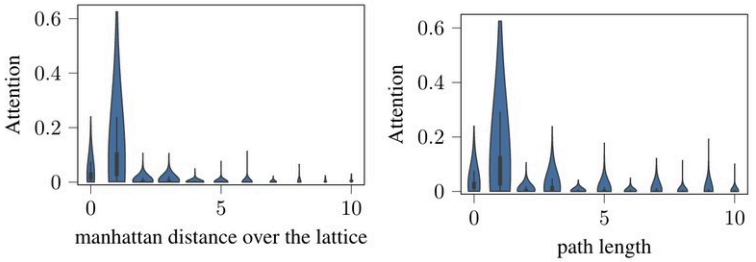


Figure 8.9: Attention of Layer 5, Head 3, on the `rand_path_token_nonend` task: violin plots of attention as a function of the distance between the current node and the node being attended to. x-axis on the left is the pure manhattan distance between the notes, while x-axis on the right is the path length between the nodes. Sample size $n = 200$. Note that while on the right, attention is overwhelmingly applied to nodes path length 1 away, some attention is applied to nodes at odd path lengths away because a node adjacent to the lattice will always be an odd path length away. Reproduced from Ivanitskiy, Spies, et al. 2024 with permission from M. Ivanitskiy.

consistently attending to the `<TARGET_END>` token. See section E.3 for more detailed analyses of these heads.

These findings suggest that maze-solving emerges through a division of labour among specialised components, with different heads tracking position history, target information, and local connectivity. We now have two parallel views of how models solve mazes: linearly decodable representations of maze structure that emerge during training, and specialised attention heads that implement navigation computation. In the next chapter we will attempt to achieve a deeper understanding of how such representations and attentional circuits interact.

8.3 Conclusion

This chapter demonstrates that transformers spontaneously develop structured spatial representations when trained on maze navigation. Our analyses reveal two parallel views of this capability: linearly decodable representations of maze structure that emerge during training, and specialised attention mechanisms, notably adjacency heads that respect maze topology. These findings show that interpretable structure can arise naturally through gradient descent, complementing engineered approaches.

The correlation between representation quality and task performance during training reinforces prior claims that structured representations enable robust reasoning. However, our analysis also highlights shortcomings of supervised interpretability methods. While linear probes successfully decode maze structure from residual streams, attempts to use these probes for interventions failed. This gap between decodability and causal influence suggests that the representations identified by supervised methods may not always capture the actual features the model uses for decision-making.

This limitation, combined with our aspirations to link the discovered representations to the specialised attention mechanisms, motivates exploring unsupervised approaches that can discover causally relevant features without imposing predefined ontologies. These ideas will form the basis of chapter 9.

CHAPTER 9

Causal World Models

The previous chapter established that maze-solving transformers develop linearly decodable spatial representations that emerge alongside task performance—but correlational evidence alone cannot tell us whether these representations actually drive behaviour. This chapter asks the causal question: are the emergent structured representations genuinely implicated in the model's decision-making, or are they merely epiphenomenal? Using sparse autoencoders to isolate individual connectivity features, we demonstrate that targeted interventions on these features predictably redirect path-finding behaviour, establishing them as causally efficacious components of a world model.[1]

The causally relevant representations we are after here are colloquially referred to as *world models*; before making our empirical case, we first clarify what we mean by the term. In lieu of an agreed upon definition of world models, we adopt that of (Millière et al. 2024), considering world models to be "structure-preserving [...] causally efficacious representations"—that is, representations that preserve the causal structure of an environment to the extent required by an agent's tasks. Such a notion of world-model captures both the agent-centric nature discussed in section 3.1, and the more grounded aspects that make such representations a faithful model of the world.

9.1 Methodology

9.1.1 Model Specifications

In the previous chapter we trained two models, varying in terms of size and the tasks they solved. In this chapter, we focus on slightly larger models which vary primarily in the way that they add position information to tokens when embedding them.

Recall from section 2.2.4 that position information here does not refer to position in the sense of maze-coordinates, but purely as a way of specifying the relative ordering of tokens in the sequence, regardless of their identity.

In particular, we compare learned position embeddings with Rotary Position Embeddings (RoPE) (Su et al. 2024). Learned position embeddings add trainable vectors to token embeddings based on absolute position (see subsection 2.2.4). In contrast, RoPE encodes position information by rotating query and key vectors in the complex plane, such that attention scores between token positions m and n depends only on their relative offset $(m - n)$, not the absolute positions. This property ensures that the

[1] This work was published as Spies, Edwards, et al. 2024.

Model Nickname	Maze Solving Accuracy	Positional Embeddings	d_{model}	n_{layers}	n_{heads}	Num. Params (M)
Stan	96.6%	**stan**dard learned	512	6	8	19.2
Terry	94.3%	ro**tary**	512	6	4	19.0

Table 9.1: Models chosen for mechanistic investigation (most performant in the sweep, given their respective position embedding schemes). Stan has more parameters as it learns position encodings ($W_{pos} \in \mathbb{R}^{512\times512}$), whereas Terry does not.

model learns relative position patterns (e.g., "two tokens apart") rather than memorising absolute positions, enhacing the generalisation performance to unseen sequence lengths.

This finding is validated by the sweeps performed in Figure F.1, which assess the generalisation performance of models with different position encoding schemes.

All models were trained for 6 epochs on 5×10^6 fully connected 5×5 mazes and sparse 6×6 mazes, embedded inside of larger 7×7 mazes. This not only yielded more performant models than those in chapter 8, but also allowed us to assess performance on unseen sequence lengths. On the basis of our sweep results, we select the best-performing models for each position encoding scheme as the targets of our analysis. These are shown in Table 9.1, and we dub these models Stan and Terry, as shorthands for the standard and rotary position encodings respectively.

9.1.2 Analysis Framework

Our methodology leverages Sparse Autoencoders (SAEs) to overcome the limitations of linear probes in detecting world model features. As discussed in section 7.2.2, SAEs enable us to identify interpretable features directly from activations without predefined ontological commitments, as well as allowing us to perform targeted interventions by manipulating specific features to observe causal impact on behaviour.

With an emphasis on understanding the causal construction and role of interpretable features, we will employ two complementary approaches:

- **Circuit-level analysis** (subsection 9.2.1): Direct examination of attention patterns and outputs.
- **Feature-level analysis** (subsection 9.2.2): Training SAEs on the residual stream to identify disentangled maze representations

Through systematic feature manipulation (section 9.3), we will establish that these representations are causally involved in maze-solving decisions,

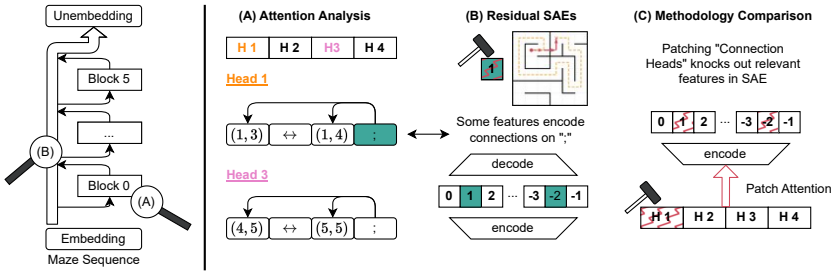


Figure 9.1: Overview of our methodology for discovering and validating world models in transformer-based maze solvers. **(A)** We analyse attention patterns in early layers, finding heads that consolidate maze connectivity information at semicolon tokens. **(B)** We train sparse autoencoders on the residual stream immediately following the first block, identifying interpretable features that encode maze connectivity. **(C)** We demonstrate the causal role of the world models in our transformers by comparing features extracted through both methods and validating them through causal interventions.

and can be understood as arising from identified attention circuits. This analysis pipeline is summarised in Figure 9.1.

9.2 Discovering World Model Representations

Before endeavouring to understand which features, if any, are implicated in the model's behaviour, we first need to identify them. To this end we will perform two open-ended explorations of the model's internals, and then use these to guide our investigation of the causal role of the world models.

In particular, we begin in subsection 9.2.1 by investigating attention heads in the earliest layer of our models and find heads specialising in the construction of representations akin to a world model. On the basis of this, we use SAEs alongside supervised classifiers reminiscent of chapter 4 to identify which features correspond to a world model. Finally, we use patching experiments and interventions to show that both investigations yield consistent features, and that these play a causal role in the model's behaviour, whilst also demonstrating some unexpected properties.

9.2.1 World Model Construction: Connectivity Attention Heads

Informed by the emergence of "adjacency heads" observed in chapter 8, we examined the attention patterns of Stan and Terry to understand how these models might construct internal representations of maze structure. Our analysis reveals that both models use specialised attention heads in their first layer to consolidate information about maze connections.

We began by visually inspecting the attention patterns of all heads in the first layer of our models across a small number of mazes. In so doing, we discovered that whilst not all heads showed clear patterns, a subset of those at layer 0 appeared to systematically process the adjacency list by focusing on connection information. These heads exhibit a consistent pattern at every 4th context position (corresponding to semicolon separation tokens) by attending either 1 or 3 tokens back, capturing one of the two coordinates in each maze connection. This pattern manifests in 3 out of 8 heads for Stan (Figure 9.2) and all 4 heads for Terry (Figure F.5).

To understand whether these heads are constructing meaningful world model representations, we analyse the outputs of these heads using their "OV-circuits" (Elhage, Nanda, Olsson, Henighan, Joseph, Mann, Askell, Yuntao Bai, et al. 2021) (see the details of attention in section 2.2.4). By examining the W_{OV} matrices per-head and computing cosine distances with coordinate token embeddings, we can characterise what information each head extracts and propagates forward.

This OV-circuit analysis reveals surprising patterns, particularly for Stan (Figure 9.3). Stan's three noteworthy heads (L0H3, L0H5, and L0H7) show clear complementary specialisation: L0H3's outputs activate strongly for even-parity maze cells, while L0H7 and L0H5 show complementary activation for odd-parity cells. This systematic partitioning suggests that Stan has developed a structured way to divide and process maze topology information across its attention heads, with each head responsible for specific subsets of the maze structure. For the other heads (those with less structured attention patterns) we observed that these were also not particularly sensitive to positional information (see Figure F.12).

Terry's OV-circuit patterns are less pronounced but still show some structure (Figure 9.4). While Terry lacks Stan's clear global even/odd specialisation, its heads show localised specialisation patterns in specific maze regions; this difference is most likely a result of the different positional encoding schemes used by the two models; Stan's need to compute relative

offsets to form meaningful representations of connections might favour learning sharply varying position encodings.

The prevalence and systematic behaviour of these connectivity attention heads strongly implicate them in world model construction.

By consolidating connection information at semicolon positions in both models, these heads appear to be constructing the lowest levels of meaningful representations—though with notably different organisational strategies between Stan and Terry.

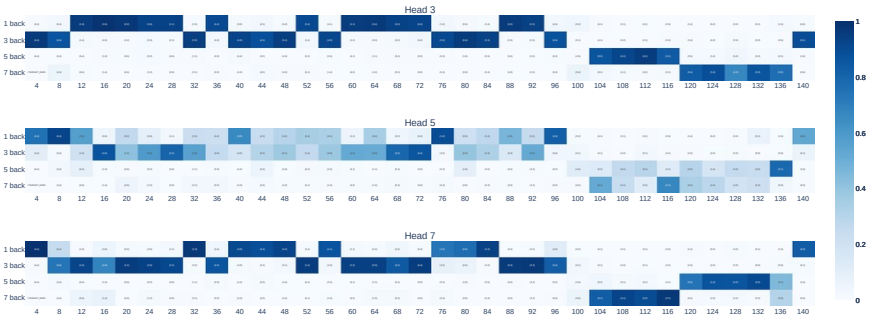


Figure 9.2: Attention patterns for Stan's "connectivity heads" (L0H3, L0H5, L0H7) showing where semicolon tokens attend to in the sequence. The x-axis shows semicolon positions; the y-axis indicates lookback distance (1 = previous token, 3 = three tokens back). Color intensity represents attention strength. Note the shift from attending 1-3 tokens back to 5-7 tokens back after position 100. Complete attention patterns for both Terry and Stan are shown in section F.2.

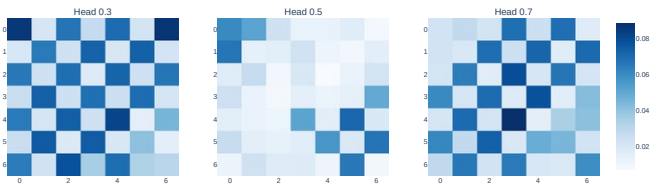


Figure 9.3: Magnitude heatmaps of Stan's L0 head outputs (W_{OV} applied to token embeddings) projected onto the maze grid. Heads show clear specialisation: L0H3 focuses on even-parity cells (lighter pattern), while L0H7 and L0H5 focus on odd-parity cells.

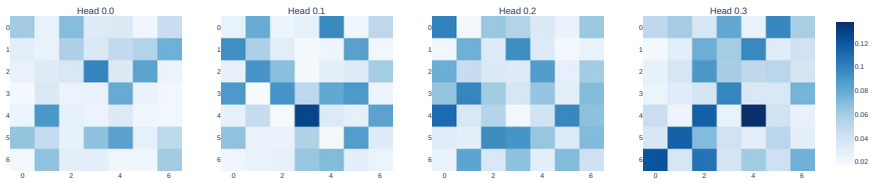


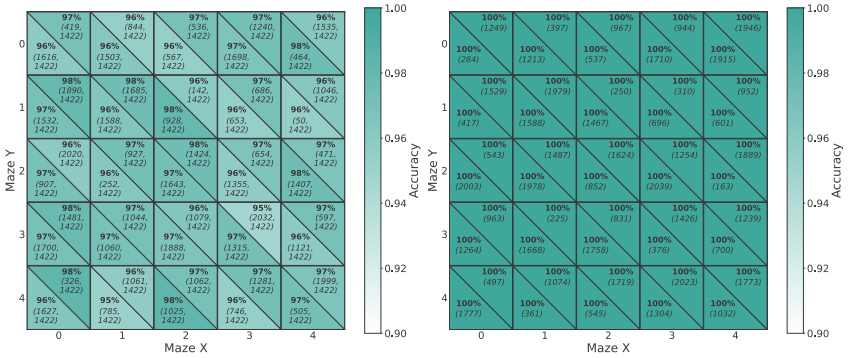
Figure 9.4: Magnitudes of vectors resulting from applying the W_{OV} matrices of layer-0 heads of Terry to maze-cell token embeddings, projected onto the maze grid. The pattern here is much less striking than that for Stan (shown in Figure 9.3) although it does suggest that the heads specialise in even/odd-parity cells in localised regions of the maze.

9.2.2 World Model Representation: Sparse Autoencoders

Having identified connectivity attention heads that appear to construct world model representations, we now turn to understanding the features in our model, with the goal of linking both investigations. To this end, we train SAEs on the residual stream immediately after the first layer (where we found connectivity heads aggregating information at semicolon tokens) of both models. The trained SAEs faithfully reconstruct activations at *all* token positions, and we find that replacing model activations with their SAE reconstructions does not affect maze-solving performance, confirming they capture all behaviourally relevant information. Full training and sweeping details are provided in section F.1.

To identify which SAE features encode maze connectivity, we initially attempted to compare feature activations between mazes with and without specific connections. However, this approach proved unreliable: feature magnitudes varied inconsistently, and multiple features often co-activated for a single connection—particularly those involved in path representations that changed when connections were modified. We therefore adopted a more systematic approach, training decision trees to predict connection presence from SAE feature activations at the relevant semicolon positions (akin to our methods for identifying salient latents in chapter 4). In Figure 9.5 we show the decision tree decoding accuracies and relevant features (in parentheses) for each connection in the maze, demonstrating that these features provide a near perfect prediction of the presence of a given connection.

This analysis revealed a notable difference between our two models. Stan employs a compositional encoding scheme in which each connection is represented by two features—a generic "semicolon" feature that activates



(a) Stan **(b)** Terry

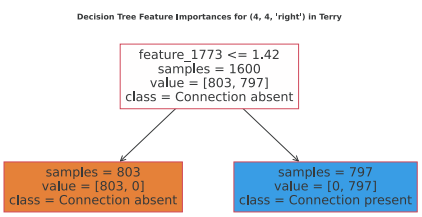
Figure 9.5: Decision tree decoding accuracies and relevant features (in parentheses) for each connection in the maze. Upper right triangles correspond to right connections, and lower left triangles correspond to down connections. The decision trees were trained to predict the presence, or absence, of a connection from the SAE feature vector at the semicolon immediately following the definition of that connection. See Figure F.6 for more details.

for all semicolons, paired with a connection-specific feature (Figure 9.6b). In contrast, Terry encodes each connection cleanly in a single dedicated feature (Figure 9.6a). Highly activating examples of these features are visualised in Figure F.6, with Stan's representation remaining stable across multiple independently trained SAEs (Figure F.8).

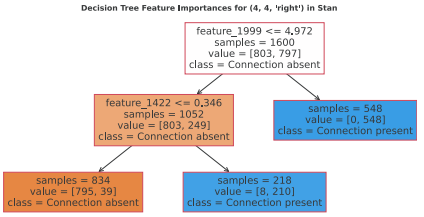
We hypothesise that Stan's compositional code arises from an imperfect separation of learned positional and semantic information. This entanglement of information was suggested in chapter 8 as an explanation for the failure to intervene on similar models with linear probes—now we see that intervening on a single direction would inadvertently affect the shared semicolon feature. Terry's cleaner single-feature encoding, despite its more entangled attention patterns, likely benefits from RoPE's explicit separation of positional information from token content.

9.2.3 Comparing SAEs and Circuits

In subsection 9.2.1 we advanced the claim that certain L0 heads construct features representing maze edges at the `;` context positions, specifically by attending to earlier positions containing maze-cell token embeddings, and rewriting those embeddings by application of their W_{OV} matrices.



(a) Terry model: A single feature almost perfectly encodes the existence of a specific connection in the maze. This demonstrates the direct encoding of maze connectivity in Terry's SAE latent space.



(b) Stan model: Two features (Feature 1422 and another) work together to encode maze connectivity. Feature 1422 appears consistently across all connections, aligning with the decision tree decoding results presented Figure 9.5.

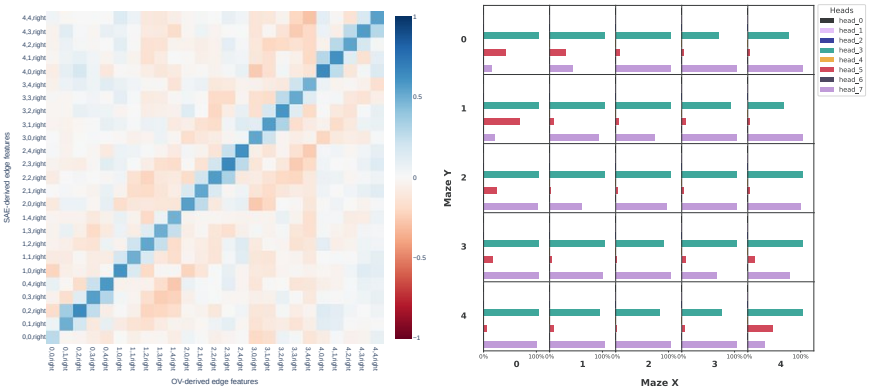
Figure 9.6: Decision trees trained on SAE latents for Terry and Stan models, predicting the existence of specific connections in the maze. These examples illustrate how maze connectivity is encoded in the residual stream at layer 0 on the corresponding semicolon position. The decision trees were trained as supervised classifiers whose target was to predict the presence of a given connection, given an SAE feature vector from the corresponding semicolon position. These SAEs were trained with 10,000 examples per connection (equally balanced between the presence / non-presence of a connection).

subsection 9.2.2 identified features representing maze connections via an independent line of reasoning, by training SAEs, and identifying which of their features were indicative of the presence of a connection.

To verify whether these approaches yielded consistent features for the world model, we first calculated the cosine similarity between the features written by isolated attention heads, and those encoded in the SAE (Figure 9.7a). These showed excellent agreement for Stan, where the attention patterns were clear, but only once the compositional code was taken into account (see section F.6 for details).

Though these results were promising, we carried out a further comparison (Figure 9.7b) to minimise the assumptions required, and to account for two potential effects: 1) There may be "wiggle room" between feature directions in the model's residual stream, and the circuits that construct them (which would lead to low cosine similarities, even for the same features), 2) As our SAEs are trained after an entire block of computation, it is possible that the MLPs, applied after attention, also played a role in forming the representations. In this second experiment we patched attention head values in the presence of a connection to the mean of a maze set without that connection (reminiscent of the Direct Logit Attribution described in

chapter 8). By looking at the effect of patching the attention heads on the resulting SAE latent vectors, we were able to observe that the features considered relevant for any given connection were indeed sensitive to the heads implicated in constructing those features.



(a) Cosine similarities between edge features derived from the SAE and edge features derived by direct application of the W_{OV} matrices of the heads discussed in subsection 9.2.1 to token embeddings for Stan (for reasons of space, only the right directed edge features are shown here, but the pattern is the same for the full set, shown in Figure F.13). Details of how features are computed are given in section F.6. This figure was produced in collaboration with W. Edwards.

(b) Effect of patching attention heads on SAE features (rightward connections) for Stan's connection-specific WM features (identified from Figure 9.5). Compared against Figure 9.3, we see agreement between the attention analysis and the SAE Feature analysis. We normalise the per-head patching effect magnitudes, such that 100% was the maximal effect seen on an SAE feature's magnitude as a result of patching the attention head (across all features for a given head).

Figure 9.7: Comparison of SAE features and attention head analysis.

In particular, we consider the effect on the SAE features identified in subsection 9.2.2 when each attention head is patched at the semicolon position with its average non-connection value across 500 examples (i.e. removing the contribution a given head toward encoding that connection). This captures the extent to which a given head contributes towards the "creation" of a maze-connection's representation in the residual stream. The results show a strong agreement — when we ablate a connection-specific head, we see that the feature for that connection is ablated too.

These plots not only confirm the link between the attention circuits and the SAE features, but even show the same spatial partitioning of different parts of the maze across heads. The same plots for Terry, and Stan's down connection, are shown in section F.5. This agreement between unsupervised methods begins to tell a more complete story of how the parameters,

embeddings and latent features interact to form the world model. This brings us to the next section, where we will use interventions to show that the features we have come to understand are indeed causally implicated in the model's behaviour.

9.3 Intervening on World Models

Having identified structured representations through both circuit analysis and SAE decomposition, we now turn to the critical question: are these representations causally implicated in the model's decision-making; are they "causally efficacious"? In Figure 9.8, we give an example of perturbing a feature to "fool" the model into behaving as though it is in a different maze.

When patching in the SAE-reconstructed residual stream without perturbations we still see the same behaviour as in the original model; when patching in with a modified feature, we see a change in the path. These successful interventions provide the strongest evidence yet that our models develop *causal world models*, not merely correlational representations.

We perform our interventions across 200 examples for each connection feature, and show the resulting intervention efficacies in Figure 9.9. The intervention process involves toggling a feature on (to the maximal value observed for that feature in a small dataset) or turning it off (setting it to 0) *at all semicolon positions*—for the case of adding a connection, this is necessary as there is no semicolon in the sequence which "belongs" to the connection that doesn't exist[2].

We measure the impact of these interventions on the model's maze-solving accuracy, with a particular focus on how activating versus removing features affects performance. Our results reveal an intriguing asymmetry that constitutes our second finding: interventions that activate features tend to be more effective in altering the model's behaviour compared to those that remove features.

This asymmetry suggests that transformer world models may be more robust to feature suppression than enhancement, with implications for both understanding model behaviour and developing intervention strategies.

[2] We also experimented with toggling to a fixed maximal value in Figure F.9, but this was generally less effective. In the case of removal, it made little difference if the feature was disabled everywhere, as it is almost always exactly 0 for a non-matching connection semicolon.

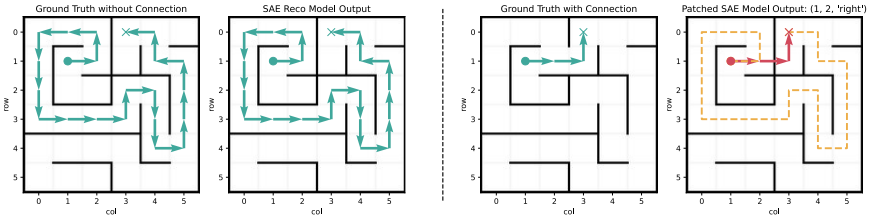
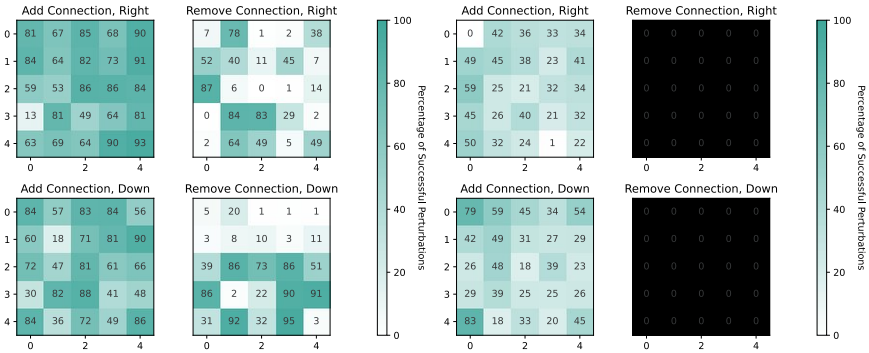


Figure 9.8: An example of an intervention on Terry where a connection is added by enabling the relevant feature in the SAE's latent space (in this case, feature 250 for `(1,2)` `<->` `(1,3)`). From left to right: 1) input maze with ground truth 2) model's prediction with the unperturbed SAE reconstruction patched in 3) perturbed ground truth 4) model's prediction with the perturbed SAE reconstruction patched into its residual stream after the first layer.



(a) Terry model **(b)** Stan model

Figure 9.9: Aggregated accuracy of interventions for examples on which the original prediction was correct. An accurate intervention is one in which the toggling of a connection feature in the SAE leads the model to act accordingly. Note that Stan removal interventions fail as the inputs in these cases have more connections than the model is able to handle (see length generalisation failure in Figure F.1).

Our final finding relates to the toggling of features in Stan. Though Stan utilised a compositional code, activating the connection-specific features at unrelated semicolons worked in 35% of cases. Conversely, we saw that all removal interventions failed, for the simple reason that Stan was unable to generalise to sequences containing more connections than it had seen during training — thus failing when shown examples containing the additional connection to be removed (a result of Stan using learned positional embeddings, as shown in Figure F.1). This failure connects to broader challenges in transformer generalisation and brittleness seen in chapter 6.

The fact that activating connections in the space of the SAE worked at all means that Stan's maze-solving behaviour was at least partially able to generalise in the latent space, where it was decoupled from the positional embeddings. This, albeit simple, demonstration shows how models can perform behaviours that they cannot achieve through their standard input interface. It is possible that similar techniques, applied at higher levels, could allow us to exploit other capabilities of existing models to greater extent.

9.4 Related Work

World Models in Game-Playing Systems We demonstrated that transformers develop causal world models in maze-solving tasks and validated these through SAE-based interventions. Works highlighted in the previous chapter often relied on linear probes to identify spatial representations (Jenner et al. 2024; McGrath et al. 2022). While Zhengfu He et al. 2024 attempted to use SAEs to re-identify Othello world models, they encountered difficulties with feature identification. We went further by painting a complete picture of how these features arise—from attention-based construction to SAE isolation to causal validation through intervention—demonstrating that maze world models are not only identifiable but causally efficacious.

Mechanistic Interpretability and Causal Interventions Our work establishes genuinely causal world models in maze-solving transformers by using SAE features as the units of causal intervention. This contrasts with previous maze-solving analyses that achieved only linear decodability or identified attention motifs: Ivanitskiy, Spies, et al. 2024 found no isolatable causal features despite mechanistic analysis, while Mini et al. 2023 manipulated representations in RL contexts using different methods. By combining attention pattern analysis with SAE feature isolation and systematic intervention, we move beyond correlational decoding to demonstrate causal efficacy. Our finding that interventions can push models beyond their training distribution—succeeding where equivalent token inputs would fail—reveals a flexibility in internal world models not previously documented.

While recent work has explored world models in large language models trained on diverse datasets (Belrose et al. 2023; Lieberum et al. 2023; Olsson, Elhage, Nanda, et al. 2022a), we maintain our focus on the controlled environment of maze-solving to enable more rigorous causal analysis.

9.5 Conclusions

This chapter completes our mechanistic interpretability investigation of transformer world models, establishing not merely their presence (Chapter 1) but their causal role in model behaviour. Through our analysis of attention patterns and SAE-based interventions, we have moved from correlation to causation—demonstrating that the structured representations in maze-solving transformers are genuine, causally efficacious world models.

9.5.1 Key Contributions

Our investigation yielded two primary contributions:

Empirical Findings: We demonstrated that transformers acquire causally relevant world models by showing that SAE feature interventions predictably alter maze-solving behaviour. Most surprisingly, we discovered a consistent asymmetry—activating connection features proved far more effective than suppressing them, suggesting transformers rely more heavily on positive evidence in their computations. Additionally, we found that SAE interventions can unlock capabilities beyond the model's standard interface: models that failed on longer sequences due to positional embedding limitations could successfully navigate when appropriate features were activated directly in latent space.

Methodological Insights: We showed that different architectural choices lead to fundamentally different world model encodings. Models with learned positional embeddings developed compositional codes requiring multiple SAE features per connection, while those with rotary embeddings encoded connections more directly. This architectural dependence has profound implications for interpretability—techniques successful on one model may fail on another. Our decision tree approach for analysing SAE features provides a systematic method for identifying these varying representational strategies.

We have only scratched the surface of the potential of mechanistic interpretability. There are many exciting directions for future work, which we will reflect upon in the following chapter.

Closing Remarks

Chapter 10

Reflections and Future Directions

This work has explored interpretability in both paradigms: engineering structure into neural architectures (Part I) and reverse-engineering representations that emerge through scale (Part II). Our journey yielded neither a victory for structure nor a vindication of scale, but rather shone a light on the trade-offs that arise when seeking interpretable systems. What follows is a detailed account of our contributions, their limitations, and the road ahead.

Nearly five years ago, we set out to answer a seemingly simple question: how can we design neural networks which are both more interpretable and more robust? The promise of interpretable neural networks seemed within reach through careful architectural design, yet the emerging power of scale-driven approaches was already reshaping the landscape. We positioned ourselves at the crossroads between these two paradigms: engineering structure for interpretability versus embracing scale for capability. Now, as we write these final words, the field has largely chosen its path—with foundation models racing down the highway of scale while our understanding struggles to keep pace.

10.1 Summary of Contributions

10.1.1 Engineering Structured Representations

Our first contribution lies in demonstrating how spatial transformers can enhance slot-based architectures to achieve more interpretable object-centric representations. By augmenting Slot Attention with learnable geometric transformations, we showed that models could not only segment scenes into objects but also capture their pose in an explicit, interpretable form. Investigating this latent structure was a challenge, requiring us to leverage the internal masks from SA to determine the identity of slots, before training auxiliary models to inspect their representations.

Building on this foundation, we explored how structured perception could enable systematic reasoning.

By adapting the PrediNet's relational architecture to operate on slot-based representations, we created a sparse, hierarchical system where structured representations were leveraged for compositional reasoning. Our experiments on both the RelationsGame and Sort-of-CLEVR tasks revealed that sparse connectivity at the reasoning layer was more beneficial than sparsity in object-property bindings. This finding suggested that disentan-

glement aided generalisation most at higher levels of the representational hierarchy — an idea echoed later in our analysis of transformer representations.

However, our investigations also uncovered the dangers of over-specification. On tasks requiring perfect object-level knowledge of a scene, our highly structured models underperformed their unstructured counterparts by significant margins. When we constrain too tightly, we risk sacrificing the very flexibility that makes neural networks powerful. This trade-off crystallised a key lesson: inductive biases are double-edged swords, and their utility depends critically on alignment with task structure; particularly when information bottlenecks in structured architectures can lead to cascading errors that compound throughout a model.

10.1.2 Understanding Emergent Representations

The interlude between our two main investigations revealed a key issue at the heart of modern AI: while foundation models demonstrate remarkable reasoning capabilities across diverse domains despite their relatively simple structure, they exhibit great fragility. Even minor perturbations to visual input format, task specification, or textual prompts can cause catastrophic performance degradation in systems that otherwise appear robust. This unpredictable generalisation behaviour in models that are now being used by hundreds of millions of people every day, motivated our shift from engineering interpretability to understanding how structure emerges naturally and whether such emergent representations might offer clues to building more robust systems.

To this end we trained and investigated transformers in a simple domain which required planning in its simplest form — solving mazes. Our mechanistic analysis revealed a wonderful phenomenon: structure emerges even without explicit architectural pressure. Using simple linear probes, we discovered that these models develop rich internal representations of maze topology, which arise rapidly as the models learn to solve mazes. Downstream of these, specific attention heads emerged as “adjacency detectors," respecting the connective structure of mazes with near-perfect fidelity.

While these investigations revealed tantalising correlations, they did not establish causal relationships between representations and behaviour. To move beyond mere observation, we employed Sparse Autoencoders (SAEs), which leverage unsupervised learning to avoid imposing our expectations about what features the model should learn. This approach discovered the

models' internal "world models" — coherent representations of the maze environment that we could causally intervene upon to alter behaviour.

10.2 Limitations

Our structured approaches in Part I, while achieving interpretable object-centric representations, encountered limitations that reveal the challenges of imposing architectural constraints. The slot-based architectures we developed in chapter 4 and chapter 5 required pre-specifying the number of object slots, often failing to capture objects or splitting them into multiple slots. As such, while our model achieved strong disentanglement metrics on toy datasets, they would be unlikely to generalise beyond these controlled environments.

Subsequent work has addressed many of these constraints through dynamic slot allocation, learnable object queries, and hierarchical decomposition (Elsayed et al. 2022; Engelcke, Parker Jones, et al. 2021; Fan et al. 2024), while scalable detection architectures demonstrate that ANNs can handle real-world scenes — though rarely interpretably.

Our findings in chapter 5 revealed that the same inductive biases that promote interpretability can hamper model flexibility. When we forced our models to factor representations into discrete objects and explicit relations, they underperformed generic architectures by significant margins on tasks that required complete scene understanding or violated our structural assumptions. Recalling the flexibility–scale spectrum of chapter 1, transformers have since emerged as a more flexible middle ground — implementing implicit relational reasoning through attention without hard-coded object structure (Vaswani et al. 2017) — with slot- and sparsity-based variants (Faulkner et al. 2022; Shazeer et al. 2017) offering ways to reincorporate these priors when needed.

Yet while these architectural innovations have made models more capable due to scale, they have done less to make them robust and interpretable. The same models that excel at complex reasoning tasks can fail catastrophically when faced with seemingly minor perturbations — a limitation that architectural improvements alone are unlikely to address. This brittleness motivated our shift in Part II: if we cannot engineer robustness through architecture, perhaps we can understand and control it through mechanistic analysis.

Our mechanistic interpretability investigations in chapter 8 and chapter 9, while revealing rich internal structure, faced severe practical con-

straints. The computational cost of training and analysing SAEs scales poorly with model size, making these techniques challenging to apply beyond small models in most academic contexts. Our SAEs required hundreds of GPU-hours to train, and analysing the thousands of resulting features is a daunting task. More limiting still was our choice of maze-solving as a test domain, which while enabling precise causal interventions and a known ground truth, raised pertinent questions about generalisability. Can insights from path-finding in discrete grids really inform our understanding of language models processing natural text or multimodal systems reasoning about the physical world?

Efficient SAE variants using gated architectures and top-k sparsity have since reduced these computational costs, and automated circuit discovery tools (Conmy et al. 2023) extend similar analyses to billion-parameter models. The extent to which our findings generalise beyond maze-solving is bolstered by convergent results elsewhere — the linear representations we identified echo those found in language models (Mikolov et al. 2013), and our adjacency heads parallel induction heads in GPT-style models (Olsson, Elhage, Nanda, et al. 2022b) — although the reality at scale remains considerably more complex (Lieberum et al. 2023).

In-aggregate, the field has made remarkable progress across both architectural design and interpretability methods in a remarkably short time. Yet despite this progress, fundamental challenges persist: we still lack principled frameworks for ensuring robustness, preventing deceptive alignment, or guaranteeing that our interpretability tools capture all safety-relevant behaviours. These open problems frame the concrete research directions we turn to next.

10.3 Future Directions

The remaining technical challenges are formidable, and our own work exemplified them: even with explicit architectural constraints in Part I, we did not disentangle all generative factors, and downstream reasoning with crisper representations was far from perfect; even with successful causal interventions in Part II, many discovered features remain uninterpreted. The directions below aim squarely at such gaps.

Advancing Object-Centric Architectures. The structured representations explored in Part I suggest several immediate research directions. The exclusivity problem — where multiple slots attend to the same object or a single object is split across slots — could be addressed through contrastive objectives (Löwe et al. 2020) or implicit gradient methods (Chang

et al. 2023), both directly applicable to our SA+ST framework. Temporal alignment, which our recurrent extension handled only coarsely, would benefit from integration with dedicated alignment methods such as AlignNet (Creswell, Nikiforou, et al. 2020) or the dynamics-aware SlotFormer (Z. Wu et al. 2023). Background handling through privileged slots (Qi et al. 2024) and explicit mask regularisation (Burgess, Matthey, et al. 2019) would address a persistent limitation of slot-based architectures. Taken together, these directions target the specific failure modes exposed in Part I, and would strengthen the disentanglement properties on which any downstream use of slot-based representations ultimately depends.

Scaling Mechanistic Analysis. The causal world models uncovered in Part II raise two complementary questions. The first concerns how such world models are *used*. Our interventions established that SAE features encoding maze connectivity causally shape single-step behaviour. Whether these representations also underpin multi-step planning remains open, as does whether the intervention asymmetry we observed — where enabling features was more effective than suppressing them — reflects a general principle of how transformers accumulate evidence.

The second question concerns how world models *form and remain intervenable across domains*. Convergent results in Othello-GPT (K. Li et al. 2022; Nanda 2023) and chess-playing transformers (Karvonen 2024) suggest linear world representations are a general phenomenon in sequence models trained on structured tasks. Yet systematic comparison of how these representations emerge during training, what features they expose, and how robustly they can be manipulated has yet to be carried out. Extending our probing and SAE analyses across these planning-oriented toy domains—and ultimately to natural language and multimodal reasoning—would test whether mechanistic understanding can scale alongside capability. The parallels between our adjacency-detecting heads and induction heads in language models (Olsson, Elhage, Nanda, et al. 2022b) offer a concrete starting point.

Bridging Perception, Reasoning, and Interpretability. The most promising future directions lie at the intersection of this thesis's two parts. The Interlude (chapter 6) revealed a substantial gap between language-mediated reasoning and visually grounded perception in foundation models—a gap that may be more efficiently closed with inductive bias than with scale. Bridging this divide may require combining the explicit object-centric representations of Part I with the mechanistic understanding developed in Part II: building models whose internal representations are

structured enough to be steerable, yet flexible enough to handle real-world complexity. A concrete convergence point is sparsity: mixture-of-experts architectures simultaneously improve scalability and interpretability (X. Yang et al. 2025), suggesting that the apparent trade-off between structure and flexibility may be less fundamental than it appears.

Concluding Remarks

Whilst the interpretability we've sought is essential, it is also but one facet of AI alignment. This broader challenge extends beyond building systems that follow our instructions correctly to ensuring those instructions themselves promote human flourishing. This demands grappling with questions of values, governance, and long-term societal impact. Today's economic incentives overwhelmingly favour capability over comprehension, automation over augmentation—a trajectory that should give us pause. Building a beneficial future with AI requires diverse expertise: policymakers versed in both opportunities and risks, researchers who refuse to sacrifice understanding for capability, and institutions designed to reward comprehension as much as advancement.

Within that broader project, the contributions of this thesis represent small steps on a longer journey—from toy problems to the real world, from small ANNs to vast LLMs, and from academic insights to aligned AI. Yet they demonstrate that progress is possible on multiple fronts simultaneously. We need not choose wholly between capability and comprehension, between engineering and analysis. Indeed, the universe has gifted us minds that arise from simple rules, both artificial and biological, and understanding these minds is not merely an academic luxury but an existential necessity for navigating the future we are rapidly creating.

And in pursuing this understanding, we may discover something profound. Writing of artificial neural networks decades before they showed their true promise, Rosenblatt 1958 mused that through their study "those fundamental laws of organisation which are common to all information handling systems, machines and men included, may eventually be understood." The ancient mysteries—*nous*, *psyche*, *logos*—may finally yield their secrets in silicon rather than flesh. In seeking to comprehend artificial minds, we may finally understand our own.

We may well be the first generation capable of building minds and the last with a chance to align them with human values, whatever those may be. History has given us this moment; let us give history an answer.

Bibliography

Achiam, Josh et al. (Mar. 2023). *GPT-4 Technical Report*. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs.CL]. URL: http://arxiv.org/abs/2303.08774 (visited on 12/21/2024).

Alain, Guillaume and Yoshua Bengio (Nov. 2018). *Understanding Intermediate Layers Using Linear Classifier Probes*. DOI: 10.48550/arXiv.1610.01644. URL: http://arxiv.org/abs/1610.01644 (visited on 08/04/2025).

Alayrac, Jean-Baptiste et al. (Nov. 2022). *Flamingo: A Visual Language Model for Few-Shot Learning*. DOI: 10.48550/arXiv.2204.14198. URL: http://arxiv.org/abs/2204.14198 (visited on 08/04/2025).

Albinhassan, Mohammad, Pranava Madhyastha, and Alessandra Russo (Mar. 2025). *SEM-CTRL: Semantically Controlled Decoding*. DOI: 10.48550/arXiv.2503.01804. URL: http://arxiv.org/abs/2503.01804 (visited on 07/09/2025).

Alrajeh, Dalal and Alessandra Russo (2018). "Logic-Based Learning: Theory and Application". In: *Machine Learning for Dynamic Software Analysis*. Ed. by Amel Bennaceur, Reiner Hähnle, and Karl Meinke. Cham: Springer, pp. 219–256. ISBN: 978-3-319-96562-8. DOI: 10.1007/978-3-319-96562-8_9. URL: https://doi.org/10.1007/978-3-319-96562-8_9.

Alviano, Mario, Ly Ly Trieu, Tran Cao Son, and Marcello Balduccini (Sept. 2023). "Explanations for Answer Set Programming". In: *Electronic Proceedings in Theoretical Computer Science* 385, pp. 27–40. ISSN: 2075-2180. DOI: 10.4204/EPTCS.385.4. URL: http://arxiv.org/abs/2308.15879 (visited on 07/05/2025).

Amazon AGI (Dec. 2024). *Introducing Amazon Nova, Our New Generation of Foundation Models*. Amazon announcement post. URL: https://www.aboutamazon.com/news/aws/amazon-nova-artificial-intelligence-bedrock-aws (visited on 11/01/2025).

Amodei, Dario (2025). *The Urgency of Interpretability*. URL: https://www.darioamodei.com/post/the-urgency-of-interpretability (visited on 07/03/2025).

Amodei, Dario et al. (2018). *AI and Compute*. OpenAI Blog. URL: https://openai.com/index/ai-and-compute/ (visited on 05/16/2018).

Andreas, Jacob, Marcus Rohrbach, Trevor Darrell, and Dan Klein (July 2017). *Neural Module Networks*. DOI: 10.48550/arXiv.1511.02799. URL: http://arxiv.org/abs/1511.02799 (visited on 07/19/2025).

Anthropic (June 2024a). *Introducing Claude 3.5 Sonnet*. Anthropic announcement post. URL: https://www.anthropic.com/news/claude-3-5-sonnet (visited on 11/01/2025).

– (Oct. 2024b). *Introducing Computer Use, a New Claude 3.5 Sonnet, and Claude 3.5 Haiku*. Anthropic announcement post. URL: https://www.anthropic.com/news/3-5-models-and-computer-use (visited on 11/01/2025).

– (Mar. 2024c). *The Claude 3 Model Family: Opus, Sonnet, Haiku*. Anthropic Technical Report. URL: https://www.anthropic.com/news/claude-3-family (visited on 12/21/2024).

– (May 2025). *Introducing Claude 4*. Anthropic announcement post. URL: https://www.anthropic.com/news/claude-4 (visited on 11/01/2025).

Aspis, Yaniv, Mohammad Albinhassan, Jorge Lobo, and Alessandra Russo (2024). "Embed2Rule Scalable Neuro-Symbolic Learning via Latent Space Weak-Labelling". In: *Neural-Symbolic Learning and Reasoning*. Ed. by Tarek R. Besold et al. Cham: Springer Nature Switzerland, pp. 195–218. ISBN: 978-3-031-71167-1. DOI: 10.1007/978-3-031-71167-1_11.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (July 2016). *Layer Normalization*. DOI: 10.48550/arXiv.1607.06450. URL: http://arxiv.org/abs/1607.06450 (visited on 07/03/2025).

Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli (2020). "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in Neural Information Processing Systems*.

Bai, Shuai et al. (Feb. 2025). *Qwen2.5-VL Technical Report*. DOI: 10.48550/arXiv.2502.13923. URL: http://arxiv.org/abs/2502.13923 (visited on 11/01/2025).

Balduzzi, David et al. (July 2017). "The Shattered Gradients Problem: If Resnets Are the Answer, Then What Is the Question?" In: *ICML*. PMLR, pp. 342–350. URL: https://proceedings.mlr.press/v70/balduzzi17b.html (visited on 07/09/2025).

Bartoldson, Brian, Ari S. Morcos, Adrian Barbu, and Gordon Erlebacher (2020). "The Generalization-Stability Tradeoff in Neural Network Pruning". In: *NeurIPS*. arXiv: 1906.03728.

Battaglia, Peter W. et al. (Oct. 2018). "Relational Inductive Biases, Deep Learning, and Graph Networks". In: *arXiv:1806.01261 [cs, stat]*. URL: http://arxiv.org/abs/1806.01261 (visited on 07/14/2021).

Belinkov, Yonatan (Mar. 2022). "Probing Classifiers: Promises, Shortcomings, and Advances". In: *Computational Linguistics* 48.1, pp. 207–219. DOI: 10.1162/coli_a_00422. URL: https://aclanthology.org/2022.cl-1.7/.

Belkin, Mikhail, Daniel Hsu, Siyuan Ma, and Soumik Mandal (2019). "Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off". In: *PNAS*. Early version arXiv:1812.11118.

Belrose, Nora et al. (2023). "Eliciting latent predictions from transformers with the tuned lens". In: arXiv: 2303.08112 [cs.LG]. URL: https://arxiv.org/abs/2303.08112.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (Apr. 2014). "Representation Learning: A Review and New Perspectives". In: *arXiv:1206.5538 [cs]*. URL: http://arxiv.org/abs/1206.5538 (visited on 07/13/2021).

Bengio, Yoshua, Geoffrey Hinton, et al. (May 2024). "Managing Extreme AI Risks amid Rapid Progress". In: *Science* 384.6698, pp. 842–845. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.adn0117. URL: http://arxiv.org/abs/2310.17688 (visited on 07/04/2025).

beren and Sid Black (Nov. 2022). "The Singular Value Decompositions of Transformer Weight Matrices Are Highly Interpretable". In: URL: https://www.alignmentforum.org/posts/mkbGjzxD8d8XqKHzA/the-singular-value-decompositions-of-transformer-weight (visited on 07/27/2025).

Bereska, Leonard and Efstratios Gavves (2024). "Mechanistic Interpretability for AI Safety: A Review". In: *CoRR* abs/2404.14082. URL: https://arxiv.org/abs/2404.14082.

Bishop, Christopher M. (2007). *Pattern Recognition and Machine Learning*. 1st. Springer. ISBN: 0387310738.

Blondel, Mathieu and Vincent Roulet (June 2025). *The Elements of Differentiable Programming*. DOI: 10.48550/arXiv.2403.14606. URL: http://arxiv.org/abs/2403.14606 (visited on 07/05/2025).

Bommasani, Rishi et al. (July 2022). *On the Opportunities and Risks of Foundation Models*. DOI: 10.48550/arXiv.2108.07258. URL: http://arxiv.org/abs/2108.07258 (visited on 07/07/2025).

Bostrom, Nick (2014). *Superintelligence: Paths, Dangers, Strategies*. 1st. USA: Oxford University Press, Inc. ISBN: 0199678111.

Bowers, J. S. (2009). "On the Biological Plausibility of Grandmother Cells: Implications for Neural Network Theories in Psychology and Neuroscience". In: *Psychological Review* 116.1, pp. 220–251. DOI: 10.1037/a0014462.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Bradbury, James, Roy Frostig, Peter Hawkins, et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. URL: http://github.com/google/jax.

Bradski, G. (2000). "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*.

Braun, Dan et al. (Feb. 2025). *Interpretability in Parameter Space: Minimizing Mechanistic Description Length with Attribution-based Parameter Decomposition*. DOI: 10.48550/arXiv.2501.14926. URL: http://arxiv.org/abs/2501.14926 (visited on 08/04/2025).

Breiman, Leo, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone (Oct. 1984). *Classification And Regression Trees*. New York: Routledge. Chap. 3. ISBN: 978-1-315-13947-0.

Bricken, Trenton et al. (Oct. 2023). "Towards monosemanticity: Decomposing language models with dictionary learning". In: *Transformer Circuits Thread*. URL: https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Brooks, Rodney A. (1991). "Intelligence without representation". In: *Artificial Intelligence* 47.1, pp. 139–159. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(91)90053-M. URL: https://www.sciencedirect.com/science/article/pii/000437029190053M.

Brown, Tom B. et al. (Dec. 2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 12/21/2024).

Bubeck, Sébastien et al. (2023). "Sparks of artificial general intelligence: Early experiments with gpt-4". In: *arXiv preprint arXiv:2303.12712*.

Burgess, Christopher P., Irina Higgins, et al. (Apr. 2018). *Understanding Disentangling in β-VAE*. DOI: 10.48550/arXiv.1804.03599. URL: http://arxiv.org/abs/1804.03599 (visited on 07/17/2025).

Burgess, Christopher P., Loic Matthey, Nicholas Watters, et al. (Jan. 2019). "MONet: Unsupervised Scene Decomposition and Representation". In: *arXiv:1901.11390 [cs, stat]*. URL: http://arxiv.org/abs/1901.11390 (visited on 01/13/2021).

Bushnaq, Lucius, Dan Braun, and Lee Sharkey (June 2025). *Stochastic Parameter Decomposition*. DOI: 10.48550/arXiv.2506.20790. URL: http://arxiv.org/abs/2506.20790 (visited on 08/04/2025).

Bussmann, Bart, Patrick Leask, and Neel Nanda (Dec. 2024). *BatchTopK Sparse Autoencoders*. DOI: 10.48550/arXiv.2412.06410. URL: http://arxiv.org/abs/2412.06410 (visited on 08/04/2025).

Carlsmith, Joseph (June 2022). *Is Power-Seeking AI an Existential Risk?* DOI: 10.48550/arXiv.2206.13353. URL: http://arxiv.org/abs/2206.13353 (visited on 10/23/2022).

Chang, Michael, Thomas L. Griffiths, and Sergey Levine (Jan. 2023). *Object Representations as Fixed Points: Training Iterative Refinement Algorithms with Implicit Differentiation*. DOI: 10.48550/arXiv.2207.00787. URL: http://arxiv.org/abs/2207.00787 (visited on 07/18/2025).

Chen, Ricky T. Q., Xuechen Li, Roger Grosse, and David Duvenaud (Apr. 2019). "Isolating Sources of Disentanglement in Variational Autoencoders". In: *arXiv:1802.04942 [cs, stat]*. URL: http://arxiv.org/abs/1802.04942 (visited on 02/26/2021).

Chen, Ting et al. (2022). "Pix2seq: A language modeling framework for object detection". In: *International Conference on Learning Representations*.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (Oct. 2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Ed. by Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi. Doha, Qatar: Association for Computational Linguistics, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012/.

Chollet, Francois et al. (May 2025). *ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems*. DOI: 10.48550/arXiv.2505.11831. URL: http://arxiv.org/abs/2505.11831 (visited on 08/04/2025).

Chomsky, Noam, Ian Roberts, and Jeffrey Watumull (Mar. 2023). "Opinion | Noam Chomsky: The False Promise of ChatGPT". In: *The New York Times*. ISSN: 0362-4331. URL: https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html (visited on 07/04/2025).

Chowdhery, Aakanksha et al. (2022). "PaLM: Scaling Language Modeling with Pathways". In: *Journal of Machine Learning Research* 23.240, pp. 1–118. ISSN: 1532-4435. URL: http://jmlr.org/papers/v23/22-1144.html (visited on 12/21/2024).

Cingillioglu, Nuri (2022). "End-to-end neuro-symbolic learning of logic-based inference". PhD thesis. Imperial College London.

Cingillioglu, Nuri and Alessandra Russo (Feb. 2022). *Pix2rule: End-to-end Neuro-symbolic Rule Learning*. DOI: 10.48550/arXiv.2106.07487. URL: http://arxiv.org/abs/2106.07487 (visited on 07/09/2025).

Conmy, Arthur et al. (Oct. 2023). *Towards Automated Circuit Discovery for Mechanistic Interpretability*. DOI: 10.48550/arXiv.2304.14997. URL: http://arxiv.org/abs/2304.14997 (visited on 08/04/2025).

Crawford, Eric and Joelle Pineau (July 2019). "Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 3412–3420. ISSN: 2374-3468. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4216 (visited on 04/26/2021).

Creswell, Antonia, Rishabh Kabra, Chris Burgess, and Murray Shanahan (Mar. 2021). "Unsupervised Object-Based Transition Models for 3D Partially Observable Environments". In: *arXiv:2103.04693 [cs]*. URL: http://arxiv.org/abs/2103.04693 (visited on 03/20/2021).

Creswell, Antonia, Kyriacos Nikiforou, Oriol Vinyals, et al. (July 2020). "AlignNet: Unsupervised Entity Alignment". In: *arXiv:2007.08973 [cs]*. URL: http://arxiv.org/abs/2007.08973 (visited on 12/10/2020).

Cunningham, Hoagy et al. (Oct. 2023). *Sparse Autoencoders Find Highly Interpretable Features in Language Models*. DOI: 10.48550/arXiv.2309.08600. URL: http://arxiv.org/abs/2309.08600 (visited on 07/07/2025).

Cunnington, Daniel, Mark Law, Jorge Lobo, and Alessandra Russo (Feb. 2023). "FFNSL: Feed-Forward Neural-Symbolic Learner". In: *Machine Learning* 112.2, pp. 515–569. ISSN: 1573-0565. DOI: 10.1007/s10994-022-06278-6. URL: https://doi.org/10.1007/s10994-022-06278-6 (visited on 07/09/2025).

Čyras, Kristijonas et al. (May 2021). *Argumentative XAI: A Survey*. DOI: 10.48550/arXiv.2105.11266. URL: http://arxiv.org/abs/2105.11266 (visited on 07/05/2025).

D'Angelo, Francesco, Maksym Andriushchenko, Aditya Varre, and Nicolas Flammarion (Nov. 2024). *Why Do We Need Weight Decay in Modern Deep Learning?* DOI: 10.48550/arXiv.2310.04415. URL: http://arxiv.org/abs/2310.04415 (visited on 07/20/2025).

Dang-Nhu, Raphaël and Angelika Steger (Jan. 2021). "Evaluating Disentanglement of Structured Latent Representations". In: *arXiv:2101.04041 [cs]*. URL: http://arxiv.org/abs/2101.04041 (visited on 03/14/2021).

Dar, Guy, Mor Geva, Ankit Gupta, and Jonathan Berant (Dec. 2022). *Analyzing Transformers in Embedding Space*. URL: http://arxiv.org/abs/2209.02535 (visited on 01/26/2023).

Davis, Ernest (July 2016). "Evaluating CYC: Preliminary Notes". In.

De Raedt, Luc, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra (Mar. 2020). "From Statistical Relational to Neuro-Symbolic Artificial Intelligence". In: *arXiv:2003.08316 [cs]*. URL: http://arxiv.org/abs/2003.08316 (visited on 07/14/2021).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dong, Honghua et al. (2019). "NEURAL LOGIC MACHINES". In.

Dong, Qingxiu et al. (Oct. 2024). *A Survey on In-context Learning*. DOI: 10.48550/arXiv.2301.00234. URL: http://arxiv.org/abs/2301.00234 (visited on 08/04/2025).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, et al. (2021). "An image is worth 16x16 words: Transformers for image recognition at scale". In: *International Conference on Learning Representations*.

Dreyfus, Hubert L. (1971). *What Computers Still Can?T Do: A Critique of Artificial Reason*. MIT Press.

Dunefsky, Jacob, Philippe Chlenski, and Neel Nanda (Nov. 2024). *Transcoders Find Interpretable LLM Feature Circuits*. DOI: 10.48550/arXiv.2406.11944. URL: http://arxiv.org/abs/2406.11944 (visited on 08/04/2025).

Eastwood, Cian and Christopher K. I. Williams (Feb. 2018). "A Framework for the Quantitative Evaluation of Disentangled Representations". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=By-7dz-AZ (visited on 02/26/2021).

Ebner, Paula and Jessica Szczuka (Mar. 2025). *Predicting Romantic Human-Chatbot Relationships: A Mixed-Method Study on the Key Psychological Factors*. DOI: 10.48550/arXiv.2503.00195. URL: http://arxiv.org/abs/2503.00195 (visited on 07/03/2025).

Elhage, Nelson, Tristan Hume, Catherine Olsson, Neel Nanda, et al. (2022). "Softmax Linear Units". In: *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/solu/index.html.

Elhage, Nelson, Tristan Hume, Catherine Olsson, Nicholas Schiefer, et al. (2022). "Toy Models of Superposition in Neural Networks". In: *Transformer Circuits Thread*. URL: https://arxiv.org/abs/2209.10652.

Elhage, Nelson, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yan Bai, et al. (2021). "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread*. URL: https://transformer-circuits.pub/2021/framework/index.html.

Elhage, Nelson, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, et al. (2021). *A Mathematical Framework for Transformer Circuits*. Transformer Circuits Thread. URL: https://transformer-circuits.pub/2021/framework/index.html.

Ellis, Kevin et al. (June 2020). "DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning". In: *arXiv:2006.08381 [cs]*. URL: http://arxiv.org/abs/2006.08381 (visited on 07/13/2021).

Elsayed, Gamaleldin F. et al. (2022). "SAVi++: Towards End-to-End Object-Centric Learning from Real-World Videos". In: *Advances in Neural Information Processing Systems*. Vol. 35.

Emami, Patrick, Pan He, Sanjay Ranka, and Anand Rangarajan (June 2021). "Efficient Iterative Amortized Inference for Learning Symmetric and Disentangled Multi-Object Representations". In: *arXiv:2106.03630 [cs]*. URL: http://arxiv.org/abs/2106.03630 (visited on 06/15/2021).

Engelcke, Martin, Oiwi Parker Jones, and Ingmar Posner (Apr. 2021). "GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement". In: *arXiv:2104.09958 [cs, stat]*. URL: http://arxiv.org/abs/2104.09958 (visited on 04/21/2021).

Engelcke, Martin, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner (Nov. 2020). "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *arXiv:1907.13052 [cs, stat]*. URL: http://arxiv.org/abs/1907.13052 (visited on 07/15/2021).

Engelcke, Martin, Oiwi Parker Jones, and Ingmar Posner (2021). "GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement". In: *Advances in Neural Information Processing Systems*. Vol. 34.

Ethayarajh, Kawin, David Duvenaud, and Graeme Hirst (July 2019). "Towards Understanding Linear Word Analogies". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, pp. 3253–3262. DOI: 10.18653/v1/P19-1315. URL: https://aclanthology.org/P19-1315/.

Evans, Richard and Edward Grefenstette (Jan. 2018). *Learning Explanatory Rules from Noisy Data*. DOI: 10.48550/arXiv.1711.04574. URL: http://arxiv.org/abs/1711.04574 (visited on 07/09/2025).

Evci, Utku, Fabian Pedregosa, Aidan Gomez, and Erich Elsen (Oct. 2020). *The Difficulty of Training Sparse Neural Networks*. DOI: 10.48550/arXiv.1906.10732. URL: http://arxiv.org/abs/1906.10732 (visited on 07/20/2025).

Fan, Ke et al. (2024). "Adaptive Slot Attention: Object Discovery with Dynamic Slot Number". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23083–23092.

Fandinno, Jorge and Claudia Schulz (Sept. 2018). *Answering the "Why" in Answer Set Programming - A Survey of Explanation Approaches*. DOI: 10.48550/arXiv.1809.08034. URL: http://arxiv.org/abs/1809.08034 (visited on 07/05/2025).

Faulkner, Ryan and Daniel Zoran (Oct. 2022). *Solving Reasoning Tasks with a Slot Transformer*. DOI: 10.48550/arXiv.2210.11394. URL: http://arxiv.org/abs/2210.11394 (visited on 08/02/2025).

Frankle, Jonathan and Michael Carbin (2019). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *ICLR*. arXiv: 1803.03635.

Fukuda, Keisuke, Edward Awh, and Edward K. Vogel (Apr. 2010). "Discrete Capacity Limits in Visual Working Memory". In: *Current opinion in neurobiology* 20.2, pp. 177–182. ISSN: 0959-4388. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3019116/ (visited on 07/14/2021).

Gale, Trevor, Erich Elsen, and Sara Hooker (2019). "The State of Sparsity in Deep Neural Networks". In: *arXiv preprint arXiv:1902.09574*.

Gallego, Juan A. et al. (2020). "Long-Term Stability of Cortical Population Dynamics Underlying Consistent Behavior". In: *Nature Neuroscience* 23.2, pp. 260–270. DOI: 10.1038/s41593-019-0555-4.

Ganguli, Deep et al. (Nov. 2022). *Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned*. DOI: 10.48550/arXiv.2209.07858. URL: http://arxiv.org/abs/2209.07858 (visited on 03/01/2023).

Garcez, Artur d'Avila and Luis C. Lamb (Dec. 2020). "Neurosymbolic AI: The 3rd Wave". In: *arXiv:2012.05876 [cs]*. URL: http://arxiv.org/abs/2012.05876 (visited on 07/14/2021).

Garipov, Timur et al. (2018). "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs". In: *NeurIPS*. arXiv: 1802.10026.

Gaunt, Alexander L. et al. (Aug. 2016). "TerpreT: A Probabilistic Programming Language for Program Induction". In: *arXiv:1608.04428 [cs]*. arXiv: 1608.04428 [cs].

Geiger, Atticus et al. (Feb. 2024). *Finding Alignments Between Interpretable Causal Variables and Distributed Neural Representations*. DOI: 10.48550/arXiv.2303.02536. URL: http://arxiv.org/abs/2303.02536 (visited on 08/04/2025).

Gibson, James Jerome (1979). *The Ecological Approach to Visual Perception.* The Ecological Approach to Visual Perception. Boston, MA, US: Houghton, Mifflin and Company, pp. xiv, 332. ISBN: 0-39527-049-9 (Hardcover).

Gong, Yuan, Yu-An Chung, and James Glass (2021). "AST: Audio Spectrogram Transformer". In: *Interspeech*.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, et al. (2016). *Deep learning*. Vol. 1. MIT Press.

Google, Gemini Team et al. (Dec. 2023). *Gemini: A Family of Highly Capable Multimodal Models*. DOI: 10.48550/arXiv.2312.11805. arXiv: 2312.11805 [cs.LG]. URL: http://arxiv.org/abs/2312.11805 (visited on 12/21/2024).

Google DeepMind (May 2024). *Gemini Breaks New Ground with a Faster Model, Longer Context, and AI Agents*. Google I/O 2024 announcement post. URL: https://blog.google/technology/ai/google-gemini-update-flash-ai-assistant-io-2024/ (visited on 11/01/2025).

– (Feb. 2025a). *Gemini 2.0 Is Now Available to Everyone*. Google announcement post. URL: https://blog.google/technology/google-deepmind/gemini-model-updates-february-2025/ (visited on 11/01/2025).

– (May 2025b). *Gemini 2.5: Our Most Intelligent Models Are Getting Even Better*. Google I/O 2025 announcement post. URL: https://blog.google/technology/google-deepmind/google-gemini-updates-io-2025/ (visited on 11/01/2025).

Google DeepMind (Mar. 2025c). *Introducing Gemma 3: The Most Capable Model You Can Run on a Single GPU or TPU*. Google announcement post. URL: https://blog.google/technology/developers/gemma-3/ (visited on 11/01/2025).

Google DeepMind, Sundar Pichai, and Demis Hassabis (Feb. 2024). *Our Next-Generation Model: Gemini 1.5*. Google announcement post. URL: https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/ (visited on 11/01/2025).

Gotmare, Akhilesh, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher (June 2018). *Using Mode Connectivity for Loss Landscape Analysis*. DOI: 10.48550/arXiv.1806.06977. URL: http://arxiv.org/abs/1806.06977 (visited on 07/20/2025).

Goyal, Anirudh, Aniket Rajiv Didolkar, Nan Rosemary Ke, et al. (2021). "Neural Production Systems". In: *NeurIPS*. URL: https://openreview.net/forum?id=xQGYquca0gB (visited on 05/19/2022).

Goyal, Yash et al. (July 2017). "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering". In: *CVPR*. Honolulu, HI, USA: IEEE, pp. 6904–6913. DOI: 10.1109/CVPR.2017.670. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Goyal_Making_the_V_CVPR_2017_paper.html (visited on 12/21/2024).

Greff, Klaus, Raphaël Lopez Kaufman, Rishabh Kabra, et al. (July 2020). "Multi-Object Representation Learning with Iterative Variational Inference". In: *arXiv:1903.00450 [cs, stat]*. URL: http://arxiv.org/abs/1903.00450 (visited on 05/28/2021).

Greff, Klaus, Sjoerd van Steenkiste, and Jürgen Schmidhuber (Dec. 2020). *On the Binding Problem in Artificial Neural Networks*. URL: http://arxiv.org/abs/2012.05208 (visited on 05/24/2022).

Gulati, Anmol, James Qin, Chung-Cheng Chiu, et al. (2020). "Conformer: Convolution-augmented transformer for speech recognition". In: *Interspeech*.

Gurnee, Wes and Max Tegmark (2023). *Language Models Represent Space and Time*. arXiv: 2310.02207 [cs.LG].

He, Zhen et al. (2019). "Tracking by Animation: Unsupervised Learning of Multi-Object Attentive Trackers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1318–1327. URL: https://openaccess.thecvf.com/content_CVPR_2019/html/He_Tracking_by_Animation_Unsupervised_Learning_of_Multi-Object_Attentive_Trackers_CVPR_2019_paper.html (visited on 07/09/2021).

He, Zhengfu, Xuyang Ge, Qiong Tang, et al. (2024). "Dictionary learning improves patch-free circuit discovery in mechanistic interpretability: A case study on othello-gpt". In: *arXiv preprint arXiv:2402.12201*.

Heerden, Imke van (Mar. 2025). *Ways of Seeing, and Selling, AI Art*. DOI: 10.48550/arXiv.2503.07685. URL: http://arxiv.org/abs/2503.07685 (visited on 07/03/2025).

Hendrycks, Dan (2024). *Introduction to AI Safety, Ethics and Society*. Taylor & Francis. ISBN: 9781032798028. URL: https://www.aisafetybook.com.

Hennigan, Tom, Trevor Cai, Tamara Norman, and Igor Babuschkin (2020). *Haiku: Sonnet for JAX*. Version 0.0.3. URL: http://github.com/deepmind/dm-haiku.

Hessel, Matteo et al. (2020). *Optax: composable gradient transformation and optimisation, in JAX!* Version 0.0.1. URL: http://github.com/deepmind/optax.

Hewitt, Luke B., Tuan Anh Le, and Joshua B. Tenenbaum (July 2020). "Learning to Learn Generative Programs with Memoised Wake-Sleep". In: *arXiv:2007.03132 [cs]*. URL: http://arxiv.org/abs/2007.03132 (visited on 07/13/2021).

Higgins, Irina et al. (Nov. 2016). "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR2017*. URL: https://openreview.net/forum?id=Sy2fzU9gl (visited on 06/13/2021).

Hinton, G., P Dayan, B. Frey, and R. Neal (May 1995). "The "Wake-Sleep" Algorithm for Unsupervised Neural Networks". In: *Science* 268.5214, pp. 1158–1161. ISSN: 0036-8075, 1095-9203. URL: https://www.sciencemag.org/lookup/doi/10.1126/science.7761831 (visited on 07/13/2021).

Hinton, Geoffrey E., Alex Krizhevsky, and Sida D. Wang (2011). "Transforming Auto-Encoders". In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. Ed. by Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski. Vol. 6791. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 44–51. ISBN: 978-3-642-21734-0 978-3-642-21735-7. URL: http://link.springer.com/10.1007/978-3-642-21735-7_6 (visited on 07/15/2021).

Hochreiter, Sepp (1991). "Untersuchungen zu dynamischen neuronalen Netzen". Diploma thesis. Technische Universität München.

Hochreiter, Sepp and Jürgen Schmidhuber (1997a). "Flat Minima". In: *Neural Computation* 9.1, pp. 1–42.

– (1997b). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (Jan. 1989). "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208 (visited on 07/04/2025).

Huang, Tiansheng et al. (Sept. 2024). *Antidote: Post-fine-tuning Safety Alignment for Large Language Models against Harmful Fine-tuning*. DOI: 10.48550/arXiv.2408.09600. URL: http://arxiv.org/abs/2408.09600 (visited on 07/07/2025).

Hubert, Lawrence and Phipps Arabie (Dec. 1985). "Comparing Partitions". In: *Journal of Classification* 2.1, pp. 193–218. ISSN: 1432-1343. URL: https://doi.org/10.1007/BF01908075 (visited on 07/06/2021).

Hudson, Drew A. and Christopher D. Manning (Apr. 2018). *Compositional Attention Networks for Machine Reasoning*. DOI: 10.48550/arXiv.1803.03067. URL: http://arxiv.org/abs/1803.03067 (visited on 07/19/2025).

– (June 2019). "GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering". In: *CVPR*. Long Beach, CA, USA: IEEE, pp. 6700–6709. DOI: 10.1109/CVPR.2019.00686. URL: https://openaccess.thecvf.com/content_CVPR_2019/html/Hudson_GQA_A_New_Dataset_for_Real-World_Visual_Reasoning_and_Compositional_CVPR_2019_paper.html (visited on 12/21/2024).

Ioffe, Sergey and Christian Szegedy (Mar. 2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. DOI: 10.48550/arXiv.1502.03167. URL: http://arxiv.org/abs/1502.03167 (visited on 07/03/2025).

Ivanitskiy, M. I., A. Sandoval, et al. (Oct. 2025). "maze-dataset: Maze Generation with Algorithmic Variety and Representational Flexibility". In: *Journal of Open Source Software* 10.114, p. 8633. DOI: 10.21105/joss.08633. URL: https://joss.theoj.org/papers/10.21105/joss.08633.

Ivanitskiy, M. I., A. F. Spies, et al. (2024). "Linearly Structured World Representations in Maze-Solving Transformers". In: *Proceedings of the UniReps: Unifying Representations in Neural Models Workshop at NeurIPS 2023*. Vol. 243. Proceedings of Machine Learning Research. PMLR, pp. 133–143. URL: https://proceedings.mlr.press/v243/ivanitskiy24a.html (visited on 12/21/2024).

Izadi, Amirmohammad et al. (July 2025). *Visual Structures Helps Visual Reasoning: Addressing the Binding Problem in VLMs*. DOI: 10.48550/arXiv.2506.22146. URL: http://arxiv.org/abs/2506.22146 (visited on 07/07/2025).

Izmailov, Pavel et al. (2018). "Averaging Weights Leads to Wider Optima and Better Generalization". In: *UAI*. arXiv: 1803.05407.

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu (Feb. 2016). "Spatial Transformer Networks". In: *arXiv:1506.02025 [cs]*. URL: http://arxiv.org/abs/1506.02025 (visited on 06/11/2021).

Jang, Eric, Shixiang Gu, and Ben Poole (Aug. 2017). "Categorical Reparameterization with Gumbel-Softmax". In: *arXiv:1611.01144 [cs, stat]*. URL: http://arxiv.org/abs/1611.01144 (visited on 07/14/2021).

Jenner, Erik, Shreyas Kapur, Vasil Georgiev, et al. (2024). "Evidence of Learned Look-Ahead in a Chess-Playing Neural Network". In: *arXiv preprint arXiv:2406.00877*.

Jermyn, Adam and Adly Templeton (Jan. 2024). *Anthropic Circuits Updates - January 2024*. URL: https://transformer-circuits.pub/2024/jan-update#dict-learning-resampling (visited on 09/30/2024).

Jiang, Jindong, Sepehr Janghorbani*, Gerard De Melo, and Sungjin Ahn (Sept. 2019). "SCALOR: Generative World Models with Scalable Object Representations". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SJxrKgStDH (visited on 02/10/2021).

Johnson, Justin et al. (July 2017). "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning". In: *CVPR*. Honolulu, HI, USA: IEEE, pp. 2901–2910. DOI: 10.1109/CVPR.2017.215. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Johnson_CLEVR_A_Diagnostic_CVPR_2017_paper.html (visited on 12/21/2024).

Jorgensen, Ole, Dylan Cope, Nandi Schoots, and Murray Shanahan (Dec. 2023). *Improving Activation Steering in Language Models with Mean-Centring*. DOI: 10.48550/arXiv.2312.03813. URL: http://arxiv.org/abs/2312.03813 (visited on 08/03/2025).

Kabra, Rishabh et al. (2019). *Multi-Object Datasets*. https://github.com/deepmind/multi-object-datasets/.

Kanwisher, N., J. McDermott, and M. M. Chun (1997). "The Fusiform Face Area: A Module in Human Extrastriate Cortex Specialized for Face Perception". In: *Journal of Neuroscience* 17.11, pp. 4302–4311. DOI: 10.1523/JNEUROSCI.17-11-04302.1997.

Karvonen, Adam (2024). "Emergent world models and latent variable estimation in chess-playing language models". In: *arXiv preprint arXiv:2403.15498*.

Karvonen, Adam et al. (June 2025). *SAEBench: A Comprehensive Benchmark for Sparse Autoencoders in Language Model Interpretability*. DOI: 10.48550/arXiv.2503.09532. URL: http://arxiv.org/abs/2503.09532 (visited on 08/04/2025).

Kawaguchi, Kenji and Yoshua Bengio (Oct. 2019). "Depth with Nonlinearity Creates No Bad Local Minima in ResNets". In: *Neural Networks* 118, pp. 167–174. ISSN: 08936080. DOI: 10.1016/j.neunet.2019.06.009. URL: http://arxiv.org/abs/1810.09038 (visited on 07/03/2025).

Keskar, Nitish Shirish et al. (2017). "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *ICLR*. arXiv: 1609.04836.

Kim, Been et al. (June 2018). *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)*. DOI: 10.48550/arXiv.1711.11279. URL: http://arxiv.org/abs/1711.11279 (visited on 08/04/2025).

Kingma, Diederik P and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *ICLR*. arXiv: 1412.6980.

Kingma, Diederik P. and Max Welling (May 2014). "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]*. URL: http://arxiv.org/abs/1312.6114 (visited on 08/26/2020).

Kipf, Thomas, Elise van der Pol, and Max Welling (2020). "Contrastive Learning of Structured World Models". In: *ICLR*. URL: http://arxiv.org/abs/1911.12247 (visited on 12/10/2020).

Kocijan, Vid, Ernest Davis, Thomas Lukasiewicz, et al. (2022). *The Defeat of the Winograd Schema Challenge*. URL: http://arxiv.org/abs/2201.02387 (visited on 05/25/2022).

Kojima, Takeshi et al. (Dec. 2022). "Large Language Models Are Zero-Shot Reasoners". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 22199–22213. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html (visited on 12/21/2024).

Kori, Avinash et al. (June 2024). *Identifiable Object-Centric Representation Learning via Probabilistic Slot Attention*. DOI: 10.48550/arXiv.2406.07141. URL: http://arxiv.org/abs/2406.07141 (visited on 07/18/2025).

Kosiorek, Adam R., Hyunjik Kim, Ingmar Posner, and Yee Whye Teh (Nov. 2018). "Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects". In: *arXiv:1806.01794 [cs, stat]*. URL: http://arxiv.org/abs/1806.01794 (visited on 07/09/2021).

Kosmyna, Nataliya et al. (June 2025). *Your Brain on ChatGPT: Accumulation of Cognitive Debt When Using an AI Assistant for Essay Writing Task*. DOI: 10.48550/arXiv.2506.08872. URL: http://arxiv.org/abs/2506.08872 (visited on 07/03/2025).

Kramár, János, Tom Lieberum, Rohin Shah, and Neel Nanda (Mar. 2024). *AtP*: An Efficient and Scalable Method for Localizing LLM Behaviour to Components*. DOI: 10.48550/arXiv.2403.00745. URL: http://arxiv.org/abs/2403.00745 (visited on 08/04/2025).

Krogh, Anders and John A. Hertz (1991). "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 4.

Kudo, Taku and John Richardson (2018). "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Law, Mark et al. (Apr. 2020). "FastLAS: Scalable Inductive Logic Programming Incorporating Domain-Specific Optimisation Criteria". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03, pp. 2877–2885. DOI: 10.1609/aaai.v34i03.5678. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5678.

Leeb, Felix et al. (July 2021). "Structure by Architecture: Disentangled Representations without Regularization". In: *arXiv:2006.07796 [cs, stat]*. URL: http://arxiv.org/abs/2006.07796 (visited on 07/14/2021).

Lettvin, J. Y., H. R. Maturana, W. S. McCulloch, and W. H. Pitts (1959). "What the Frog's Eye Tells the Frog's Brain". In: *Proceedings of the IRE* 47.11, pp. 1940–1951. DOI: 10.1109/JRPROC.1959.287207.

Li, Hao et al. (Nov. 2018). *Visualizing the Loss Landscape of Neural Nets*. DOI: 10.48550/arXiv.1712.09913. URL: http://arxiv.org/abs/1712.09913 (visited on 07/02/2025).

Li, Junnan, Dongxu Li, Silvio Savarese, and Steven Hoi (June 2023). *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. DOI: 10.48550/arXiv.2301.12597. URL: http://arxiv.org/abs/2301.12597 (visited on 08/04/2025).

Li, Kenneth, Aspen K Hopkins, David Bau, et al. (2022). "Emergent world representations: Exploring a sequence model trained on a synthetic task". In: *arXiv preprint arXiv:2210.13382*.

Li, Yifan et al. (Dec. 2023). "Evaluating Object Hallucination in Large Vision-Language Models". In: *EMNLP*. ACL. URL: http://arxiv.org/pdf/2305.10355 (visited on 12/21/2024).

Liang, Weixin et al. (Feb. 2025). *The Widespread Adoption of Large Language Model-Assisted Writing Across Society*. DOI: 10.48550/arXiv.2502.09747. URL: http://arxiv.org/abs/2502.09747 (visited on 07/03/2025).

Lieberum, Tom, Matthew Rahtz, János Kramár, et al. (2023). *Does Circuit Analysis Interpretability Scale? Evidence from Multiple Choice Capabilities in Chinchilla*. arXiv: 2307.09458 [cs.LG]. URL: http://arxiv.org/abs/2307.09458.

Lillicrap, Timothy P. et al. (June 2020). "Backpropagation and the Brain". In: *Nature Reviews. Neuroscience* 21.6, pp. 335–346. ISSN: 1471-0048. DOI: 10.1038/s41583-020-0277-3.

Lin, Zhixuan, Yi-Fu Wu, Skand Vishwanath Peri, et al. (Mar. 2020). "SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition". In: *arXiv:2001.02407 [cs, eess, stat]*. URL: http://arxiv.org/abs/2001.02407 (visited on 05/28/2021).

Liu, Pengfei et al. (2023). "Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing". In: *ACM Computing Surveys* 55.9, pp. 1–35. ISSN: 0360-0300. DOI: 10.1145/3560815. URL: https://doi.org/10.1145/3560815 (visited on 12/21/2024).

Liu, Ziming, Ouail Kitouni, Niklas Nolte, et al. (Oct. 2022). *Towards Understanding Grokking: An Effective Theory of Representation Learning*. URL: http://arxiv.org/abs/2205.10343 (visited on 09/27/2023).

Locatello, Francesco, Stefan Bauer, Mario Lucic, et al. (June 2019). "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *arXiv:1811.12359 [cs, stat]*. URL: http://arxiv.org/abs/1811.12359 (visited on 02/18/2021).

Locatello, Francesco, Stefan Bauer, Mario Lucic, et al. (July 2020). "A Commentary on the Unsupervised Learning of Disentangled Representations". In: *arXiv:2007.14184 [cs, stat]*. URL: http://arxiv.org/abs/2007.14184 (visited on 07/13/2021).

Locatello, Francesco, Dirk Weissenborn, Thomas Unterthiner, et al. (2020). "Object-Centric Learning with Slot Attention". In: *NeurIPS*. URL: http://arxiv.org/abs/2006.15055.

Loshchilov, Ilya and Frank Hutter (Jan. 2019). *Decoupled Weight Decay Regularization*. DOI: 10.48550/arXiv.1711.05101. URL: http://arxiv.org/abs/1711.05101 (visited on 08/03/2025).

Love, Fraser (2024). *NNTikZ - TikZ Diagrams for Deep Learning and Neural Networks*. GitHub repository. URL: https://github.com/fraserlove/nntikz.

Löwe, Sindy et al. (Nov. 2020). "Learning Object-Centric Video Models by Contrasting Sets". In: *arXiv:2011.10287 [cs]*. URL: http://arxiv.org/abs/2011.10287 (visited on 12/11/2020).

Lu, Jiasen, Dhruv Batra, Devi Parikh, and Stefan Lee (Aug. 2019). *ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks*. DOI: 10.48550/arXiv.1908.02265. URL: http://arxiv.org/abs/1908.02265 (visited on 08/04/2025).

Lu, Jiasen, Christopher Clark, et al. (Dec. 2023). *Unified-IO 2: Scaling Autoregressive Multimodal Models with Vision, Language, Audio, and Action*. DOI: 10.48550/arXiv.2312.17172. URL: http://arxiv.org/abs/2312.17172 (visited on 08/04/2025).

Lu, Yao et al. (May 2022). "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: ACL, pp. 8086–8098. DOI: 10.18653/v1/2022.acl-long.556. URL: https://aclanthology.org/2022.acl-long.556 (visited on 12/21/2024).

Mandt, Stephan, Matthew D. Hoffman, and David M. Blei (2017). "Stochastic Gradient Descent as Approximate Bayesian Inference". In: *Journal of Machine Learning Research* 18.134, pp. 1–35. arXiv: 1704.04289.

Manhaeve, Robin et al. (Dec. 2018). "DeepProbLog: Neural Probabilistic Logic Programming". In: *arXiv:1805.10872 [cs]*. URL: http://arxiv.org/abs/1805.10872 (visited on 12/10/2020).

Marks, Samuel et al. (2024). "Sparse feature circuits: Discovering and editing interpretable causal graphs in language models". In: arXiv: 2403.19647 [cs.LG]. URL: https://arxiv.org/abs/2403.19647.

Martín Abadi, Ashish Agarwal, Paul Barham, et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Martins, André F. T. and Ramón Fernandez Astudillo (2016). "From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification". In: *ICML*. URL: http://arxiv.org/abs/1602.02068 (visited on 05/24/2022).

Mazzia, Vittorio, Francesco Salvetti, and Marcello Chiaberge (Jan. 2021). "Efficient-CapsNet: Capsule Network with Self-Attention Routing". In: *arXiv:2101.12491 [cs]*. URL: http://arxiv.org/abs/2101.12491 (visited on 02/28/2021).

McCarthy, John, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon (Aug. 1955). "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence". In: *AI Magazine* 27.4. Reprinted in AI Magazine, Winter 2006, pp. 12–14. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/1904.

McCulloch, Warren S. and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4, pp. 115–133.

McDermott, Drew (1987). "A critique of pure reason". In: *Computational Intelligence* 3.1, pp. 151–160. DOI: https://doi.org/10.1111/j.1467-8640.1987.tb00183.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8640.1987.tb00183.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.1987.tb00183.x.

McDougall, Callum (2024). *SAE Visualizer*. URL: https://github.com/callummcdougall/sae_vis.

McGrath, Thomas, Andrei Kapishnikov, Nenad Tomašev, et al. (2022). "Acquisition of chess knowledge in alphazero". In: *Proceedings of the National Academy of Sciences* 119.47, e2206625119.

McKenzie, Ian R. et al. (2025). *STACK: Adversarial Attacks on LLM Safeguard Pipelines*. arXiv: 2506.24068 [cs.CL]. URL: https://arxiv.org/abs/2506.24068.

Merullo, Jack, Carsten Eickhoff, and Ellie Pavlick (2025). "Talking heads: understanding inter-layer communication in transformer language models". In: *Proceedings of the 38th International Conference on Neural Information Processing Systems*. NIPS '24. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 9798331314385.

Meta AI (Apr. 2025). *The Llama 4 Herd: The Beginning of a New Era of Natively Multimodal AI Innovation*. Meta announcement post. URL: https://ai.meta.com/blog/llama-4-multimodal-intelligence/ (visited on 11/01/2025).

Metaculus (2025). *Date of Artificial General Intelligence*. Accessed 6 June 2025. URL: https://www.metaculus.com/questions/5121/date-of-artificial-general-intelligence/ (visited on 06/06/2025).

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (Sept. 2013). *Efficient Estimation of Word Representations in Vector Space*. DOI: 10.48550/arXiv.1301.3781. URL: http://arxiv.org/abs/1301.3781.

Miller, George A. (1956). "The Magical Number Seven, plus or Minus Two: Some Limits on Our Capacity for Processing Information". In: *Psychological Review* 63.2, pp. 81–97. ISSN: 1939-1471(Electronic),0033-295X(Print).

Millière, Raphaël and Cameron Buckner (2024). *A Philosophical Introduction to Language Models - Part II: The Way Forward*. arXiv: 2405.03207 [cs.CL]. URL: https://arxiv.org/abs/2405.03207.

Min, Sewon et al. (Dec. 2022). "Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?" In: *EMNLP*. Abu Dhabi, UAE: ACL, pp. 11048–11064. DOI: 10.18653/v1/2022.emnlp-main.759. URL: https://aclanthology.org/2022.emnlp-main.759 (visited on 12/21/2024).

Minervini, Pasquale et al. (Nov. 2020). "Learning Reasoning Strategies in End-to-End Differentiable Proving". In: *ICML*. PMLR, pp. 6938–6949. URL: https://proceedings.mlr.press/v119/minervini20a.html (visited on 07/09/2025).

Mini, Ulisse et al. (Oct. 2023). *Understanding and Controlling a Maze-Solving Policy Network*. DOI: 10.48550/arXiv.2310.08043. URL: https://arxiv.org/abs/2310.08043.

Minsky, Marvin and Seymour Papert (1969). *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press.

Mistral AI (Mar. 2025). *Mistral Small 3.1*. Mistral AI announcement post. URL: https://mistral.ai/news/mistral-small-3-1 (visited on 11/01/2025).

Montero, Milton Llera et al. (Sept. 2020). "The Role of Disentanglement in Generalisation". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=qbH974jKUVy (visited on 06/13/2021).

Musil, Tomáš and David Mareček (Sept. 2024). *Exploring Interpretability of Independent Components of Word Embeddings with Automated Word Intruder Test*. DOI: 10.48550/arXiv.2212.09580. URL: http://arxiv.org/abs/2212.09580 (visited on 07/27/2025).

Nair, Vinod and Geoffrey E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pp. 807–814.

Nakkiran, Preetum et al. (Dec. 2019). *Deep Double Descent: Where Bigger Models and More Data Hurt*. DOI: 10.48550/arXiv.1912.02292. URL: http://arxiv.org/abs/1912.02292 (visited on 07/20/2025).

Nanda, Neel (2023). "Actually, Othello-GPT Has A Linear Emergent World Representation". In: *Neel Nanda's Blog* 7.

Nanda, Neel and Joseph Bloom (2022). *TransformerLens*. URL: https://github.com/neelnanda-io/TransformerLens.

Nanda, Neel, Lawrence Chan, Tom Lieberum, et al. (Jan. 2023). *Progress Measures for Grokking via Mechanistic Interpretability*. DOI: 10.48550/arXiv.2301.05217. URL: http://arxiv.org/abs/2301.05217 (visited on 01/28/2023).

Newell, Allen and Herbert A. Simon (1976). "Computer Science as Empirical Inquiry: Symbols and Search". In: *Communications of the ACM* 19.3. ACM Turing Award Lecture, pp. 113–126. DOI: 10.1145/360018.360022. URL: https://dl.acm.org/doi/10.1145/360018.360022.

Nguyen, Quynh and Matthias Hein (2017). "The Loss Surface of Deep and Wide Neural Networks". In: *ICML*, pp. 2603–2612. arXiv: 1704.08045.

nostalgebraist (Aug. 2020). *interpreting GPT: the logit lens*. LessWrong. URL: https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Olah, Chris, Nick Cammarata, et al. (2020). "Zoom In: An Introduction to Circuits". In: *Distill*. DOI: 10.23915/distill.00024.001. URL: https://distill.pub/2020/circuits/zoom-in.

Olah, Chris, Arvind Satyanarayan, et al. (2018). "The Building Blocks of Interpretability". In: *Distill*. https://distill.pub/2018/building-blocks. DOI: 10.23915/distill.00010.

Olsson, Catherine, Nelson Elhage, Neel Nanda, et al. (2022a). *In-context Learning and Induction Heads*. arXiv: 2209.11895 [cs.LG]. URL: http://arxiv.org/abs/2209.11895.

Olsson, Catherine, Nelson Elhage, Neel Nanda, et al. (Sept. 2022b). *In-Context Learning and Induction Heads*. DOI: 10.48550/arXiv.2209.11895. URL: http://arxiv.org/abs/2209.11895.

OpenAI (Oct. 2023). *GPT-4V(ision) System Card*. OpenAI Technical Report. URL: https://openai.com/research/gpt-4v-system-card (visited on 12/21/2024).

– (July 2024a). *GPT-4o Mini: Advancing Cost-Efficient Intelligence*. OpenAI announcement post. URL: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ (visited on 11/01/2025).

– (May 2024b). *Hello GPT-4o*. OpenAI announcement post. URL: https://openai.com/index/hello-gpt-4o/ (visited on 11/01/2025).

– (Apr. 2025). *Introducing GPT-4.1 in the API*. OpenAI announcement post. URL: https://openai.com/index/gpt-4-1/ (visited on 11/01/2025).

Ouyang, Long et al. (Mar. 2022). *Training Language Models to Follow Instructions with Human Feedback*. DOI: 10.48550/arXiv.2203.02155. URL: http://arxiv.org/abs/2203.02155 (visited on 07/07/2025).

Panickssery, Nina et al. (July 2024). *Steering Llama 2 via Contrastive Activation Addition*. DOI: 10.48550/arXiv.2312.06681. URL: http://arxiv.org/abs/2312.06681 (visited on 08/04/2025).

Park, Kiho, Yo Joong Choe, and Victor Veitch (July 2024). *The Linear Representation Hypothesis and the Geometry of Large Language Models*. DOI: 10.48550/arXiv.2311.03658. URL: http://arxiv.org/abs/2311.03658 (visited on 08/04/2025).

Pearce, Michael T. et al. (June 2025). *Bilinear MLPs Enable Weight-Based Mechanistic Interpretability*. DOI: 10.48550/arXiv.2410.08417. URL: http://arxiv.org/abs/2410.08417 (visited on 08/04/2025).

Pedregosa, F., G. Varoquaux, A. Gramfort, et al. (2011). *Scikit-learn: Machine Learning in Python*.

Pfau, Jacob, William Merrill, and Samuel R. Bowman (Apr. 2024). *Let's Think Dot by Dot: Hidden Computation in Transformer Language Models*. DOI: 10.48550/arXiv.2404.15758. URL: http://arxiv.org/abs/2404.15758 (visited on 08/04/2025).

Phan, Long et al. (Apr. 2025). *Humanity's Last Exam*. DOI: 10.48550/arXiv.2501.14249. URL: http://arxiv.org/abs/2501.14249 (visited on 08/04/2025).

Polyak, Boris T. (1964). "Some Methods of Speeding Up the Convergence of Iterative Methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5, pp. 1–17.

Qi, Di, Tong Yang, and Xiangyu Zhang (Jan. 2024). *Slot-Guided Volumetric Object Radiance Fields*. DOI: 10.48550/arXiv.2401.02241. URL: http://arxiv.org/abs/2401.02241 (visited on 07/18/2025).

Qwen Team (Jan. 2025). *Qwen2.5-VL*. Qwen team announcement post. URL: https://qwenlm.github.io/blog/qwen2.5-vl/ (visited on 11/01/2025).

Racah, Evan and Sarath Chandar (July 2020). "Slot Contrastive Networks: A Contrastive Approach for Representing Objects". In: *arXiv:2007.09294 [cs, stat]*. URL: http://arxiv.org/abs/2007.09294 (visited on 01/13/2021).

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI blog*.

Radford, Alec et al. (July 2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *ICML*. Vol. 139. Proceedings of Machine Learning Research. Virtual: PMLR, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html (visited on 12/21/2024).

Rajamanoharan, Senthooran, Arthur Conmy, et al. (Apr. 2024). *Improving Dictionary Learning with Gated Sparse Autoencoders*. DOI: 10.48550/arXiv.2404.16014. URL: http://arxiv.org/abs/2404.16014 (visited on 08/04/2025).

Rajamanoharan, Senthooran, Tom Lieberum, et al. (Aug. 2024). *Jumping Ahead: Improving Reconstruction Fidelity with JumpReLU Sparse Autoencoders*. DOI: 10.48550/arXiv.2407.14435. URL: http://arxiv.org/abs/2407.14435 (visited on 08/04/2025).

Rajeev, Meghana et al. (Mar. 2025). *Cats Confuse Reasoning LLM: Query Agnostic Adversarial Triggers for Reasoning Models*. DOI: 10.48550/arXiv.2503.01781. URL: http://arxiv.org/abs/2503.01781 (visited on 07/04/2025).

Rand, William M. (1971). "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336, pp. 846–850. ISSN: 0162-1459. URL: https://www.jstor.org/stable/2284239 (visited on 07/06/2021).

Reed, Scott, Konrad Zolna, Emilio Parisotto, et al. (May 2022). "A Generalist Agent". In: *arXiv*. URL: http://arxiv.org/abs/2205.06175 (visited on 05/25/2022).

Reynolds, Laria and Kyle McDonell (Feb. 2021). *Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm*. DOI: 10.48550/arXiv.2102.07350. arXiv: 2102.07350 [cs.CL]. URL: http://arxiv.org/abs/2102.07350 (visited on 12/21/2024).

Riegel, Ryan et al. (June 2020). "Logical Neural Networks". In: *arXiv:2006.13155 [cs]*. URL: http://arxiv.org/abs/2006.13155 (visited on 07/14/2021).

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6, pp. 386–408. ISSN: 1939-1471. DOI: 10.1037/h0042519.

Rudin, Cynthia (Sept. 2019). *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. DOI: 10.48550/arXiv.1811.10154. URL: http://arxiv.org/abs/1811.10154 (visited on 07/05/2025).

Rule, Michael E., Timothy O'Leary, and Christopher D. Harvey (2019). "Causes and Consequences of Representational Drift". In: *Current Opinion in Neurobiology* 58, pp. 141–147. DOI: 10.1016/j.conb.2019.08.005.

Rumbelow, Jessica and Matthew Watkins (Feb. 2023). *SolidGoldMagikarp (plus, prompt generation)*. LessWrong. URL: https://www.lesswrong.com/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (Oct. 1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088, pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0 (visited on 07/17/2025).

Rumelhart, David E., James L. McClelland, and PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Vol. 1: Foundations. Cambridge, MA: MIT Press. ISBN: 978-0-262-18120-4. URL: https://mitpress.mit.edu/9780262680530/parallel-distributed-processing/.

Russell, Stuart and Peter Norvig (2020). *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson. ISBN: 9780134610993. URL: http://aima.cs.berkeley.edu/.

Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton (Nov. 2017). "Dynamic Routing Between Capsules". In: *arXiv:1710.09829 [cs]*. URL: http://arxiv.org/abs/1710.09829 (visited on 12/10/2020).

Sadowski, Albert and Jarosław A. Chudziak (June 2025). *Explainable Rule Application via Structured Prompting: A Neural-Symbolic Approach*. DOI: 10.48550/arXiv.2506.16335. URL: http://arxiv.org/abs/2506.16335 (visited on 07/05/2025).

Saharia, Chitwan, William Chan, Saurabh Saxena, et al. (May 2022). "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *arXiv*. URL: http://arxiv.org/abs/2205.11487 (visited on 05/25/2022).

Santoro, Adam et al. (Dec. 2017). "A Simple Neural Network Module for Relational Reasoning". In: *Advances in Neural Information Processing Systems*. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc., pp. 4967–4976. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/e6acf4b0f69f6f6e60e9a815938aa1ff-Abstract.html (visited on 12/21/2024).

Scholkopf, B., Francesco Locatello, S. Bauer, et al. (2021). "Towards Causal Representation Learning". In: URL: https://arxiv.org/abs/2102.11107.

Sclar, Melanie, Yejin Choi, Yulia Tsvetkov, and Alane Suhr (Oct. 2023). *Quantifying Language Model Sensitivity to Spurious Features in Prompt Design*. DOI: 10.48550/arXiv.2310.11324. arXiv: 2310.11324 [cs.CL]. URL: http://arxiv.org/abs/2310.11324 (visited on 12/21/2024).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural machine translation of rare words with subword units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Shanahan, Murray, Kyriacos Nikiforou, Antonia Creswell, et al. (2020). "An Explicitly Relational Neural Network Architecture". In: *ICML*. URL: http://arxiv.org/abs/1905.10307 (visited on 05/28/2019).

Sharkey, Lee, Bilal Chughtai, Joshua Batson, et al. (Jan. 2025). *Open Problems in Mechanistic Interpretability*. DOI: 10.48550/arXiv.2501.16496. URL: http://arxiv.org/abs/2501.16496 (visited on 07/26/2025).

Shazeer, Noam et al. (Jan. 2017). *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. DOI: 10.48550/arXiv.1701.06538. URL: http://arxiv.org/abs/1701.06538 (visited on 08/02/2025).

Shepard, Roger N. (1984). "Ecological Constraints on Internal Representation: Resonant Kinematics of Perceiving, Imagining, Thinking, and Dreaming". In: *Psychological Review* 91.4, pp. 417–447. DOI: 10.1037/0033-295X.91.4.417.

Smith, Samuel L., Andrew Brock, Leonard Berrada, and Soham De (Oct. 2023). *ConvNets Match Vision Transformers at Scale*. DOI: 10.48550/arXiv.2310.16764. URL: http://arxiv.org/abs/2310.16764 (visited on 07/17/2025).

Spies, A. F., W. Edwards, et al. (2024). *Transformers Use Causal World Models in Maze-Solving Tasks*. Accepted to ICLR 2025 Workshop on World Models. arXiv: 2412.11867 [cs.LG]. URL: https://arxiv.org/abs/2412.11867.

Spies, A. F., A. Russo, and M. Shanahan (2022). *Sparse Relational Reasoning with Object-Centric Representations*. Spotlight paper at ICML 2022 DyNN Workshop. arXiv: 2207.07512 [cs.LG]. URL: https://arxiv.org/abs/2207.07512 (visited on 12/21/2024).

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Stutz, David (2022). *Collection of LaTeX resources and examples*. Accessed on 07.01.2025. URL: https://github.com/davidstutz/latex-resources (visited on 07/01/2025).

Su, Jianlin et al. (2024). "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568, p. 127063.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan (June 2017). *Axiomatic Attribution for Deep Networks*. DOI: 10.48550/arXiv.1703.01365. URL: http://arxiv.org/abs/1703.01365 (visited on 08/04/2025).

Sutton, Richard (Mar. 2019). *The Bitter Lesson*. URL: http://www.incompleteideas.net/IncIdeas/BitterLesson.html (visited on 05/25/2022).

Tan, Hao and Mohit Bansal (Dec. 2019). *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*. DOI: 10.48550/arXiv.1908.07490. URL: http://arxiv.org/abs/1908.07490 (visited on 08/04/2025).

Templeton, Adly et al. (2024). *Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet*. URL: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Thrush, Tristan et al. (June 2022). "Winoground: Probing Vision and Language Models for Visio-Linguistic Compositionality". In: *CVPR*. IEEE, pp. 5238–5248. DOI: 10.1109/CVPR52688.2022.00517. URL: https://openaccess.thecvf.com/content/CVPR2022/html/Thrush_Winoground_Probing_Vision_and_Language_Models_for_Visio-Linguistic_Compositionality_CVPR_2022_paper.html (visited on 12/21/2024).

Touvron, Hugo et al. (July 2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. DOI: 10.48550/arXiv.2307.09288. arXiv: 2307.09288 [cs.CL]. URL: http://arxiv.org/abs/2307.09288 (visited on 12/21/2024).

Träuble, Frederik et al. (Mar. 2021). "On Disentangled Representations Learned From Correlated Data". In: *arXiv:2006.07886 [cs, stat]*. URL: http://arxiv.org/abs/2006.07886 (visited on 07/13/2021).

Tripathi, Rishik M. (2020). *Sort-of-CLEVR Reimplementation*. https://github.com/RishikMani/Sort-of-CLEVR.

Turner, Alexander Matt et al. (Oct. 2024). *Steering Language Models With Activation Engineering*. DOI: 10.48550/arXiv.2308.10248. URL: http://arxiv.org/abs/2308.10248 (visited on 08/04/2025).

Van Rossum, Guido and Fred L Drake Jr (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

van Steenkiste, Sjoerd, Klaus Greff, and Jürgen Schmidhuber (June 2019). "A Perspective on Objects and Systematic Generalization in Model-Based RL". In: *arXiv:1906.01035 [cs, stat]*. URL: http://arxiv.org/abs/1906.01035 (visited on 07/14/2021).

van Steenkiste, Sjoerd, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem (Jan. 2020). "Are Disentangled Representations Helpful for Abstract Visual Reasoning?" In: *arXiv:1905.12506 [cs, stat]*. URL: http://arxiv.org/abs/1905.12506 (visited on 07/13/2021).

Vaswani, Ashish et al. (Dec. 2017). "Attention is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., pp. 5998–6008. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 12/21/2024).

Wang, Kevin, Alexandre Variengien, Arthur Conmy, et al. (2022). *Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small*. arXiv: 2211.00593 [cs.LG]. URL: http://arxiv.org/abs/2211.00593.

Wang, Tianlong et al. (Apr. 2025). "Adaptive Activation Steering: A Tuning-Free LLM Truthfulness Improvement Method for Diverse Hallucinations Categories". In: *Proceedings of the ACM on Web Conference 2025*, pp. 2562–2578. DOI: 10.1145/3696410.3714640. URL: http://arxiv.org/abs/2406.00034 (visited on 08/04/2025).

Watters, Nicholas, Loic Matthey, Matko Bosnjak, et al. (Aug. 2019). "COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration". In: *arXiv:1905.09275 [cs]*. URL: http://arxiv.org/abs/1905.09275 (visited on 10/26/2020).

Watters, Nicholas, Loic Matthey, Christopher P. Burgess, and Alexander Lerchner (2019). "Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs". In: *ICLR LLD Workshop*. URL: http://arxiv.org/abs/1901.07017.

Wei, Jason et al. (Dec. 2022). "Chain of Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 24824–24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html (visited on 12/21/2024).

Weis, Marissa A. et al. (June 2020). "Unmasking the Inductive Biases of Unsupervised Object Representations for Video Sequences". In: *arXiv:2006.07034 [cs]*. URL: http://arxiv.org/abs/2006.07034 (visited on 06/13/2021).

Whittington, James C. R. and Rafal Bogacz (Mar. 2019). "Theories of Error Back-Propagation in the Brain". In: *Trends in Cognitive Sciences* 23.3, pp. 235–250. ISSN: 1879-307X. DOI: 10.1016/j.tics.2018.12.005. URL: https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(19)30012-9 (visited on 07/17/2025).

Williams, Ronald J. (May 1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8.3, pp. 229–256. ISSN: 1573-0565. URL: https://doi.org/10.1007/BF00992696 (visited on 07/13/2021).

Wilson, Andrew Gordon (July 2025). *Deep Learning Is Not So Mysterious or Different*. DOI: 10.48550/arXiv.2503.02113. URL: http://arxiv.org/abs/2503.02113 (visited on 07/20/2025).

Wong, Lionel et al. (Dec. 2023). *Learning Adaptive Planning Representations with Natural Language Guidance*. DOI: 10.48550/arXiv.2312.08566. URL: http://arxiv.org/abs/2312.08566 (visited on 07/09/2025).

Wu, Yangzhen et al. (Mar. 2025). *Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for Problem-Solving with Language Models*. DOI: 10.48550/arXiv.2408.00724. URL: http://arxiv.org/abs/2408.00724 (visited on 08/04/2025).

Wu, Yonghui et al. (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Wu, Ziyi et al. (Jan. 2023). *SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models*. DOI: 10.48550/arXiv.2210.05861. URL: http://arxiv.org/abs/2210.05861 (visited on 07/03/2025).

Xiao, Tete et al. (2021). "Early convolutions help transformers see better". In: *Advances in Neural Information Processing Systems*.

Yamagiwa, Hiroaki, Momose Oyama, and Hidetoshi Shimodaira (Dec. 2023). "Discovering Universal Geometry in Embeddings with ICA". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, pp. 4647–4675. DOI: 10.18653/v1/2023.emnlp-main.283. URL: https://aclanthology.org/2023.emnlp-main.283/.

Yang, Xingyi et al. (Mar. 2025). *Mixture of Experts Made Intrinsically Interpretable*. DOI: 10.48550/arXiv.2503.07639. URL: http://arxiv.org/abs/2503.07639 (visited on 08/02/2025).

Yang, Zhun, Adam Ishay, and Joohyung Lee (July 2020). "NeurASP: Embracing Neural Networks into Answer Set Programming". In: *Twenty-Ninth International Joint Conference on Artificial Intelligence*. Vol. 2, pp. 1755–1762. URL: https://www.ijcai.org/proceedings/2020/243 (visited on 07/13/2021).

Yao, Xinhao, Ruifeng Ren, Yun Liao, and Yong Liu (May 2025). *Unveiling the Mechanisms of Explicit CoT Training: How CoT Enhances Reasoning Generalization*. DOI: 10.48550/arXiv.2502.04667. URL: http://arxiv.org/abs/2502.04667 (visited on 08/04/2025).

Yuksekgonul, Mert et al. (July 2023). "When and Why Vision-Language Models Behave Like Bags-of-Words, and What to Do About It?" In: *ICLR*. OpenReview.net. URL: https://openreview.net/forum?id=KRLUvxh8uaX (visited on 12/21/2024).

Zaeemzadeh, Alireza, Nazanin Rahnavard, and Mubarak Shah (Apr. 2020). *Norm-Preservation: Why Residual Networks Can Become Extremely Deep?* DOI: 10.48550/arXiv.1805.07477. URL: http://arxiv.org/abs/1805.07477 (visited on 07/03/2025).

Zaheer, Manzil et al. (Apr. 2018). "Deep Sets". In: *arXiv:1703.06114 [cs, stat]*. URL: http://arxiv.org/abs/1703.06114 (visited on 12/11/2020).

Zaidi, Julian, Jonathan Boilard, Ghyslain Gagnon, and Marc-André Carbonneau (Jan. 2021). "Measuring Disentanglement: A Review of Metrics". In: *arXiv:2012.09276 [cs]*. URL: http://arxiv.org/abs/2012.09276 (visited on 06/07/2021).

Zhang, Yang et al. (Oct. 2024). *FinerCut: Finer-grained Interpretable Layer Pruning for Large Language Models*. DOI: 10.48550/arXiv.2405.18218. URL: http://arxiv.org/abs/2405.18218 (visited on 08/04/2025).

Zimmer, Matthieu et al. (July 2023). *Differentiable Logic Machines*. DOI: 10.48550/arXiv.2102.11529. URL: http://arxiv.org/abs/2102.11529 (visited on 07/09/2025).

Zou, Andy et al. (Dec. 2023). *Universal and Transferable Adversarial Attacks on Aligned Language Models*. DOI: 10.48550/arXiv.2307.15043. URL: http://arxiv.org/abs/2307.15043 (visited on 07/07/2025).

List of Abbreviations

AGI	Artificial General Intelligence. Definitions of what consistute AGI are numerous and contested. For the purposes of Figure 1.2 we defer to the resolution criteria of the Metaculus question "Date of Artificial General Intelligence" (accomplishment of certain scores on various existing benchmarks). 16
ANN	Artificial Neural Network 5, 13, 14, 20–22, 24, 25, 38–40, 43, 105, 106, 119, 151, 154
ARI	Adjusted Rand Index 53, 54, 57, 168
CNN	Convolutional Neural Network 27, 28, 42, 71, 72, 76, 78, 164, 165, 175, 179
CoT	Chain-of-Thought 85
DCI Scores	Disentanglement, Completeness, Informativeness Scores (Eastwood et al. 2018) 45, 46, 54
GRU	Gated Recurrent Unit 28, 42
ICL	In-Context Learning 84
LLM	Large Language Model 116, 154, 217
LSTM	Long Short-Term Memory 28
MHA	Multi-Head Attention (Vaswani et al. 2017) 29
MIG	Mutual Information Gap (R. T. Q. Chen et al. 2019) — subsection 3.3.1 46
MLP	Multi-Layer Perceptron 22, 23, 30, 50–52, 68, 164
MSE	Mean Squared Error 54
OCL	Object-Centric Learning 40, 43, 44, 49, 69, 163, 166
pRDFS	percolation Randomised Depth-First Search, an RDFS maze combined (via OR) with a random percolation maze where adjacent connections occur with probability p, thus introducing cycles and multiple valid paths 115, 121, 123, 209
RDFS	Randomised Depth-First Search, a maze generation algorithm producing acyclic spanning trees with unique shortest paths 114, 115, 121, 123, 209
RL	Reinforcement Learning 40, 166
RNN	Recurrent Neural Network 28, 40
SA	Slot-Attention Module (Locatello, Weissenborn, et al. 2020) 41, 42, 47, 49–53, 55–64, 71, 72, 76, 78, 149, 165, 171, 175, 176, 180
SAE	Sparse Autoencoders (Cunningham et al. 2023) 19, 110–113, 134, 135, 138–145, 150, 152, 153, 217–219, 221–223, 227–229
SB	Spatial Broadcast Decoder (Watters, Matthey, Burgess, et al. 2019) 42, 43, 47, 49–51, 57, 58, 60–63, 72, 165
ST	Spatial Transformer (Jaderberg et al. 2016) 47, 49, 51–53, 56–63, 164
VAE	Variational Autoencoder 47, 54, 166
ViT	Vision Transformer 89

List of Figures

List of Tables

Appendices

Appendix A

Extended Part I Background

A.1 Neuro-symbolic Approaches

It bears mentioning that there is much research presently being carried out on the synthesis of neural networks with formal logic[1], with such work generally referred to as "Neuro-symbolic" (De Raedt et al. 2020; Garcez et al. 2020). Of particular interest are those approaches which carry out differentiable Inductive Logic Programming in conjunction with neural networks, to achieve end-to-end learning of both sub-symbolic representations for perception and logic programs for reasoning. Notable examples of these include DeepProbLog (Manhaeve et al. 2018), NeurASP (Z. Yang et al. 2020) and δ-ILP (Evans et al. 2018), to list but a few.

The appeal of these approaches is two fold. First, researchers can impose structure on the learned logic programs, or explicitly add prior knowledge to the background knowledge of the induction problem, providing an ideal means for knowledge injection — notions of object permanence could be directly encoded. Second, the logic programs learned by such approaches are, whilst not completely transparent, much more amenable to interpretation than latent embeddings in neural network approaches. One may also make arguments concerning the robustness and generalisability of logic programs, but these may not hold for the perception network feeding into the program.

[1] The language of e.g. first-order logic directly expresses the existence of objects and the relationships they obey.

Unfortunately, most existing methods which achieve non-monotonic program induction fail to scale to more complex problems. Whilst this may be partially due to the complexity class of the induction task, the more pertinent issue relates to the highly non-convex structure of the loss landscape when learning discrete programs (Gaunt et al. 2016) — the learning of certain rules may present an irrecoverable error, and it is not clear that gradient descent is well suited to tackle such problems by itself. Whilst work is being done, for example on increasingly "neuralizing" aspects of the logic program, such as rules (Cingillioglu and Alessandra Russo 2022; Riegel et al. 2020), and on non-committal learning strategies[2] (Ellis et al. 2020; Hewitt et al. 2020), we do not consider the existing techniques sufficiently capable to utilise in our own work.

A.2 Object-Centric Learning Models

Below we highlight notable examples of OCL models, implementing different kinds of slot-based architectures, which inform our own work.

Note first that there is an important distinction to be made between models which are designed to operate on static images versus those that operate on videos. The latter typically leverage a dynamics model at the level of object representations, which introduces a number of challenges regarding object interactions. However, assumptions about object permanence can also be leveraged to aid the process of learning good representations. In the case of RL, an agent interacting with its environment could perform causal interventions serving to disentangle its representation, without weak supervision needing to be provided by the engineer (Scholkopf et al. 2021).

Spatial Slots One group of approaches which have excelled at tackling both images and videos where many objects are simultaneously present are those which level structured representations in conjunction with spatial slots. Lin et al. 2020 propose SPACE which breaks up an image into a grid, with each grid cell processed by a number of networks which parameterise the various terms of a factored joint distribution. These terms associate, with each slot, latents: $z_{depth}, z_{scale}, z_{position}, z_{what}$. These are used to decode and transform slot-wise "glimpses" of individual objects via a Spatial

[2] By which we mean, for example: i) wake-sleep as a form of iterative refinement capable of backing out of local optima (G. Hinton et al. 1995), ii) Distributional approaches, either on model output, or by maintaining an ensemble of models, iii) Evolutionary approaches.

Transformer (ST) Network (Jaderberg et al. 2016). A very similar approach is taken by Crawford et al. 2019 with SPAIR, though their model suffers more severely from "box-splitting", whereby the prior on bounding box sizes leads to single objects taking up multiple spatial slots — SPACE partly addresses this through the addition of a "boundary loss" term.

SCALOR presents another spatial slot architecture, extending SQAIR to handle videos with many tens of (simple) objects (Jiang et al. 2019). SQAIR (Kosiorek et al. 2018) is very similar to SPACE and SPAIR, except that it does not model the background, but is able to process videos by instantiating a dynamics model for every slot. SCALOR scales up SQAIR by parallelising object propagation through a mask-similarity based "propose-reject" model, as well as a module for handling dynamic backgrounds. What interests us about these approaches is that they leverage a very highly structured representation for every object, which may prove useful for generalisation, as argued in section 3.3.

Instance Slots Still leveraging a structured representation, but making use of instance slots rather than spatial slots, TBA (Zhen He et al. 2019) is an architecture most similar to the one we investigate. They extract slot representations from an image by passing feature maps from a CNN through a "Tracker Array".

The tracker array consists of a GRU which takes the previous hidden state as well as a weighted combination of feature maps at the current time step. This weighting is performed via a cosine similarity between the previous hidden state[3] and the feature maps, helping to achieve slot alignment. Similarly to the spatial slot methods discussed above, a structured latent representation is then learned on top of the slots by using an MLP to parameterise various distributions, values of which are sampled[4] and used to reconstruct the final image via a ST.

Most generic of all are those approaches which utilise both an unstructured representation and instance slots. Amongst these IODINE (Greff, Kaufman, et al. 2020), Slot-Attention (SA) (detailed in section 3.2.3) and GENESIS-V2 (Engelcke, Jones, et al. 2021) are the key examples. IODINE uses iterative variational inference (IVI) to infer latent variables for each object in an image, with multiple encode-decode steps required to arrive at

[3] with a linear transformation applied.

[4] in spite of sampling, gradients can be propagated using the Straight-Through Gumbel-Softmax estimator from Jang et al. 2017.

the final representation (similar to GENESIS and MONet, discussed below). The primary issue of IODINE is that the IVI procedure is computationally expensive, which has been addressed very recently by Emami et al. 2021. IODINE, MONet and SA use a Spatial Broadcast Decoder (SB) (Watters, Matthey, Burgess, et al. 2019) (detailed in section 3.2.3) across slots. SA is simpler, faster to train and more performant than both MONet and IODINE, however the recently released GENESIS-V2 outperforms all three approaches.

GENESIS-V2 attempts to address two key shortcomings of some previous works. Firstly, all approaches using a form of iterative refinement have thus far required an a priori choice on the number of slots to be initialised[5]. Second, they argue that methods using sequential slots are prone to imposing meaningless ordering over objects which also affects the gradient signal each object receives (the first object detected gets the weakest signal). To address these issues they combine GENESIS (Engelcke, Kosiorek, et al. 2020) (which performs sequential processing in latent space rather than at the level of CNNs like MONet) with Slot Attention — formalising the problem as a VAE with an autoregressive prior, and using an attention mechanism to retrieve unordered object representations. Instead of performing iterative attention, they propose a differentiable clustering mechanism which allows an arbitrary number of objects to be captured and does not require multiple steps. Their method outperforms MONet and Slot Attention, but is also limited to images.

Sequential Slots MONet provides an example of a sequential slot model with unstructured representations (Burgess, Matthey, et al. 2019). MONet has two key components: i) an "Attention Network" which produces a mask determining which *remaining* sub-region of an image will be processed at a given step, ii) a "Component VAE" which is tasked with reconstructing the masked image (i.e. object) — its latent vector is the "slot" for that object. The number of steps performed by the model determine the number of slots. Unlike many other OCL papers, the authors perform latent traversals to show that their learned representations are somewhat disentangled, as expected from the direct use of VAEs.

MONet has been extended to work on videos by both Weis et al. 2020 and Creswell, Kabra, et al. 2021, though Creswell's approach may be appli-

[5] Though, as outlined at the beginning of this section, we believe this is not an issue if top-down information can be used to select which objects are represented.

cable to any slot-based model. Weis et al. 2020 introduce ViMON which adds a recurrent component on top of MONet's encoder, as well as a loss term pushing slot representations at subsequent timesteps to be linearly predictable from their immediate predecessors (encouraging alignment). Creswell, Kabra, et al. 2021 propose OAT which utilises a module known as AlignNet (Creswell, Nikiforou, et al. 2020) on top of the slots produced by MONet. The module uses a transformer to permute slots from the current time-step such as to match slots from the previous time-step which have been passed through a dynamics module. They also add a slot-wise memory component to handle occluded objects in their "Memory AlignNet".

It is worth noting that an aligned slot representation enables one to train dynamics models in an RL setting with relative ease. There are other approaches such as contrastive learning over representations which have been aligned through a matching procedure (Kipf et al. 2020), such as a Hungarian Algorithm, or using permutation invariant models such as DeepSets (Zaheer et al. 2018) to aggregate slots (Löwe et al. 2020). However, contrastive learning approaches are typically costly to train and have shown only limited success in slot based models so far.

Category Slots The primary example of category slots is provided by Capsule Networks (G. E. Hinton et al. 2011). In Capsule Networks slots ("capsules") specialise to represent specific object categories. Part-whole hierarchies are then modelled (in principle) through a "routing by agreement" procedure, whereby capsules at a low level in the hierarchy predict the potential activations of capsules at the next level (Sabour et al. 2017). Each such predicted activation corresponds to an object which the lower level capsule believes it could be representing part of. The maximally agreed upon prediction of neighbouring capsules will then correspond to the actual object which is modelled at the next level, and thus the signal passed on is determined by the extent of agreement between lower level capsules. Unfortunately, training capsule networks is computationally expensive which has hampered their uptake, however, recent work has begun to address these issues (Mazzia et al. 2021).

Appendix B

Chapter 4 Appendices

B.1 Data

B.1.1 Spriteworld

To facilitate the calculation of the ARI, binary masks were added to the SpriteWorld dataset, as shown in Figure B.1. Empty masks were provided up to some maximum number of entities specified when the datasets were generated (in our case 1-4 objects).

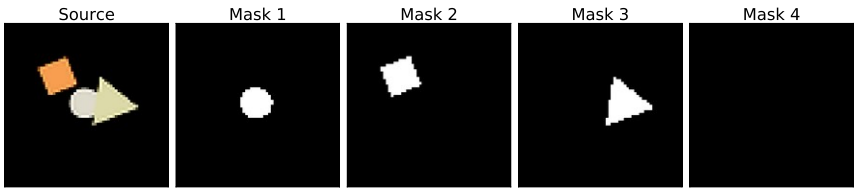


Figure B.1: Binary object masks for SpriteWorld dataset. To facilitate ARI calculation, mask generation functionality was added to the Spriteworld dataset generator. Each mask corresponds to a single object in the scene, with entries containing empty masks to pad up to the maximum number of objects (1-4 in our experiments).

Note that in SpriteWorld there are no interactions between objects.

B.2 Slot-Attention Encoder

B.2.1 Object property encoding in slots

We investigate the extent to which ground truth factors of variations are well captured by the Slot Attention latents. This is challenging, as when utilising slot-based representations, multiple latent vectors encode information about the image, and it is not always known which latent corresponds to which feature. To address this we use the model's attention maps to perform slot-object pairing. In particular, we use openCV to find contours in the normalised attention masks, and then apply an adaptive threshold to establish whether there is: i) exactly one contour in the map, and ii) whether the contour area is less than a fixed fraction of the total image (a heuristic for discarding background slots). The centre position of the resulting contour is then used to match the slot to a ground truth object, the properties of which are assigned as labels to the corresponding slot.

After this labelling procedure has been performed, we collect 20,000 slots with associated object labels. Following Watters, Matthey, Burgess, et al. (2019), we then train auxiliary models (Gradient Boosted Regressors) to predict ground truth factors of variation, such as position, from these slots. By querying the feature importances of the resulting models, and assessing their prediction accuracies on held-out slots, we are able to assess the representation of objects within slots.

The feature importances for multiple generative factors are then combined into a matrix, as shown in Fig. B.2. Using this matrix, we compute the completeness and disentanglement of the slot representations, shown in Table B.1. These are defined as: **Completeness** — the extent to which a single latent encodes information about only a single factor of variation, and **Disentanglement** — the extent to which all information about a single factor of variation is captured in only one latent.

Table B.1: Latent properties obtained using the slot-object matching procedure outlined in subsection B.2.1. The completeness and disentanglement metrics are computed following the definitions of Eastwood et al. (2018), and the Gradient-boosted Regressors were used as auxiliary models. Random performance on Size prediction on Relations Game would equal 33%.

		Latent Structure Measures		Auxiliary Model Accuracy				
Model	Dataset	Completeness	Disentanglement	x	y	shape	color	size
Small SA	Relations Game	0.33	0.55	77	75	-	-	51
	Sort-of-CLEVR	0.40	0.62	93	92	90	92	-
Big SA	Relations Game	0.43	0.62	84	86	-	-	69
	Sort-of-CLEVR	0.49	0.82	95	95	96	91	-

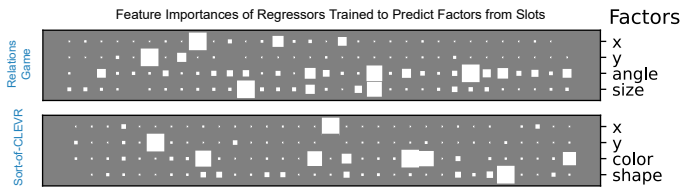


Figure B.2: Hinton diagram showing disentanglement properties of the Slot Attention latents. The matrix shows feature importances where each row corresponds to a latent dimension and each column to a factor of variation. Completeness corresponds to the average one-hotness of the columns (each factor captured by few latents), and disentanglement corresponds to the average one-hotness of the rows (each latent captures few factors). Position information is strongly disentangled, as expected when using a spatial-broadcast decoder for pre-training Slot Attention Watters, Matthey, Burgess, et al. 2019.

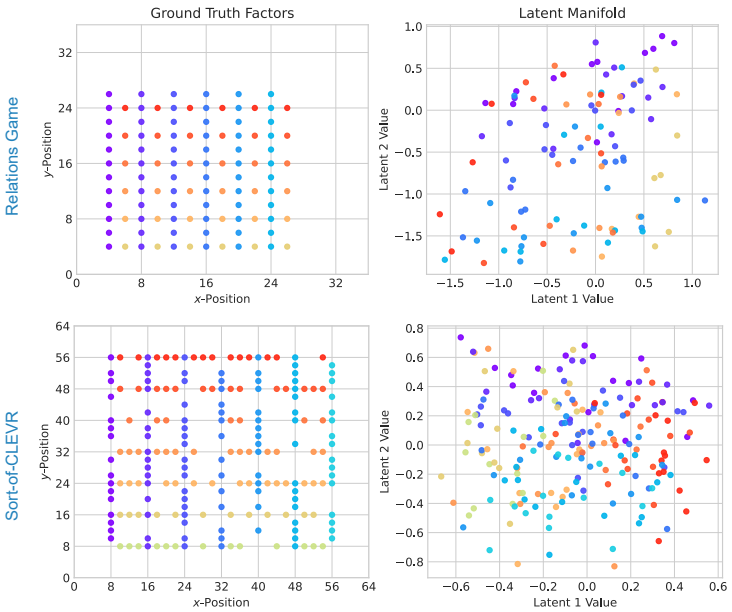


Figure B.3: Learned vs. ground-truth position manifolds for the small Slot Attention model. The dimensions deemed to most saliently encode x and y position are compared to their ground-truth equivalents, revealing that SA learns an approximately linear mapping for position information (up to a rotation and scaling). The apparent noisiness of the latent manifolds is primarily due to imperfections in the slot-object matching procedure. Missing points correspond to cases where no matching slot was found in the subset of the test-set considered.

B.2.2 Temporal Slot Alignment

We conducted a preliminary experiment to assess whether slot-object assignments could be maintained consistently across time steps without explicit temporal modelling. This investigation used the Bouncing Balls dataset, where objects undergo simple translational motion.

Method

We trained a standard Slot Attention model Locatello, Weissenborn, et al. 2020 on individual frames (non-sequential training), then evaluated temporal consistency by propagating slot representations across consecutive frames. Specifically, slot initialisation at time t was modified as:

$$\mathbf{z}_k^{(t)} = \begin{cases} \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)), & \text{if } t = 0 \\ \mathbf{z}_k^{(t-1)}, & \text{if } t > 0 \end{cases} \tag{B.1}$$

where $\mathbf{z}_k^{(t)}$ denotes the k-th slot at time t, and $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are learned parameters.

Figure B.4 illustrates this recurrent initialisation scheme, where slot representations from the previous time step serve as initialisation for the current step.

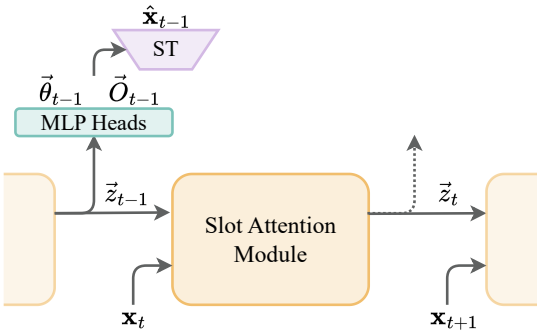


Figure B.4: Recurrent slot attention architecture. Slot representations $\vec{z}_{t-1}$ from the previous time step initialise the slot attention module at time t, enabling temporal consistency in slot-object assignments.

Evaluation Metric

To quantify alignment stability, we measured the fraction of slots that maintained consistent object assignments across consecutive frames:

$$\text{Alignment}(T) = \frac{1}{K(T-1)} \sum_{t=1}^{T-1} \sum_{k=1}^{K} \mathbb{1}[o_k^{(t)} = o_k^{(t-1)}] \tag{B.2}$$

where $o_k^{(t)}$ denotes the object assigned to slot k at time t, K is the number of slots, and $\mathbb{1}[\cdot]$ is the indicator function. Note that random assignment would yield an expected alignment of > 0.

To detect failure modes, we relaxed the exclusive matching constraint, allowing multiple slots to bind to the same object. This helps identify when the slot attention mechanism destabilises.

Results

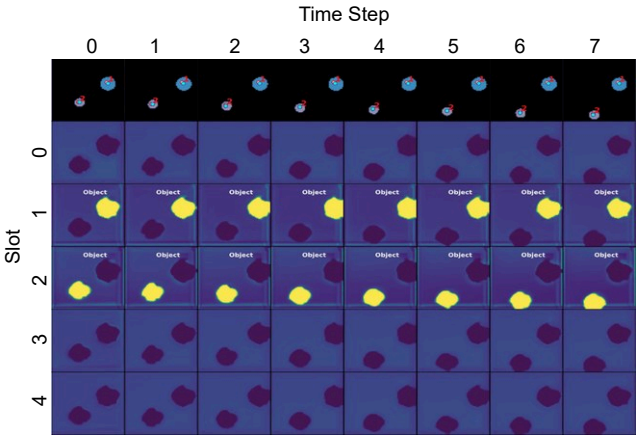
The slot-object assignment procedure maintained consistent mappings in **98.4**% of consecutive frame pairs. Figure B.5 demonstrates successful tracking across 8 time steps.

Limitations

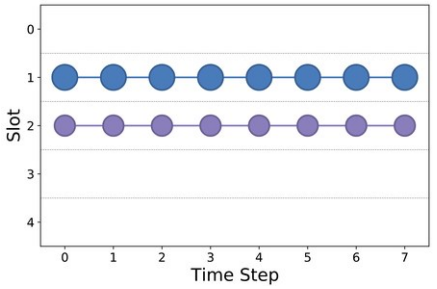
This approach has clear limitations:

- Simple motion patterns in Bouncing Balls may not generalise to complex real-world scenarios
- Occlusions and rapid motion could disrupt slot continuity
- The method lacks explicit temporal modelling present in dedicated video architectures

Despite these limitations, the high alignment rate (98.4%) suggests that slot representations capture sufficient object-centric information to enable tracking through simple propagation, at least in controlled settings.



(a) Slot attention masks across 8 consecutive time steps. Each row represents a slot, consistently tracking the same object (indicated by colour) throughout the sequence.



(b) Abstract representation of slot-object assignments over time, showing consistency.

Figure B.5: Temporal consistency of slot-object assignments using recurrent initialisation. The slot attention mechanism maintains stable object tracking without explicit temporal supervision.

Appendix C

Chapter 5 Appendices

C.1 Comparison between low and high dimensional Slot-Attention

To fairly compare the slot-attention based models against ones with a CNN encoder, it was necessary to train a version of the model with a latent feature space of lower resolution than the input images. This comparison helps isolate the effect of representation dimensionality from architectural differences. Figure C.1 shows the qualitative difference in model behaviour. Essentially, both models capture relevant information, and differences in quantitative measures likely reflect shortcomings of the slot-object assignment procedure.

It is worth noting that the inaccuracy of the assignment procedure can be seen both in the manifolds in Figure B.3 as well as the mask fitting examples in Figure C.1. This inaccuracy is more dramatic for the small SA model on the relations game, as contours may be very crude, and thus nearby objects may be matched to incorrect slots. We suspect this accounts for the majority of the difference in *measured disentanglement* between the two models' latent spaces.

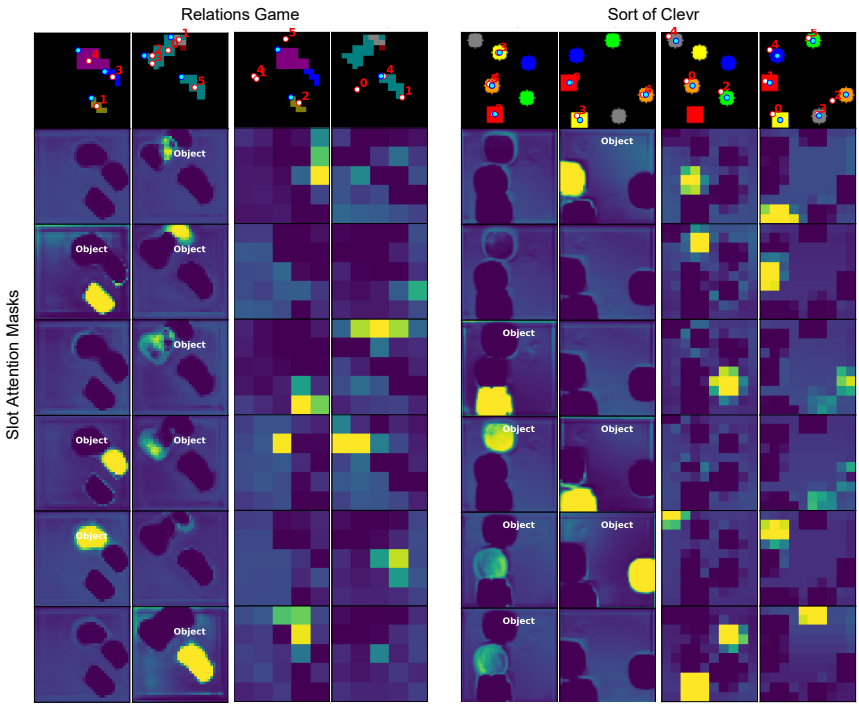


Figure C.1: Comparison of large and small Slot Attention models across different datasets. The top row shows the input images, and the remaining rows show the attention masks of the SA module for different slots. The first two columns for each dataset correspond to the full-size (latent resolution matching image resolution) SA models, while the latter columns show the reduced-resolution versions. The slot-object matching procedure is visualised: masks with detected contiguous contours are labelled with "object", and the red numbers and positions show the centroids of the mask contours with corresponding slot indices. Blue points denote ground-truth object positions for comparison

C.2 Failure to Capture Objects

We observe that slot attention may fail to capture objects separately when they are close, and have similar colours. This under-segmentation can make it impossible for the downstream networks to successfully perform tasks.

	Relations Game	Sort of Clevr
% Objects Missed	13	24

Table C.1: We approximated the frequency with which Slot Attention fails to capture an object in any of its slots. This was done by using the slot-object assignment procedure used in all experiments, and comparing against ground-truth labels.

C.3 Additional Results

Here we provide complete results for the sparsity sweeps carried out on the relations game curricula, as shown in Figs. C.2 to C.4. We also report the "Sparsity Gap" — defined as the average drop in performance across pruning.

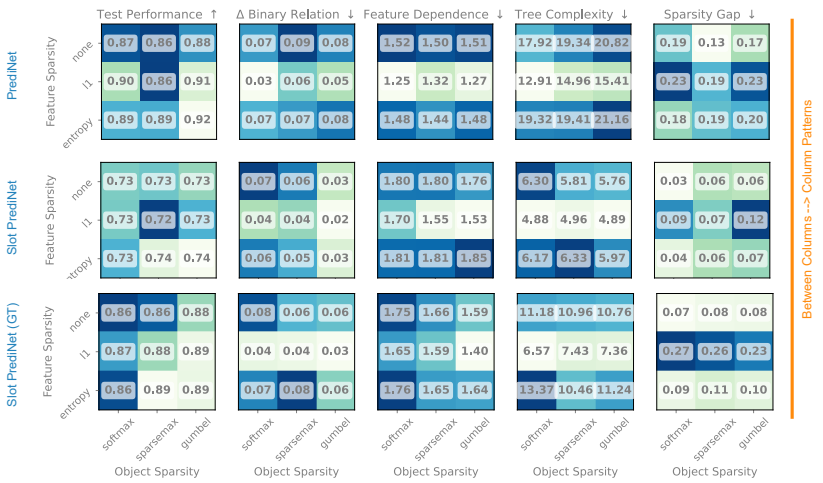


Figure C.2: Sparsity sweeps on the Between Columns →Column Patterns curriculum.

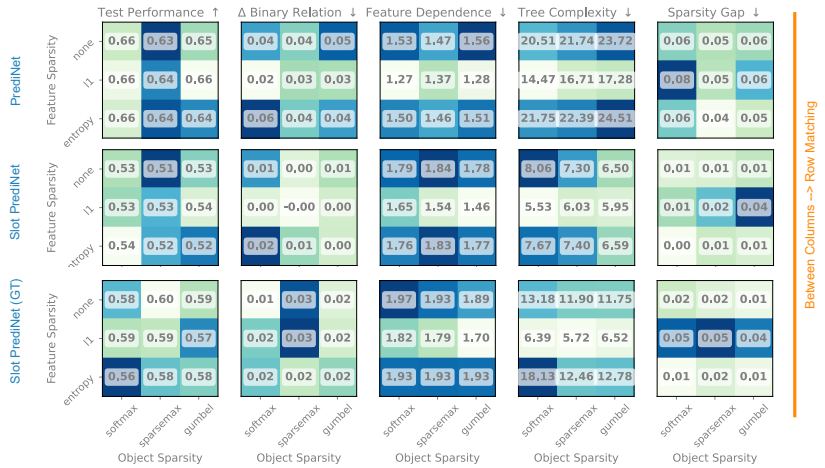


Figure C.3: Sparsity sweeps on the Between Columns →Row Matching curriculum.

Figure C.4: Sparsity sweeps on the Row Matching →Row Patterns curriculum.

C.3.1 Additional Discussion

Whilst we have seen that there are few general conclusions which can be drawn about the properties of the relational reasoning architectures we tested, there are points to be made which apply to the PrediNet, Slot PrediNet and RelationNet individually.

First we note that the sparsity gap metric mostly correlates with the test performance — a model which performs very poorly prior to pruning or auxiliary model replacement will show little drop in performance during pruning. As such, we omitted discussion of the sparsity gap in the main paper. In fact, the same is true to some extent of all sparsity metrics considered. As such, we focussed our discussion on the BC→CP curriculum, in which all models performed well.

Limitations of one-step relation application: Both the PrediNet and RelationNet are unable to re-apply the same rule multiple times (i.e. recursion), and so cannot straightforwardly handle cases where the same relationship is required many times. The PrediNet is able to re-learn the same relationship across heads, but will only be inclined to do so if it is trained on a task demanding this in the first instance. This explains why the PrediNet is able to perform reasonably well on the RM→RP curriculum, but performs poorly on BC→RM.

Lacking spatial inductive bias: A downside of using Slot Attention rather than CNNs is that, whilst the latent representation disentangles x and y position, models will not learn relations which are translation or rotation invariant unless trained to do so. It is primarily[1] for this reason that the Slot PrediNet, even when provided with ground-truth object representations, performs worse than the PrediNet on the BC→RM curriculum.

Lack of a null slot in the PrediNet: We also observed that the PrediNet performs worse than the RelationNet on the non-relational Sort-of-CLEVR tasks. The simplest explanation for this is that the PrediNet's relation learning mechanism has two steps which can discard salient information about objects; namely, the attention mechanism, and the subtraction-based comparator. As each relation head applies the same set of weights to both of its aggregated input slots, it is not trivial for a single head to implement the identity operation. This could be resolved most easily by adding a null slot[2] to the PrediNet's inputs (similar to A. Goyal et al. 2021).

[1] The fact that the CNN in the PrediNet is able to learn relational encodings directly also contributes, but based on differences in BC→CP, it is unlikely that this is the leading order cause of differences on BC→RM.

C.4 Pretraining Slot Attention

To pretrain Slot Attention we used the same autoencoding setup as Kipf et al. (2020), performing unsupervised reconstruction with an L2 loss.

C.5 Implementation Details

Architectural details are provided in Table C.2. When ground truth inputs are provided to the Slot PrediNet and RelationNet, these are constructed into vectors by:

1. Mapping all object properties to the interval $[-1, 1]$
2. Concatenating (in some fixed order) these properties for a given object, to form a slot
3. Stacking the set of slots for all objects present in the scene
4. Appending empty slots (zeros), so that there are six slots in total (never needed for Sort-of-CLEVR)

Additionally, a learned linear layer with 34 units is applied across these slots, and considered part of the "Encoder".

C.5.1 Libraries

Decision Trees were implemented using Scikit-learn Pedregosa et al. 2011. For neural network training and experimentation we used Jax Bradbury et al. 2018, Haiku Hennigan et al. 2020, Optax Hessel et al. 2020 and Tensorflow Martín Abadi et al. 2015. Decision trees were pruned using Minimal Cost-Complexity Breiman et al. 1984.

[2] As there are six slots in our SA model, and six objects in all Sort-of-CLEVR scenes, there were no consistently free slots by default.

Table C.2: Architectural Details for the architectures used. The spatial-broadcast decoder architecture used for slot-attention pre-training was identical to that of Locatello, Weissenborn, et al. (2020).

Model	Component(s)	Architectural Details
Enconder	Conv2D	32 Channels, (12, 12) Kernel Size, (6, 6) Stride, (1, 1) Padding, ReLU
	Learned Position Embedding	Linear Layers: [32]
	Linear Layers	LayerNorm, Linear Layers: [32, ReLU, 32]
Slot Attention	Slots	6
	Latent Sizes	32 (K,Q,V Projections do not use bias)
	Latent Resolution	Relations Game (5, 5); Sort-of-CLEVR (12, 12)
	Attention Iterations	3 at train and test
PrediNet	Number of Heads	16
	Relations per Head	8
RelationNet	g_{mlp}	Linear Layers: [2000, ReLU, 2000, ReLU, 2000, ReLU, 2000, ReLU]
	f_{mlp}	Linear Layers: [2000, ReLU, 1000, ReLU, 500, ReLU, 100, ReLU]
Task MLP	Relations Game	Linear Layers: [8, ReLU, 2, Softmax]
	Sort-of-CLEVR	Linear Layers: [16, ReLU, 10, Softmax]

Table C.3: CNN encoder for all experiments.

Type	Size/Channels	Activation	Comment
Conv 5×5	32	ReLU	stride: 1
Conv 5×5	32	ReLU	stride: 1
Conv 5×5	32	ReLU	stride: 1
Conv 5×5	32	ReLU	stride: 1
Position Embedding	—	—	See Section E.2
Flatten	axis: [B,W×H,32]	—	flatten x, y pos.
Layer Norm	—	—	—
MLP (per location)	32	ReLU	—
MLP (per location)	32	—	—

Table C.4: Spatial Broadcast Decoder used with Slot Attention.

Type	Size/Channels	Activation	Comment
Spatial Broadcast	width × height	—	—
Position Embedding	—	—	See Section E.2
Conv 5×5	32	ReLU	stride: 1
Conv 5×5	32	ReLU	stride: 1
Conv 5×5	32	ReLU	stride: 1
Conv 3×3	4	—	stride: 1
Split Channels	RGB (3), alpha mask (1)	Softmax (on alpha masks)	—
Recombine Slots	—	—	—

Table C.5: Latent transformation MLPs used with Spatial Transformer.

Type	Size/Channels	Activation	Comment
Linear	64	ReLU	Dropout
Linear	32	ReLU	—
Linear	6	—	$\vec{\theta}$ (to ST)
Linear	64	ReLU	—
Linear	26	ReLU	$\vec{O}$ (to Decoder)

Table C.6: Decoder used with Spatial Transformer.

Type	Size/Channels	Activation	Comment
MLP	—	—	See Table C.5
Flatten	axis: [B×K,W, H, C]	—	Combine batch and slot dim
Conv 5×5	64	ReLU & BatchNorm	stride: 1, kernel: 4
Conv 5×5	32	ReLU & BatchNorm	stride: 1, kernel: 4
Conv 5×5	16	ReLU & BatchNorm	stride: 2, kernel: 3
Conv 3×3	4	—	stride: 2, kernel: 4
Spatial Transformer	—	—	—
Split Channels	RGB (3), alpha mask (1)	Softmax (on alpha masks)	—
Recombine Slots	—	—	—

Appendix D

Interlude Appendices

D.1 Final Prompts

Below we present the full prompts used for each experimental condition in their optimal configurations, as determined by the ablation studies summarised in Table D.4. For the Relations Game visual reasoning task, we show the "between" task as an exemplar; the other three tasks (same, occurs, xoccurs) follow an identical template structure with the task description and examples substituted accordingly.

D.1.1 Textual Reasoning

Relations Game

RelationsGame Textual Question Answering Prompt Example

TASK

```
You are an AI agent tasked with performing a reasoning task in
the "relations game", as presented in the PrediNet paper by
Shanahan. You will be provided with the description of a partially
filled 3x3 grid containing objects. Your goal will be to perform
the reasoning task described below.

All objects have been pre-labelled with character assignments,
which capture whether they have identical shape and color. For
example, two objects labelled A and A have the same shape and
color, whereas two objects labelled A and B may differ in color,
shape or both.

You will be performing the "same" task --- Identify whether all
shapes are identical in color and shape.
```

FORMAT

```
Provide your answer between <answer>...</answer> tags.

The grid is formatted as follows:
a. A 3x3 grid using '+', '-', and '|' characters
b. Empty cells are represented with '.'
c. Objects are represented with their corresponding labels
   (A, B, C, etc.)
d. Objects with the same label have identical shape and color

Answer format:
<answer>True</answer> or <answer>False</answer>
```

EXAMPLES

```
<example>
Object Grid:
+---+---+---+
| . | . | . |
+---+---+---+
| . | A | . |
+---+---+---+
| A | . | . |
+---+---+---+
<question>Are all shapes identical in color and shape?</question>
<answer>True</answer>
</example>
...[we provide 3 examples]
```

TASK INSTANCE

```
Now complete this task for the grid provided below, and provide
your answer between <answer>...</answer> tags.

--------------------------------
Object Grid:
+---+---+---+
| . | A | . |
+---+---+---+
| . | . | B |
+---+---+---+
| . | . | . |
+---+---+---+
<question>Are all shapes identical in color and shape?</question>
```

CLEVR

CLEVR Textual Question Answering Prompt Example

TASK

```
You are an AI agent tasked with providing a question answering
task from the CLEVR dataset. You will be provided with an accurate
textual description of an image that contains various simple
shapes arranged on a table. Your goal is to accurately answer the
question with the scene description provided.
```

FORMAT

```
You will be provided with a description and task in the format:
A list of object details, formatted as follows:
a. Each object on one line in the form "Object [unique number]:
   A [shape] [color] [material] [shape]. Positioned at
   Row [row]/20, Column [column]/20,
   Z-order: [z-order]/[num_objects]"
b. Assume Row 1 is the first row, and Column 1 is the first
   column (i.e. top-left of the image).
c. The object with Z-order 1/N is at the "front" of the scene,
   whilst the one with Z-order N/N is at the back.

Provide your final answer between <answer> tags.
```

EXAMPLES

```
Here is an example of a scene obeying the correct formatting
rules for a valid CLEVR scene, along with a question and answer:

<scene>
Object 1: A Small Red Rubber Sphere. Located at Row 9/20,
  Column 4/20, Z-order: 2/3
Object 2: A Large Brown Rubber Sphere. Located at Row 11/20,
  Column 5/20, Z-order: 1/3
Object 3: A Small Purple Rubber Cylinder. Located at Row 6/20,
  Column 10/20, Z-order: 3/6
</scene>
<question>What is the shape of the small purple object?</question>
<answer>cylinder</answer>
```

TASK INSTANCE

```
Now proceed to answer the question for the scene provided, making
sure you provide your final answer between <answer> tags.

--------------------------------
<scene>
Object 1: A Large Brown Rubber Cylinder. Located at Row 8/20,
  Column 4/20, Z-order: 3/5
Object 2: A Large Gray Rubber Cube. Located at Row 11/20,
  Column 8/20, Z-order: 1/5
Object 3: A Small Green Rubber Cylinder. Located at Row 7/20,
  Column 6/20, Z-order: 4/5
Object 4: A Large Purple Metal Sphere. Located at Row 6/20,
  Column 11/20, Z-order: 5/5
Object 5: A Small Gray Metal Cube. Located at Row 12/20,
  Column 14/20, Z-order: 2/5
</scene>
<question>Is there a big brown object of the same shape as the
  green thing?</question>
```

D.1.2 Scene Description

Relations Game

The Relations Game scene description prompt is shown in the main chapter (chapter 6). It uses the same format as the textual reasoning prompt, but with the image provided instead of the textual grid, and with the output format specified as an object list. We reproduce it in subsection D.2.2 below.

CLEVR

CLEVR Scene Description Prompt (Optimal Configuration)

TASK

```
You are an AI agent tasked with describing a scene from the
CLEVR dataset. You will be provided with an image containing
various simple shapes arranged on a table. Your goal is to
accurately describe the image provided.
```

FORMAT

```
You are required to describe the scene in the following output
format: A list of object details, as shown below, with your
answer provided between <answer> tags. You should assign a
unique number to each object.

To be clear, the list should be formatted as follows:
   a. Describe each object on one line in the form
      "Object [unique number]: A [size] [color] [material]
      [shape]. Positioned at (x=[x], y=[y], z=[z])"
   b. Assume x=0, y=0 is the centre of the scene, with the
      positive x, y quadrant at the bottom right of the image,
      and the negative x, y quadrant at the top left.
   c. Assume the z coordinate is relative to the centre of the
      object along the axis normal to the surface of the table,
      which is at z=0.

Here is an example of an answer obeying the correct formatting
rules for a valid CLEVR scene:
<answer>
Object 1: A Small Red Rubber Sphere.
    Positioned at (x=-1.37, y=-2.16, z=0.35)
Object 2: A Large Brown Rubber Sphere.
    Positioned at (x=0.47, y=-2.67, z=0.70)
Object 3: A Small Purple Rubber Cylinder.
    Positioned at (x=-1.67, y=1.53, z=0.35)
</answer>
```

TASK INSTANCE

```
Now proceed to describe the scene provided.
```

D.1.3 End-to-End Visual Reasoning

CLEVR

CLEVR End-to-End Visual Reasoning Prompt (Optimal Configuration)

TASK

```
You are an AI agent tasked with answering questions about images
from the CLEVR dataset. You will be provided with an image
containing various simple shapes arranged on a table and a
question about the scene. Your goal is to accurately analyze the
visual information and answer the question.

In particular, you will be provided with an image and a question
about the scene, formatted as follows:
   a. An image containing various simple shapes (spheres, cubes,
      cylinders) arranged on a table
   b. Each object has properties: color (red, blue, green,
      yellow, gray, purple, cyan, brown), material (metal,
      rubber), size (small, large), and shape (sphere, cube,
      cylinder)
   c. A question asking about the objects in the scene
```

EXAMPLES

```
<examples>
<example>
[An image showing 5 objects: a large brown rubber cylinder, a
large gray rubber cube, a small green rubber cylinder, a large
purple metal sphere, and a small gray metal cube arranged on a
table]
<question>How many rubber objects are there?</question>

Analysis: Let me identify each object and its material:
- Object 1: Large brown rubber cylinder (rubber material)
- Object 2: Large gray rubber cube (rubber material)
- Object 3: Small green rubber cylinder (rubber material)
- Object 4: Large purple metal sphere (metal material)
- Object 5: Small gray metal cube (metal material)

Now counting rubber objects: Objects 1, 2, and 3 are made of
rubber.

<answer>3</answer>
</example>
...[we provide 3 examples with worked reasoning in total]
</examples>
```

REASONING

```
Please follow these detailed instructions to solve the task:
1. Carefully analyze the image, noting the position, shape,
   material, color, and size of each object in the scene.
2. Identify the objects that are relevant to the question
   being asked.
3. Based on the visual information and relevant objects, perform
   the reasoning task outlined in the question.
4. Double-check your visual analysis and reasoning before
   providing your final answer.

Now proceed to answer the question for the scene provided,
making sure you provide your final answer between <answer> tags.
```

Relations Game

Relations Game End-to-End Visual Reasoning Prompt (Optimal Configuration, "between" task)

TASK

```
You are an AI agent tasked with performing a reasoning task in
the "relations game", as presented in the PrediNet paper by
Shanahan. You will be provided with an image of a partially
filled 3x3 grid containing objects. Your goal will be to perform
the reasoning task described below by analyzing the visual
information.

An object is considered identical to another if its shape,
rotation and color are the same; if any of these differ, you may
consider the objects distinct. All objects will be tetrominoes
which are not particularly amenable to linguistic explanation
--- as such, you will benefit from assigning labels to objects
when reasoning.

You will be performing the "between" task --- Identify whether
two identical shapes are on either side of a distinct shape.
```

EXAMPLES

```
<examples>
<example>
[An image showing a 3x3 grid with tetromino pieces: a red
L-shaped tetromino [A] in row 1 column 1; a blue T-shaped
tetromino [B] in row 1 column 2; and another red L-shaped
tetromino [A] in row 1 column 3]
<question>Are two identical shapes on either side of a distinct
shape?</question>
<answer>True</answer>
</example>
<example>
[An image showing a 3x3 grid with tetromino pieces: a green
Z-shaped tetromino [A] in row 1 column 1; a yellow I-shaped
tetromino [B] in row 2 column 2; and a purple S-shaped
tetromino [C] in row 3 column 3]
<question>Are two identical shapes on either side of a distinct
shape?</question>
<answer>False</answer>
</example>
<example>
[An image showing a 3x3 grid with tetromino pieces: a blue
T-shaped tetromino [A] in row 1 column 2; another blue T-shaped
tetromino [A] in row 2 column 2; and a red L-shaped tetromino
[B] in row 3 column 2]
<question>Are two identical shapes on either side of a distinct
shape?</question>
<answer>False</answer>
</example>
</examples>
```

TASK INSTANCE

```
Now complete this task for the image provided below, and provide
your answer between <answer>...</answer> tags.
```

D.2 Prompt Variants

Our ablation studies vary several dimensions of the prompt design. Rather than reproducing the full prompts for every configuration, we summarise the key dimensions of variation below.

D.2.1 Ablation Dimensions

Table D.1 summarises the dimensions varied across our ablation experiments. Each experiment type varies a subset of these dimensions, resulting in a combinatorial grid of configurations tested per model.

Dimension	Options	Applies To
Output format	Object list vs. ASCII grid (RG) Properties vs. Colloquial (CLEVR)	Scene description, textual reasoning
Coordinate format	Absolute (x, y, z) vs. grid $(\text{Row}/20, \text{Col}/20)$	Scene description (CLEVR)
In-context examples	With (3 examples) vs. without	All experiments
Reasoning prompt	With step-by-step instructions vs. without	All experiments
Example style	Direct (answer only) vs. Descriptive (worked reasoning)	End-to-end
Image scale	$\{1\times, 2\times\}$ (CLEVR); $\{1\times, 4\times, 8\times, 16\times\}$ (Relations Game)	Scene description
Grid lines	Present vs. absent on Relations Game images	Scene description
Background colour	Original vs. flipped on Relations Game images	Scene description

Table D.1: Summary of prompt and image configuration dimensions varied in our ablation experiments.

D.2.2 Output Format Variants

For the Relations Game scene description task, we test two output formats. The **ASCII grid** format asks the model to produce a 3×3 character grid:

```
<answer>
+---+---+---+
| A | B | A |
+---+---+---+
| . | . | . |
+---+---+---+
| . | . | . |
+---+---+---+
</answer>
```

The **object list** format asks for a structured enumeration:

```
<answer>
- Object [A] in Row [1], Column [2]
- Object [A] in Row [3], Column [1]
- Object [B] in Row [3], Column [3]
</answer>
```

For CLEVR, the two description formats (Properties and Colloquial) are shown in Figure D.4.

D.2.3 Relations Game Task Descriptions

The Relations Game comprises four relational reasoning tasks, each requiring a True/False response:

Task	Description
`same`	Identify whether all shapes are identical in colour and shape
`between`	Identify whether two identical shapes are on either side of a distinct shape
`occurs`	Identify whether the shape on the top row is identical to a shape on the bottom row, both in colour and shape
`xoccurs`	Holds iff the object in the top row occurs in the bottom row and the other two objects in the bottom row are different

Table D.2: The four Relations Game reasoning tasks. Each task is presented with the same prompt template; only the task description and question text are substituted.

D.2.4 Example Style Comparison

In end-to-end visual reasoning, we test two styles of in-context examples:

- **Direct examples** provide only the question and answer, e.g.:

```
<question>Are two identical shapes on either side of a distinct
   shape?</question>
<answer>True</answer>
```

- **Descriptive examples** include worked reasoning before the answer, e.g.:

```
Analysis: Let me identify each object and its material:
- Object 1: Large brown rubber cylinder (rubber)
- Object 2: Large gray rubber cube (rubber)
- Object 3: Small green rubber cylinder (rubber)
Now counting rubber objects: 3 are rubber.
<answer>3</answer>
```

As noted in Table D.4, descriptive examples were optimal for CLEVR and direct examples for the Relations Game.

D.3 Ablations

Two notes on the figures in this section before the per-task results. First, the number of prompt configurations (and hence the number of points per box or cell in the heatmaps and boxplots below) differs across (dataset, task) cells because each ablation sweep varied a different subset of the dimensions listed in Table D.1. Per model, the CLEVR scene description sweep yields 32 configurations, the Relations Game scene description and both textual-reasoning sweeps yield 8, and the two end-to-end sweeps yield 6. Second, CLEVR scene description is scored in this appendix (both heatmap and boxplot) with the strict *perfect property match* criterion, under which a prediction counts as correct only if every property of every ground-truth object is named exactly right. This is a more demanding metric than the lenient mean per-property match used in the main-body violin (Figure 6.2), so absolute accuracies for CLEVR scene description are lower here than in the main body. All other cells — Relations Game scene description, and the CLEVR and Relations Game textual-reasoning and end-to-end cells — use binary per-sample correctness in both the main body and the appendix, and are therefore directly comparable across figures.

D.3.1 Scene Description

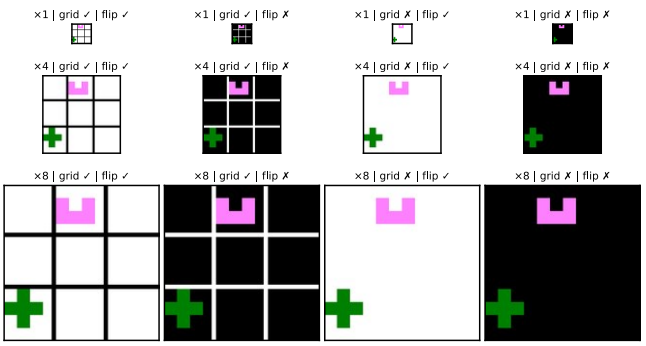


Figure D.1: We vary the image inputted to the models, with the most complete prompt, to find those which models are best able to describe. In particular, we vary the image size (upscale), add or omit a grid, and flip the background colour of the image.

Figure D.5 shows the distribution of per-configuration accuracy across models for the scene description task. Each point represents the accuracy

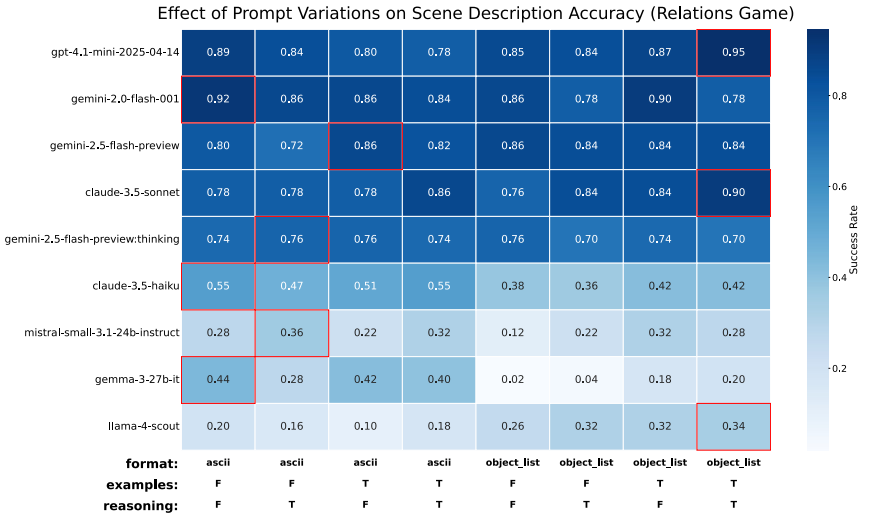


Figure D.2: We vary the contents of the prompt fed to the models, along with relations game images. Here the results are less clear cut, though we first observe that the stronger models tend to perform better on object-list output representations than ASCII representations. As these are also more compatible with the CLEVR dataset, we opt to utilise the object-list format for remaining experiments. Focussing in on the relevant variations to the prompt in the object-list setting, we do not observe a consistent impact from either the addition of in-context examples nor the reasoning, though adding both is sometimes more beneficial for stronger models (which also tend to have larger context windows). As such, our final default prompt uses the object-list format, with in-context examples and reasoning added, but the reader is encouraged to take note of the high variance in performance across the board.

achieved by a model under a particular prompt/image configuration, with the boxplot summarising the spread. This complements the heatmaps above by showing the aggregate sensitivity of each model to prompt and format variations.

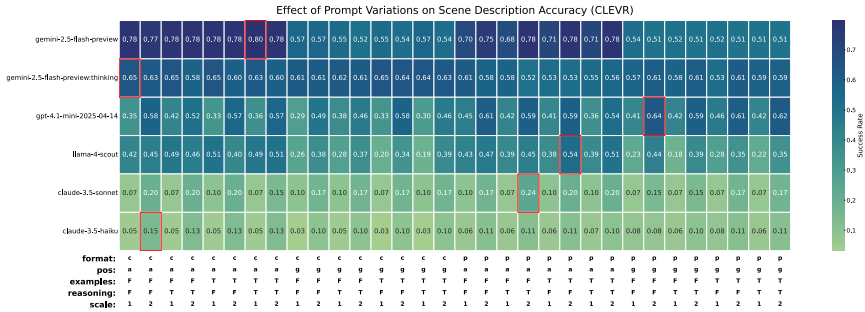


Figure D.3: The percentage of examples (out of ∼60) on which the model correctly describes the properties of all of the objects in the scene. We vary the contents of the prompt fed to the models, as well as attempting to experiment with upscaling the inputted image (by 1× or 2×). For CLEVR we consistently observe that modifying the prompt contents does not considerably impact the performance, but upscaling the image is often beneficial (particularly for weaker models). Additionally, the 'properties' format is beneficial for some, but not all models. Strangely, we see that for Gemini-2.5-flash specifically, requesting the model to provide absolute position coordinates leads to considerably better description of the properties. In Table D.3 we see that the actual positions are far from accurate though, meaning downstream reasoning would be significantly detrimentally affected. As such, we opt to use the grid position format for our primary experiments nonetheless.

Model	Position Format	Perfect Position Match Rate
claude-3.5-haiku	absolute	0.00
claude-3.5-haiku	grid	0.01
claude-3.5-sonnet	absolute	0.00
claude-3.5-sonnet	grid	0.01
gemini-2.5-flash-preview	absolute	0.00
gemini-2.5-flash-preview	grid	0.09
gemini-2.5-flash-preview:thinking	absolute	0.00
gemini-2.5-flash-preview:thinking	grid	0.06
llama-4-scout	absolute	0.00
llama-4-scout	grid	0.00
gpt-4.1-mini-2025-04-14	absolute	0.00
gpt-4.1-mini-2025-04-14	grid	0.03

Table D.3: Perfect position match rates for various models and position formats on the CLEVR dataset. This table shows the proportion of examples where the model correctly identified the exact position of all objects. We specify an acceptable error margin of 1 for z-index (per object), 2 for grid position (manhattan distance), and 0.3 for absolute coordinates (euclidean distance).

Properties Format (with absolute coordinates)

```
Object 1:
- Position: Located at Row 7/20, Column
     9/20, Z-order: 4/5
- Color: Purple
- Shape: Cube
- Size: Small
- Material: Metal
Object 2:
- Position: Located at Row 8/20, Column
     17/20, Z-order: 3/5
- Color: Green
- Shape: Cube
- Size: Large
- Material: Metal
Object 3:
- Position: Located at Row 8/20, Column
     3/20, Z-order: 2/5
- Color: Green
- Shape: Cube
- Size: Large
- Material: Metal
```

Colloquial Format (with grid coordinates)

```
Object 1: A Small Purple Metal Cube.
          Positioned at (x=-1.12, y=0.38,
               z=0.35)
Object 2: A Large Green Metal Cube.
          Positioned at (x=2.49, y=2.92,
               z=0.70)
Object 3: A Large Green Metal Cube.
          Positioned at (x=-1.62,
               y=-2.83, z=0.70)
Object 4: A Small Purple Rubber Cylinder.
          Positioned at (x=1.48, y=-0.29,
               z=0.35)
Object 5: A Large Green Metal Cube.
          Positioned at (x=-0.17, y=2.24,
               z=0.70)
```

Figure D.4: The two scene description formats used in our CLEVR experiments. Left: The structured properties format lists each attribute on a separate line. Right: The colloquial format provides a more natural language description with Cartesian coordinates. In our ablations we experiment with both positional representations for each grid representation. The newlines on the Colloquial format are for readability, and are not included in the prompt.

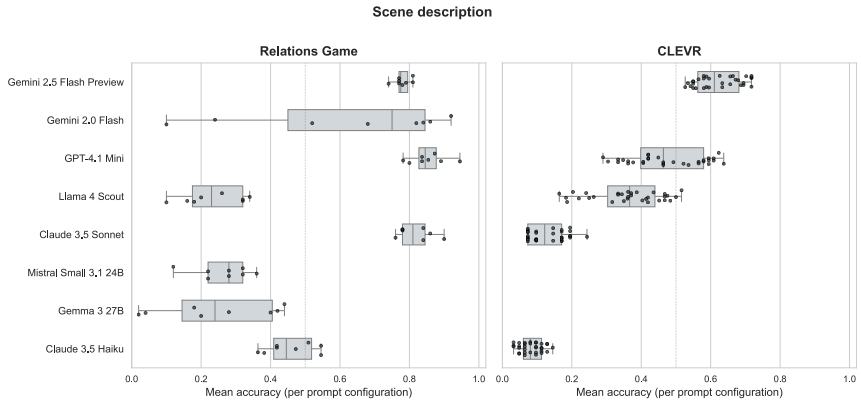


Figure D.5: Distribution of scene description accuracy across prompt and image configurations for each model. Each data point represents the accuracy of a single configuration. The boxplots show the median, interquartile range, and outliers, illustrating how sensitive each model is to prompt and format choices. CLEVR scene description is scored with the strict perfect-property-match metric; Relations Game scene description uses binary per-sample correctness (an object list is correct only if it matches the ground truth exactly).

D.3.2 Textual Reasoning

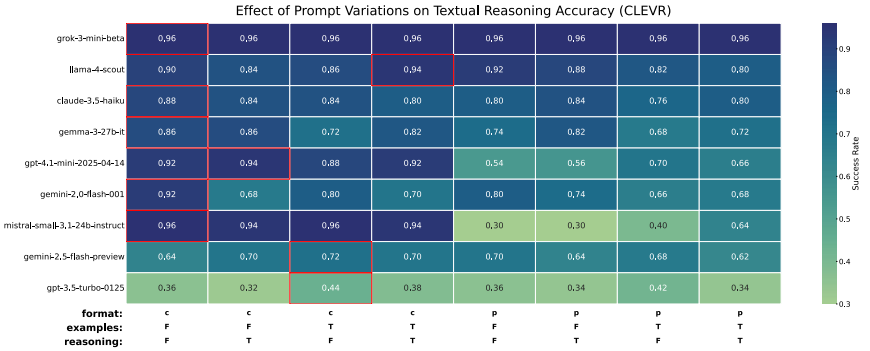


Figure D.6: On the textual reasoning tasks we perform a similar ablation to the prompt formats as for the Relations Game. Here we find that the Colloquial list format considerably outperforms the Properties format across most models, and that both adding examples and prompting for reasoning have little effect.

Figure D.7 shows the distribution of textual reasoning accuracy across prompt configurations for each model. The relatively tight distributions for most models confirm that textual reasoning performance is less sensitive to prompt variations than scene description.

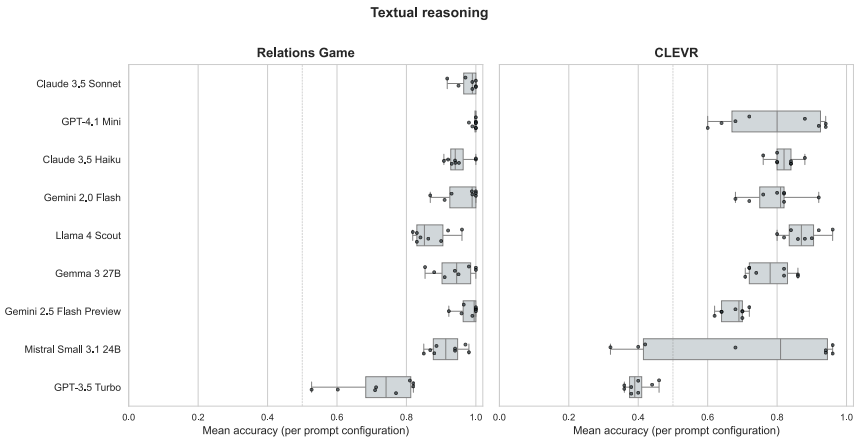


Figure D.7: Distribution of textual reasoning accuracy across prompt configurations for each model. Compared to scene description (Figure D.5), the distributions are generally tighter, indicating lower sensitivity to prompt format choices when models reason from text. Scored with binary per-sample correctness on the final True/False (Relations Game) or short-answer (CLEVR) labels.

D.3.3 End-to-End Reasoning

Figure D.8 shows the results of our end-to-end ablation on the CLEVR dataset, varying in-context example style, reasoning prompts, and image scale.

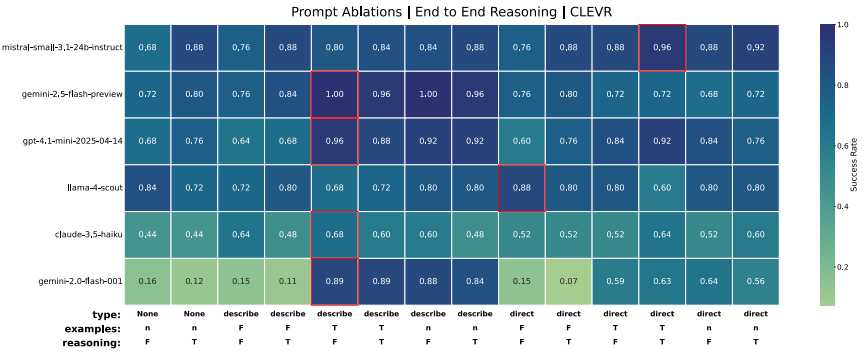


Figure D.8: End-to-end visual reasoning ablation results on CLEVR, varying example style (descriptive vs. direct), reasoning prompts, and image scaling. Descriptive examples and 2× upscaling were found to be optimal.

Figure D.9 shows the distribution of end-to-end reasoning accuracy across configurations. On CLEVR the spread is driven by a handful of weaker models—most notably Gemini 2.0 Flash, whose per-configuration accuracies span 0.15–0.85—while on Relations Game the majority of models cluster tightly in the 0.7–0.9 band. The prompt-sensitivity contrast between concrete and abstract scenes is thus subtler at the end-to-end stage than the scene-description results above might suggest.

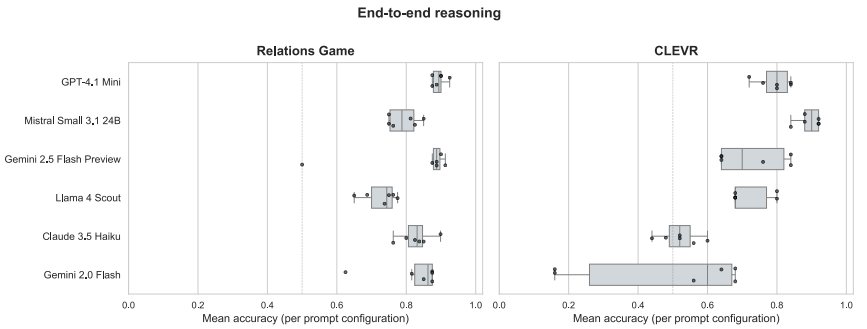


Figure D.9: Distribution of end-to-end reasoning accuracy across prompt and image configurations for each model. On CLEVR the spread is driven by a small number of weaker models, while on Relations Game most models cluster tightly—end-to-end reasoning is less prompt-sensitive here than scene description. Scored with binary per-sample correctness on the final True/False (Relations Game) or short-answer (CLEVR) labels.

D.3.4 Optimal Configuration Summary

Based on our systematic ablations across prompt design and image representation factors, we identify the following optimal configurations for each experimental condition:

We attempt to robustly parse the output formats to account for various formatting choices by the models, below we provide some examples of model outputs which would be correctly classified as correct after parsing[1].

[1] We are more interested in the model's ability to reason about the relations between objects, than we are in prompt adherence.

Task	Dataset	Configuration Type	Optimal Setting
Textual Reasoning	CLEVR	Output Format Examples Reasoning	Colloquial Without Without
Textual Reasoning	Relations Game	Output Format Examples Reasoning	ASCII Grid With Without
Scene Description	CLEVR	Output Format Examples Reasoning Image Scale	Colloquial Without Without ×2
Scene Description	Relations Game	Output Format Examples Reasoning Image Scale Grid Lines	Object List With With ×8 Yes
End-to-End	CLEVR	Example Style Reasoning	Descriptive With
End-to-End	Relations Game	Example Style Reasoning	Direct Without

Table D.4: Summary of optimal prompt and image configurations identified through systematic ablations for each experimental condition. Image-representation settings (image scale, grid lines) were ablated at the scene description stage and then reused unchanged for the end-to-end task within each dataset; only the scene description rows list them. These settings were used for the main results reported in the paper.

Structured Text Format

Target:

```
--- Object [A] in Row [2], Column [1]
--- Object [B] in Row [3], Column [3]
```

Model Answer:

```
1) object [F] (Yellow Tetromino) in Row 2, Column 1
2) object [G] (Pink Tetromino) in Row 3, Column 3
```

ASCII Grid Format

Target:

```
+---+---+---+
| . | A | . |
+---+---+---+
| . | . | . |
+---+---+---+
| B | . | . |
+---+---+---+
```

Model Answer:

```
.  z  .
.  .  .
3  .  .
```

Figure D.10: Whilst we cannot catch all possible output format deviations, our parsing method is able to robustly handle the majority of cases which we observed. Above are two extreme (modified from observed) examples of deviations which our parsing method is able to handle. In these cases, we would correctly classify the model's answers as correct.

D.4 Model Output Examples

To provide concrete illustrations of model behaviour across different tasks and capability levels, we present representative output traces from our experiments. These examples are drawn from the final (optimal configuration) experiment runs. Traces are presented verbatim except where [...] indicates content condensed for space; responses have been lightly reformatted for readability but not otherwise altered.

D.4.1 Textual Reasoning

Figure D.11 shows Claude 3.5 Sonnet performing textual reasoning on the Relations Game "between" task. The model produces a clear chain-of-thought analysis, correctly identifying the spatial relationship between the labelled objects.

Input Grid

```
+---+---+---+
| A | . | . |
+---+---+---+
| B | . | . |
+---+---+---+
| A | . | . |
+---+---+---+
```

Question: Are two identical shapes on either side of a distinct shape?

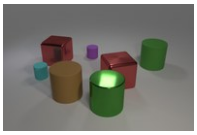
Claude 3.5 Sonnet Response

```
Let me analyse this grid:

1. Looking at the grid, I can see:
   - An 'A' object in the top-left corner
   - A 'B' object in the middle-left position
   - Another 'A' object in the bottom-left corner

2. For the "between" relationship to be true:
   - We need two identical objects (same letter)
   - A different object (different letter) must be
     between them

3. In this case:
   - We have two 'A' objects with a 'B' object
     between them vertically
   - The 'A's are in the top-left and bottom-left
   - The 'B' is in the middle-left, directly
     between the two 'A's

4. This matches the criteria for the "between"
   relationship.

<answer>True</answer>
```

Figure D.11: A representative textual reasoning trace from Claude 3.5 Sonnet on the Relations Game "between" task. The model correctly identifies the A-B-A pattern in the left column and produces a well-structured chain-of-thought analysis. **Answer: True (correct).**

D.4.2 Scene Description

Figure D.12 shows a typical scene description failure from GPT-4o Mini on CLEVR. The model identifies only 6 of the 9 ground-truth objects, hallucinating properties (e.g., "Medium" sizes, which do not exist in CLEVR) and providing incorrect positions (most easily seen by looking at the z coordinates). This example illustrates how descriptions which are mostly accurate (per-property) can still be sufficiently wrong overall to make performing tasks difficult (if using the textual scene descriptions).



GPT-4o Mini (6 objects)

```
<answer>
Obj 1: A Small Cyan Metal
   Cylinder.
  at (x=-1.00, y=-1.50, z=0.25)
Obj 2: A Medium Brown Metal
   Cylinder.
  at (x=0.00, y=0.00, z=0.50)
Obj 3: A Large Green Metal
   Cylinder.
  at (x=1.00, y=0.50, z=0.50)
Obj 4: A Small Purple Metal
   Cylinder.
  at (x=-0.50, y=1.00, z=0.25)
Obj 5: A Large Red Metal Cube.
  at (x=0.50, y=1.50, z=0.75)
Obj 6: A Medium Red Metal Cube.
  at (x=0.50, y=-1.00, z=0.75)
</answer>
```

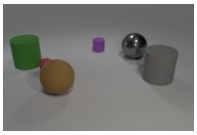
Ground Truth (9 objects)

```
1: Small Green Rubber
   Sphere
2: Small Red Rubber Sphere
3: Small Brown Metal
   Cylinder
4: Large Brown Rubber
   Cylinder
5: Large Gray Metal
   Cylinder
6: Small Purple Rubber
   Cylinder
7: Small Red Metal Sphere
8: Large Green Rubber
   Sphere
9: Large Purple Metal
   Cylinder
```

Figure D.12: A scene description from GPT-4o Mini on CLEVR (scene index 140, shown left). The model identifies only 6 of the 9 ground-truth objects, hallucinating "Medium" sizes (not a valid CLEVR attribute) and misidentifying shapes. This type of partial failure is common among mid-tier models.

D.4.3 End-to-End Visual Reasoning: Successful Traces

Figure D.13 shows Claude Sonnet 4 performing complex compositional reasoning on CLEVR. The model correctly decomposes a multi-clause question into a series of referent-resolution steps and arrives at the correct answer.



GT answer: no

Claude Sonnet 4 Response

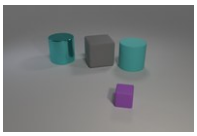
```
Q: Is there a metallic object left of the gray object that is
   behind the large cylinder that is in front of the green
   matte object?

First, I need to identify all the objects in the scene:
1. Large green rubber cylinder (on the left)
2. Small red object (a small sphere, behind/left of the brown
   sphere)
3. Large brown rubber sphere (in the center-left)
4. Small purple rubber cylinder (in the center-back)
5. Large gray metal sphere (on the right side)
6. Large gray rubber cylinder (on the far right)

Now I need to parse the question:
1. "the large cylinder in front of the green matte object" -
   The large gray rubber cylinder on the right, in front of
   the green cylinder.
2. "the gray object behind [it]" - The gray metal sphere.
3. "Is there a metallic object left of [that gray object]" -
   To the left I see: the purple cylinder, the brown sphere,
   the small red object, the green cylinder. The red object
   appears matte (rubber).

<answer>no</answer>
```

Figure D.13: Claude Sonnet 4 performing complex multi-step compositional reasoning on CLEVR (scene index 4). The model correctly decomposes the nested spatial question, resolves each referent, and arrives at the correct answer. Compare with the failure modes shown in Figure D.14 and Figure D.15.

D.4.4 End-to-End Visual Reasoning: Failure Modes

Figure D.14 shows Gemini 2.0 Flash producing an incorrect answer with confused reasoning. The model correctly decomposes part of the question but makes an unjustified assumption when it cannot find an object satisfying all constraints.

Figure D.15 shows a more subtle failure: the model arrives at the correct answer despite self-contradictory reasoning.

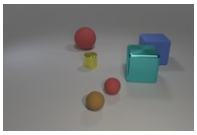


GT answer: cylinder

Gemini 2.0 Flash Response

```
Q: The big rubber object in front of the big matte object left
   of the large cylinder on the right side of the shiny object
   is what shape?

1. Identify the shiny object: the cyan cylinder on the left.
2. The large cylinder on the right side of the shiny object:
   the cyan cylinder on the right.
3. The big matte object left of [it]: the gray cube.
4. The big rubber object in front of the gray cube: not visible.
   However, the small purple cube is in front of the gray cube.
   Since the question asks for a big rubber object, and there
   is no rubber object in the scene, I will assume the question
   is asking about the small purple cube.
5. The shape of the small purple object is a cube.

<answer>cube</answer>
```

Figure D.14: Gemini 2.0 Flash on a CLEVR compositional question (scene index 27). The model answers "cube" (incorrect; ground truth: "cylinder"). In Step 4 it explicitly notes that no rubber object satisfies the constraints, but rather than re-examining earlier steps, it makes an unjustified assumption and defaults to a different object. This failure mode—premature commitment followed by ad-hoc rationalisation—is common across models.



GT answer: large

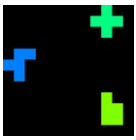
Gemini 2.0 Flash Response

```
Q: What is the size of the rubber object that is behind the
   tiny red rubber ball and on the right side of the tiny
   brown rubber object?

First, I need to identify the tiny red rubber ball and the
tiny brown rubber object.
- There is a small red sphere (assumed to be rubber).
- There is a small brown sphere (assumed to be rubber).
The question asks about the rubber object behind the tiny red
rubber ball and on the right side of the tiny brown rubber
object.
- The blue cube is behind the small red sphere and to the
  right of the small brown sphere.
- The blue cube is not rubber.

There is no rubber object that satisfies the conditions.

I made a mistake in assuming the objects are rubber. Let me
re-analyse.
[...]
- The blue cube is behind the small red sphere and to the
  right of the small brown sphere. The blue cube is large.

<answer>large</answer>
```

Figure D.15: Gemini 2.0 Flash on CLEVR scene index 56. The model answers "large" (correct) but arrives there despite explicitly noting "there is no rubber object that satisfies the conditions," then silently dropping the rubber constraint in its second attempt ([…] indicates condensed content). While the final answer happens to be correct, the reasoning process is unreliable—the model reaches the right conclusion for partially wrong reasons.

D.4.5 Relations Game Visual Reasoning

Figure D.16 shows Gemini 2.0 Flash performing visual reasoning on the Relations Game "between" task. The model systematically checks rows and columns but does not consider diagonal arrangements.



(8× upscaled)

GT answer:

False

Gemini 2.0 Flash Response

```
Q: Are two identical shapes on either side of a distinct shape?

Let's analyse the image.
- The object in row 1, column 3 is a green cross shape, let's call
  it [A].
- The object in row 2, column 1 is a blue L shape, let's call it [B].
- The object in row 3, column 3 is a light green L shape, let's call
  it [C].

Now, let's check for the "BETWEEN" condition:
- Row 1: [Empty] [Empty] [A] - No
- Row 2: [B] [Empty] [Empty] - No
- Row 3: [Empty] [Empty] [C] - No
- Column 1: [Empty] [B] [Empty] - No
- Column 2: [Empty] [Empty] [Empty] - No
- Column 3: [A] [Empty] [C] - No
Therefore, there are no two identical shapes on either side of a
distinct shape.

<answer>False</answer>
```

Figure D.16: Gemini 2.0 Flash performing end-to-end visual reasoning on the Relations Game (BETWEEN task, index 26; the original 36×36 image is shown at 8× upscale as fed to the model). The model answers correctly ("False"), systematically assigning labels and checking rows and columns. However, the model omits checking diagonal arrangements, a potential failure mode.

D.5 Models

Table D.5 lists the eighteen multimodal foundation models evaluated in chapter 6, together with their providers, public release dates, and the announcement posts or technical reports from which we source these dates. Release dates correspond to API availability rather than model announcements where these differ. Models are grouped by provider and ordered chronologically within each group.

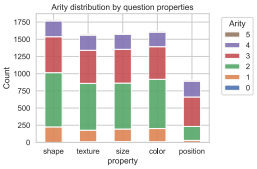
Provider	Model	Release date	Reference
Anthropic	Claude 3 Sonnet	4 Mar 2024	(Anthropic 2024c)
	Claude 3 Haiku	13 Mar 2024	(Anthropic 2024c)
	Claude 3.5 Sonnet `claude-3-5-sonnet-20240620`	20 Jun 2024	(Anthropic 2024a)
	Claude 3.5 Sonnet `claude-3-5-sonnet-20241022`	22 Oct 2024	(Anthropic 2024b)
	Claude 3.5 Haiku	4 Nov 2024	(Anthropic 2024b)
	Claude Sonnet 4	22 May 2025	(Anthropic 2025)
OpenAI	GPT-4o	13 May 2024	(OpenAI 2024b)
	GPT-4o Mini	18 Jul 2024	(OpenAI 2024a)
	GPT-4.1 Mini	14 Apr 2025	(OpenAI 2025)
Google	Gemini 1.5 Pro	8 Feb 2024	(Google DeepMind et al. 2024)
	Gemini 1.5 Flash	14 May 2024	(Google DeepMind 2024)
	Gemini 2.0 Flash	5 Feb 2025	(Google DeepMind 2025a)
	Gemma 3 27B	12 Mar 2025	(Google DeepMind 2025c)
	Gemini 2.5 Flash Preview	20 May 2025	(Google DeepMind 2025b)
Meta	Llama 4 Scout	5 Apr 2025	(Meta AI 2025)
Alibaba (Qwen)	Qwen2.5-VL 7B Instruct	26 Jan 2025[†]	(Qwen Team 2025) (S. Bai et al. 2025)
Amazon	Nova Pro 1.0	5 Dec 2024	(Amazon AGI 2024)
Mistral AI	Mistral Small 3.1 24B	17 Mar 2025	(Mistral AI 2025)

[†]Announcement date; API release date not separately documented.

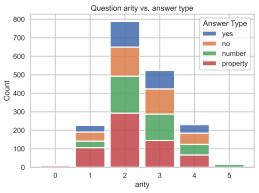
Table D.5: The eighteen multimodal foundation models evaluated in chapter 6, with providers, public release dates, and source references. Models are grouped by provider and ordered chronologically within each group.

D.6 Datasets

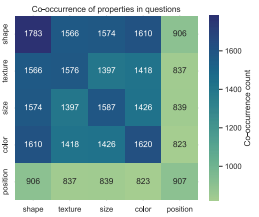
In our ablations we attempt to robustly extract answers, accounting for minor deviations in the model's output format, such as using incorrect delimiters, varying punctuation etc. If we are unable to extract an answer, we mark the example as incorrect.



(a) Histogram of question arities



(b) Distribution of answer types across question arities



(c) Co-occurrence of properties in the contents of a question

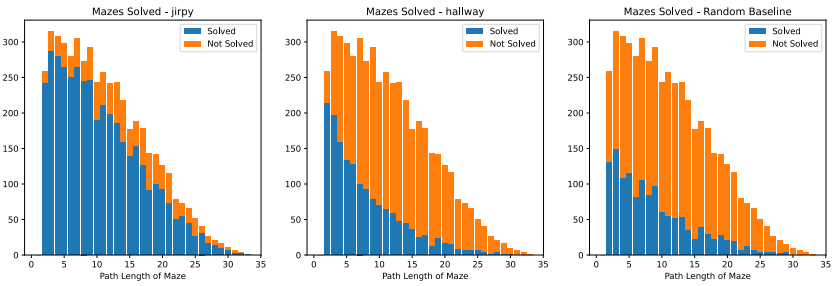
Figure D.17: We perform a crude analysis of the distribution of CLEVR question arities and answer types in our dataset. Arity roughly corresponds to the number of objects in a question.

APPENDIX E

Chapter 8 Appendices

dataset:	forkless		RDFS		pRDFS	
model:	`hallway`	`jirpy`	`hallway`	`jirpy`	`hallway`	`jirpy`
exactly correct rollouts	36.3%	52.7%	36.7%	29.3%	17.6%	21.1%
valid rollouts	50.8%	59.8%	41.4%	29.7%	74.2%	26.6%
rollouts with target reached	87.1%	80.5%	76.6%	98.4%	28.9%	98.4%
`path_start`	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
`origin_after_path_start`	94.5%	95.3%	100.0%	100.0%	97.3%	100.0%
`first_path_choice`	70.3%	70.3%	65.6%	60.2%	66.8%	54.3%
`rand_path_token`	96.9%	93.8%	97.3%	88.7%	83.6%	79.7%
`final_before_path_end`	98.8%	87.5%	94.5%	95.7%	82.8%	90.6%
`path_end`	88.3%	81.6%	100.0%	100.0%	99.6%	100.0%

Table E.1: Results on both models on 7×7 forkless, pRDFS, and RDFS mazes (See subsection 7.3.1), also with $n = 256$ samples. Note that the `jirpy` model was trained only mazes that were only dense in 6×6 subgrids, and generalises relatively poorly to these larger mazes. Conversely, the `hallway` model was trained on only sparse mazes to start with, and performs similarly.



(a) `jirpy` **(b)** `hallway` **(c)** Random Baseline

Figure E.1: Performance of our models on a held-out test set of RDFS mazes. Noticeably, the performance appears roughly consistent over path length for `jirpy` which was trained to solve mazes, and similarly for the random baseline (which follows corridors and chooses a random continuation of its path when reaching a fork). The hallway model, on the other hand, is able to "solve" some of the very short mazes but struggles with longer mazes (where it is likely to encounter forking points).

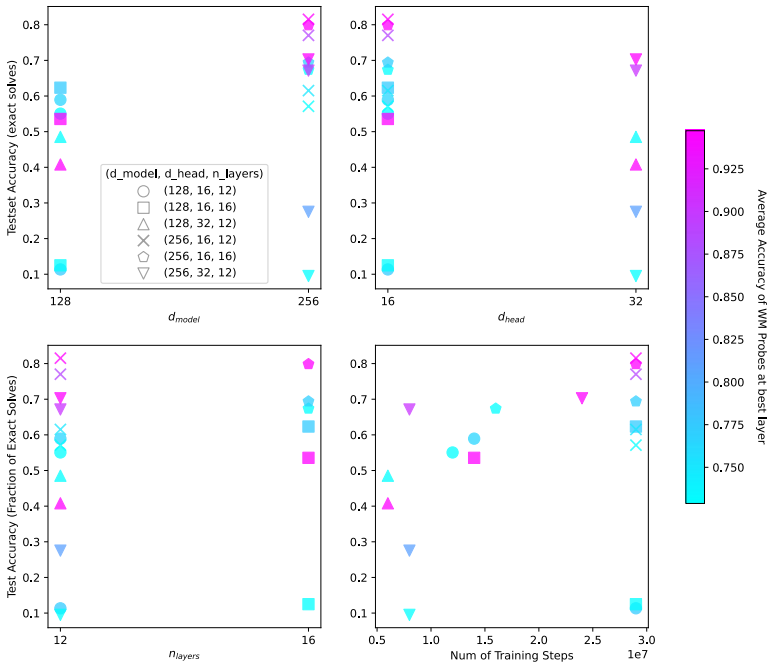


Figure E.2: Results of sweeps carried out across a variety of models which were trained for varying lengths. We find that 1) Models can do well even if they don't have a linear maze representation, which we can decode, but the very best models also have a linear maze representation. 2) d_{model} and training time are the only hyperparameters that seemed to correlate with performance in the regimes we considered, but any such correlations are too weak to draw strong conclusions. The model with the highest test accuracy and probe accuracy is `jirpy`.

E.1 Embedding Structure

The randomly initialised token-embedding layer acquires structure such that Manhattan distances between coordinate-token embeddings correlate with spatial distances between the corresponding cells in the maze grid. This parallels the main-chapter observation that both models learn embeddings whose geometry reflects the underlying coordinate system, providing a substrate on which down-stream computations can factor in coordinate proximity when predicting path tokens. The grids in Figure E.3 visualise this correspondence for a single reference coordinate in each model.

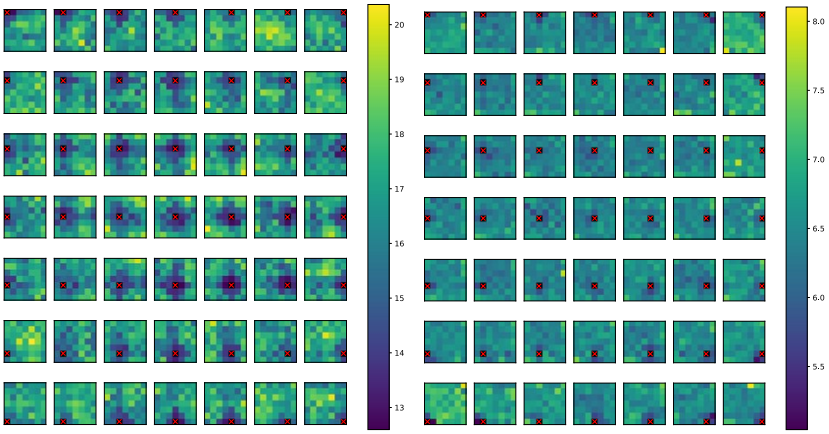


Figure E.3: Manhattan distance visualisation for coordinate token embeddings. Given the coordinate at the x, we view the Manhattan distance to the other tokens on the grid. **Left:** `hallway` model. **Right:** `jirpy` model. Both models show similar spatial structure in their embedding spaces, with nearby coordinates having smaller embedding distances. Reproduced from Ivanitskiy, Spies, et al. 2024 with permission from M. Ivanitskiy.

E.2 Additional Probing Results

Across the training sweep, linearly decodable maze representations emerge most clearly at layer 2 of the `jirpy` model, where per-position probe accuracies peak before declining at deeper layers (Figure E.4). Table E.2 ranks checkpoints by maze-solving accuracy alongside the best-probe layer and its accuracy, making it clear that the best-performing models also admit the most linearly decodable world models. The apparent saturation of edge-cell

accuracies reflects an artefact of outer-wall priors—roughly 1/4 of probes on edge cells and 2/4 on corners predict a wall correctly by default—rather than additional spatial information being present at those positions.

Maze-solving Acc (%)	Model	Best-probe Layer	Training Steps (M)	Average Probe Acc (%)
29	bcgldy45	4	29	76
31	gvz6wk68	1	29	77
64	2fl2nstp	8	14	76
67	pepyeckt	12	14	77
68	1pfu6p0k	1	29	80
70	9hz4sg8x	10	12	73
73	tjqgo9js	13	29	77
76	badteylo	9	29	82
78	fbjfutxq	2	8	92
80	jwzm08kc	3	24	95
81	b8l1p5yz	13	29	80
84	7j5u30hx	12	16	77
86	sa973hyn	2	29	91
86	spb2yilg	2	29	90
90	`jirpy`	2	29	95

Table E.2: Best probe accuracies (across layers) for best performing model checkpoints (on the test-set). The `jirpy` model was the most performant model, both in terms of solving mazes and yielding the best set of probes (see the sweep results in Figure E.2).

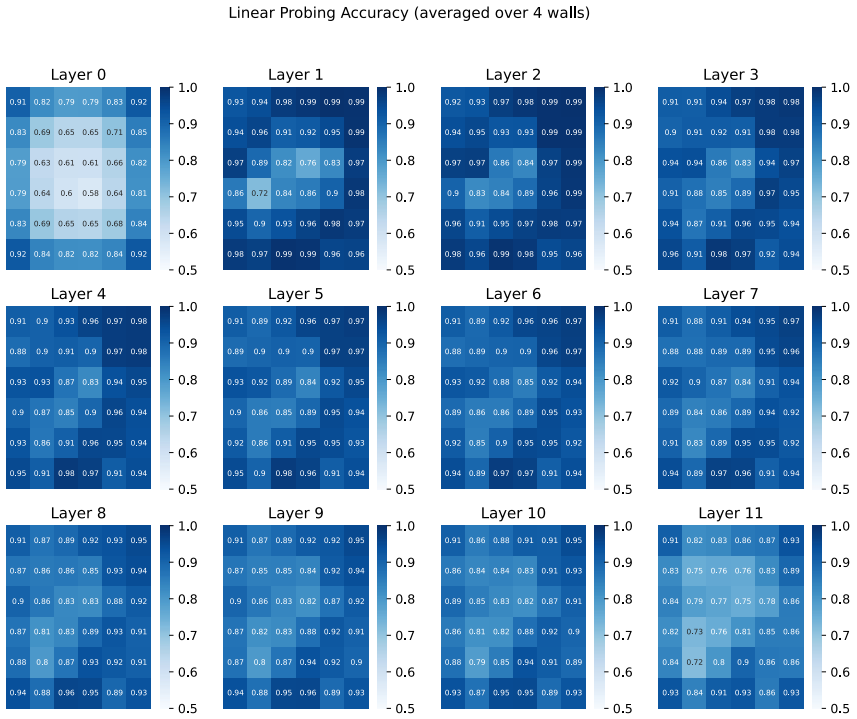


Figure E.4: Accuracy of linear probes trained on the residual stream at all layers of the `jirpy` model. We see that on Layer 2 the highest average probing accuracy is achieved, and the accuracy decreases with later layers. Note that the edges have very high accuracy as outer walls are always present, meaning that 1/4 of the probes on edges (and 2/4 on corners) will achieve 100% accuracy by always predicting a wall.

E.3 Direct Logit Attribution

Direct Logit Attribution (DLA) quantifies the contribution of each attention head's output to the correct-token direction in the residual stream, indicating which heads actively participate in producing the next-token prediction. Figure E.5 reports DLA for the `hallway` model across a subset of tasks taken from held-out samples of its training distribution. The per-task contrast is informative: only `first_path_choice` requires computation beyond simple path-following, and the corresponding shift in per-head contributions aligns with the failure modes visible in Table E.1, where `hallway` performs well on corridor-like mazes but poorly when forking choices must be resolved.

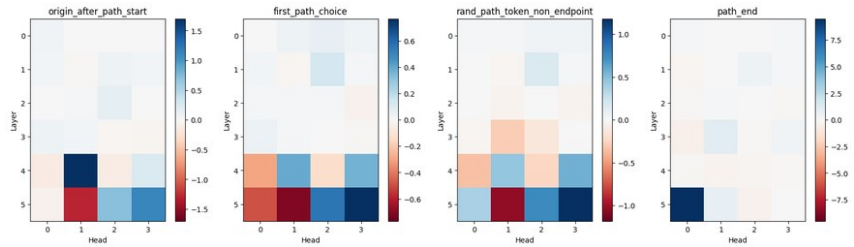


Figure E.5: DLA of the `hallway` model across a subset of tasks, on held-out samples from the training distribution. The numerical value is the contribution of a given attention head to the "correct" direction in the residual stream. Note that only for `first_path_choice` must the model do anything besides path-following, as reflected in e.g. Table E.1. Reproduced from Ivanitskiy, Spies, et al. 2024 with permission from M. Ivanitskiy.

APPENDIX F

Chapter 9 Appendices

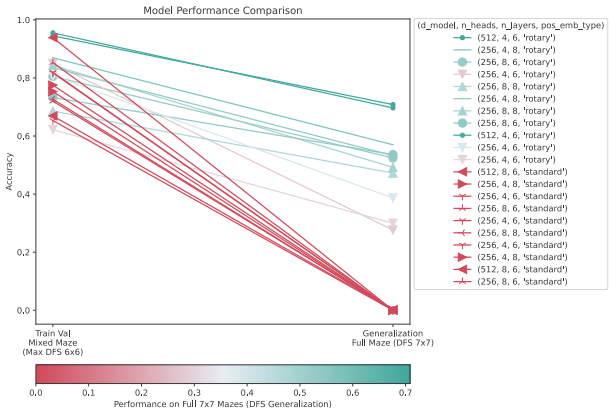


Figure F.1: Generalisation performance across different hyperparameters. "Train Val" shows the accuracy on held-out in-length-distribution mazes from training time, while "Full Maze" features mazes with more connections (longer input sequences) than those seen at training time. The results demonstrate that only models with rotary positional embeddings are able to generalise effectively to longer sequences, highlighting the importance of appropriate positional encoding schemes for length generalisation.

F.1 SAE Training Details

To choose optimal hyperparameters for our SAEs we ran a sweep over SAEs at layers 2 to 4 on Terry, finding consistent trends across layers. The results of this sweep are shown in Figure F.2, and the final details of the SAEs analysed in the main paper are given in Table F.1. We also provide feature density histograms for the SAEs analysed in the main paper in Figure F.3 noting that these look good, in that many features are sparse, but also rather distinct from is typically observed in LLMs. This is not surprising, as our token and features distributions will be very distinct from those of natural language, as most mazes have many active connections, and connections are similarly likely to be present in any given maze.

To prevent "neuron death" in the SAE latent space, resulting from high sparsity penalties, we apply the method of "Ghost Gradients" proposed by Jermyn et al. 2024—these provide explicit gradients to features that have been inactive for a long time in the direction of the (so-far) un-captured parts of the SAE's reconstruction of the residual stream.

Sparsity			**Dataset**		**Optimizer**	
Expansion Factor	Ghost Threshold	(L0) Sparsity Weight	Batch Size	Training Steps	Learning Rate	Linear Warm Up Steps
4	100	0.01	1024	$\sim 10^6$	10^{-4}	1000

Table F.1: Hyperparameter values for the final SAEs analysed in the main paper.

		SAE Feature Metrics		**Average Token Reconstruction Errors**		
	Residual Reconstruction Error (L2)	Sparsity (L1)	L0	Unperturbed	Zero Patched	SAE Patched
Terry	2.87×10^{-4}	10.7	20.9	8.18	8.52	8.18
Stan	6.35×10^{-4}	9.53	28.4	6.35	8.61	6.35

Table F.2: SAE Metrics for the final SAEs trained on Stan and Terry. We see that replacing the residual stream with the SAE reconstructions has very little impact on the sequence produced by the model, providing confidence that the SAEs are encoding all the relevant information in the model's residual stream.

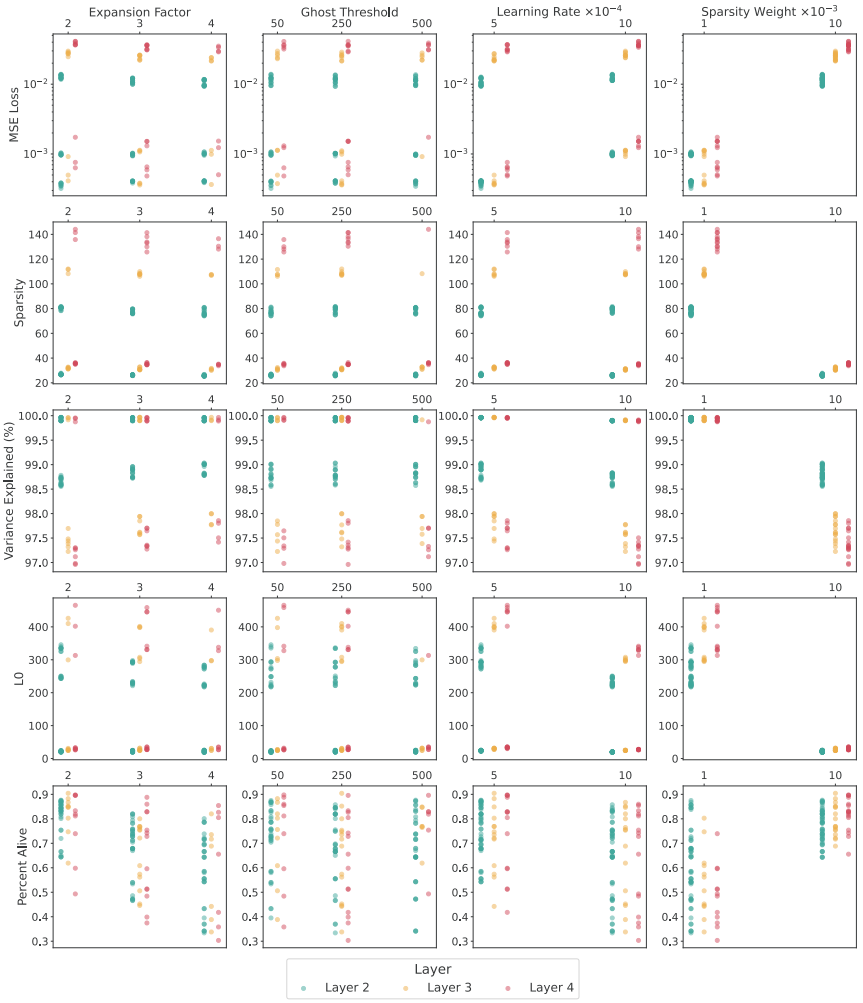
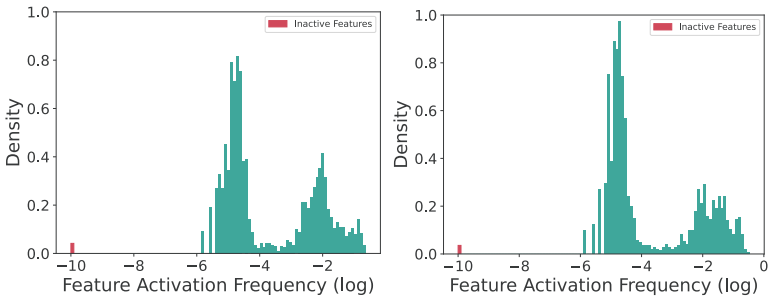


Figure F.2: Results of an SAE sweep carried out on Terry.



(a) Terry SAE density **(b)** Stan SAE density.

Figure F.3: Feature density histograms for the SAEs analysed in the main paper.

F.2 Further Attention Visualisations

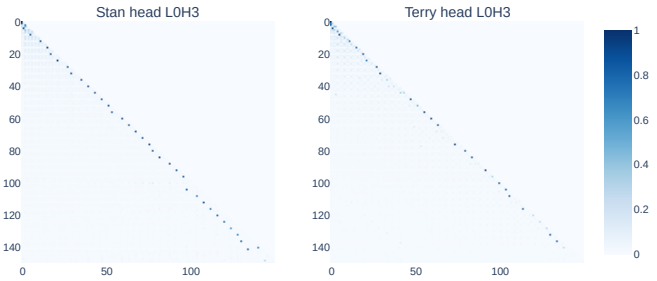


Figure F.4: Attention patterns for head L0H3 in Stan and Terry, for a specific example maze. At every fourth context position from 4 through to 140 (the `;` positions in the adjacency-list) attention is directed very strongly back to one or two positions, typically 1 or 3 positions earlier in the context (though for Stan, after context position 100, this shifts to 5 or 7 positions earlier in the context). This pattern is qualitatively repeated across all examples examined, for heads L0H3, L0H5 and L0H7 in Stan, and for all four L0 heads in Terry. Reproduced with permission from W. Edwards.

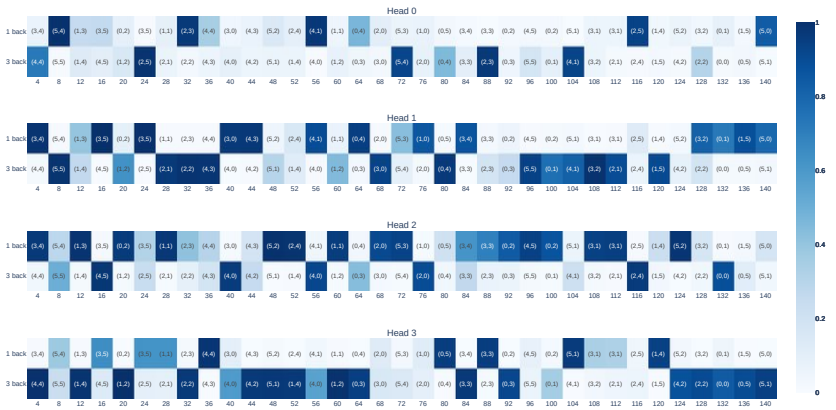
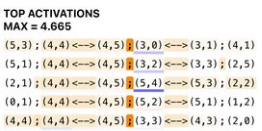


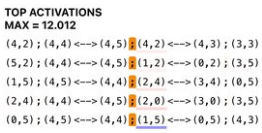
Figure F.5: Attention values for layer 0 heads in Terry, from positions holding the ; token (shown along the x-axis) to positions 1 and 3 earlier in the context (shown along the y-axis), for an example maze input. This pattern is typical across all inputs examined. The pattern is less clear-cut than for Stan (Figure 9.2), but note that at every fourth context position, there is at least one head attending strongly to positions 1 and 3 earlier in the context. Reproduced with permission from W. Edwards.

F.3 How SAE Representations Differ

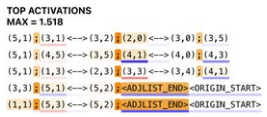
The three panels of Figure F.6 illustrate the encoding-strategy contrast developed in the main chapter: Terry encodes each connection cleanly in a single dedicated feature, whereas Stan employs a compositional code in which each connection is represented jointly by a generic "semicolon" feature and a connection-specific feature. We hypothesise that Stan's compositional code arises from an imperfect separation of learned positional and semantic information, while Terry's cleaner single-feature encoding benefits from RoPE's explicit separation of positional information from token content.



(a) Terry model: A single SAE feature directly encodes a specific maze connection, demonstrating Terry's straightforward representation of maze connectivity.

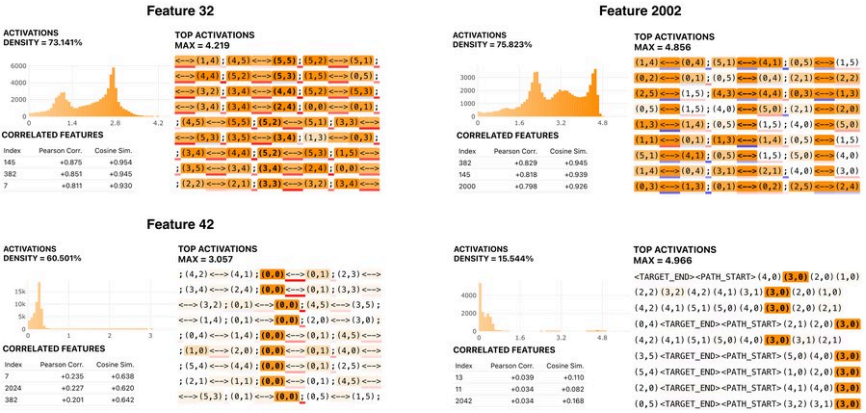


(b) Stan model: The connection-specific feature activates at the semicolon corresponding to the encoded connection, similar to Terry's encoding strategy (see Figure F.6a).



(c) Stan model: Feature 1422, in conjunction with another feature, encodes maze connectivity. This feature appears consistently across all connections, corroborating the decision tree decoding results in Figure 9.5.

Figure F.6: Maximally activating examples, displayed using a modified version of McDougall 2024 for SAE features encoding the connection `(4,4)` `<->` `(4,5)`, as identified by decision tree decoding. Underlines correspond to loss contribution (blue for positive, red for negative) and highlighting indicates feature activation at a given token position. Connection-specific features in both models (Figure F.6a and Figure F.6b) show clear activation patterns, while Stan's generic semicolon feature (Figure F.6c) exhibits a less obvious trend. Produced using a modified version of McDougall 2024



(a) Representative SAE features for Terry.

(b) Representative SAE features for Stan.

Figure F.7: We provide examples for the types of features observed in Stan and Terry, beyond the connection features which form the primary focus of the main paper. We observe the same kinds of features between both transformers, and in both cases the predominant features are of the form observed in the top-left (Feature 32 in Terry and 2 in Stan) — These features are more distributed and harder to interpret than the others, and may be suppressed by higher sparsity penalties.

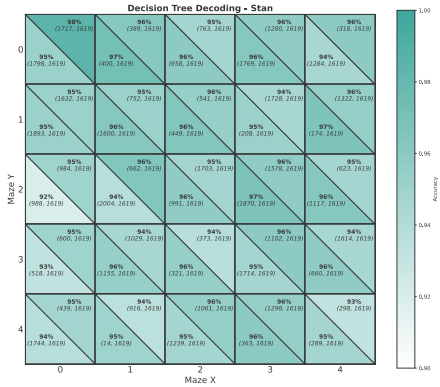
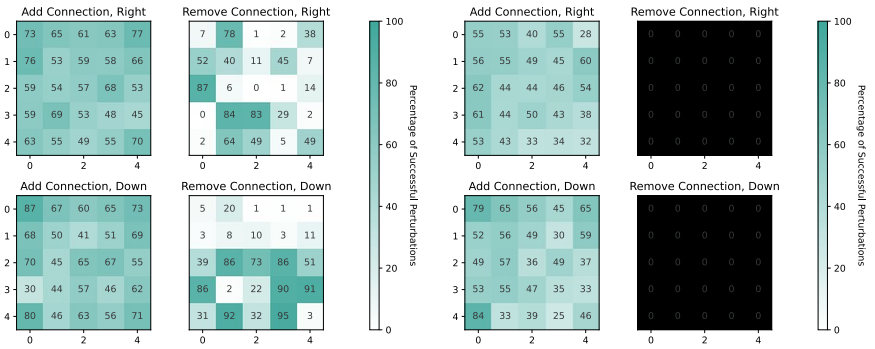


Figure F.8: Another SAE trained on Stan gives rise to the same compositional code.

F.3.1 Magnitude of interventions

To complement the intervention results presented in the main text, we also conducted fixed-value interventions on both the Stan and Terry models. In these interventions, instead of calculating new activations based on the modified input, we directly set the activations of the targeted features to fixed values. This approach allows us to examine how the models respond to more controlled manipulations of their internal representations.



(a) Terry model **(b)** Stan model

Figure F.9: Aggregated accuracy of fixed-value interventions for examples on which the original prediction was correct. As opposed to Figure 9.9, the addition interventions were performed with a fixed value of 10 (removal interventions were the same, with a fixed value of 0). Here we see that the fixed-value interventions are mostly less effective than the calculated interventions, suggesting magnitude sensitivity for feature magnitudes in the transformer's use of the World Model.

The fixed-value intervention results shown in Figure F.9 reveal interesting patterns that both complement and contrast with the calculated intervention results presented in the main text. Consistent with the main-chapter observation that fixed-value toggles were generally less effective than calculated interventions, the reduced accuracy here indicates magnitude sensitivity in how features participate in the world model: downstream circuits do not simply read feature presence but are tuned to the distribution of activation strengths seen during training. The same asymmetry reported in the main text—activating features alters behaviour more reliably than suppressing them—is visible here as well, suggesting that the transformers' world models are more robust to feature suppression than to feature enhancement.

F.4 Investigation of QK-circuit in Stan model

This section complements the OV analysis of the main chapter by inspecting the query–key side of the same three heads. The main chapter reports that L0H3, L0H5 and L0H7 attend, at every fourth context position (the `;` separator tokens), either 1 or 3 tokens back—each time picking up one of the two coordinates that define a maze connection—and that the heads specialise complementarily by maze-cell parity (L0H3 on even-parity cells; L0H5 and L0H7 on odd-parity cells). The figures that follow show that these patterns are visible not only in the OV circuit but already in the QK overlaps for the token embeddings, suggesting that the attention-pattern selection itself is implemented through structured query/key geometry.

In an effort to better understand the notable "1- and 3-back" attention patterns appearing in heads L0H3, L0H5 and L0H7 of Stan, described in subsection 9.2.1, we investigated the query and key vectors for token and positional embeddings, and their overlaps. The scalar products between queries and keys of token embeddings for L0H3 are shown in figure F.10. The most striking feature of this plot is the row corresponding to the query vector of the `;` token, and in particular its overlap with the maze cell tokens. Plotting these scalar products on the maze cell grid (figure F.11) a clear pattern emerges, analogous to that shown in figure 9.3, accounting for LH03's tendency to attend to even-parity cells, and LH05's and LH07's tendencies to attend to odd-parity cells.

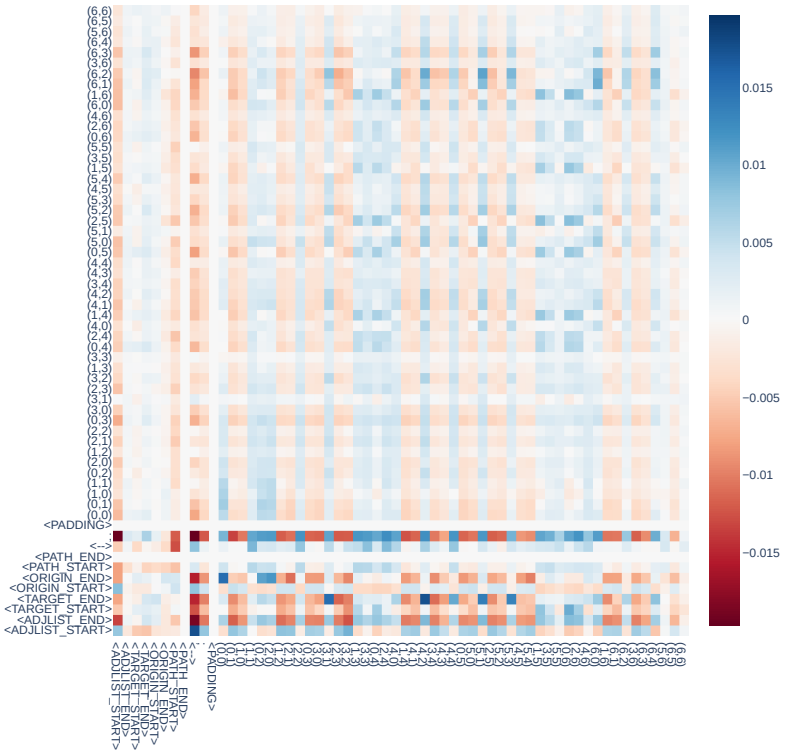


Figure F.10: Scalar products of Stan LH03 of query (rows) and key (columns) vectors for token embeddings. Note that the most pronounced pattern is found on the row corresponding to the query vector of the `;` token, reflecting the importance of this head in establishing the attention pattern from context positions containing the `;` token. This figure was produced in collaboration with W. Edwards.

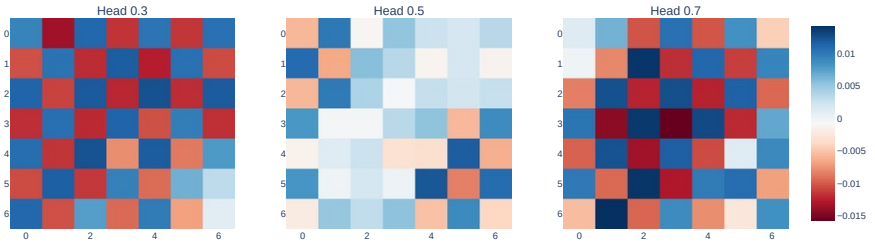


Figure F.11: Stan scalar products of query vector for `;` token and key vectors for maze-cell tokens, arranged on maze grid. Note the clear correspondence with Figure 9.3. These patterns account for why LH03 directs its attention to even-parity cells, while odd-parity cells are attended to by LH07 or LH05.

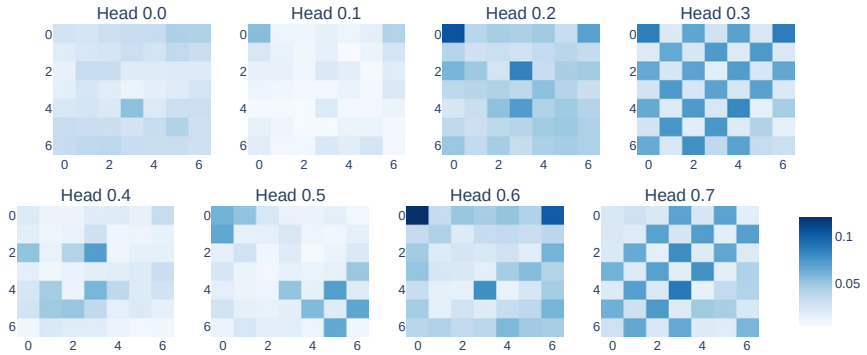


Figure F.12: Stan OV projections across position embeddings for all heads.

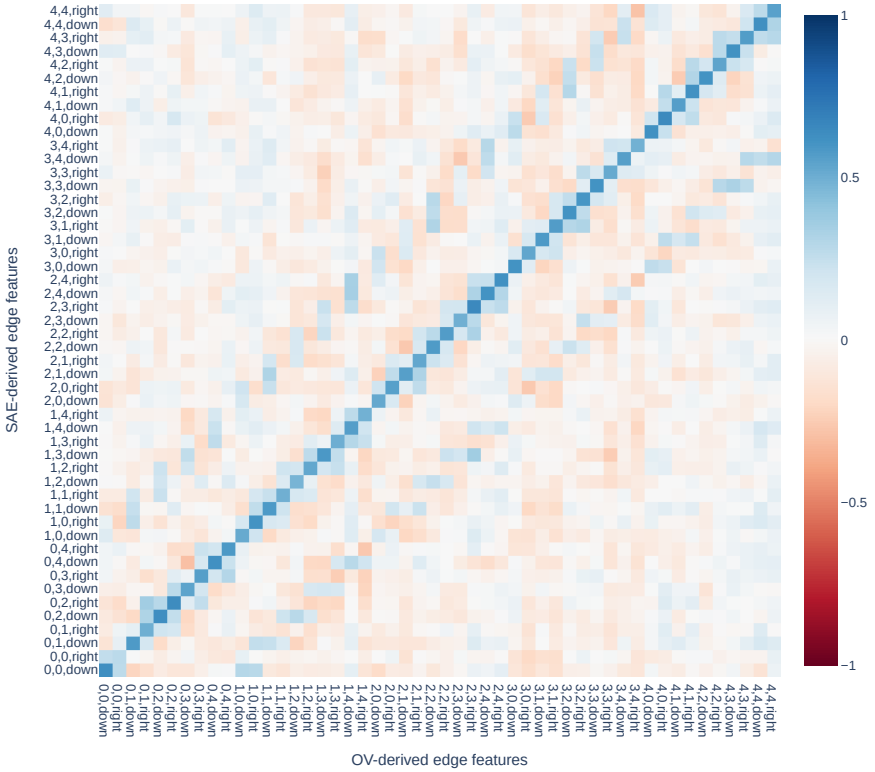


Figure F.13: Cosine similarities between edge features derived from the SAE and edge features derived by direct application of the W_{OV} matrices of the heads discussed in subsection 9.2.1 to token embeddings for Stan, showing both right- and down-directed edge features. This extends Figure 9.7a, which shows only right-directed features. This figure was produced in collaboration with W. Edwards.

F.5 Comparing SAE features and Connectivity Attention Heads

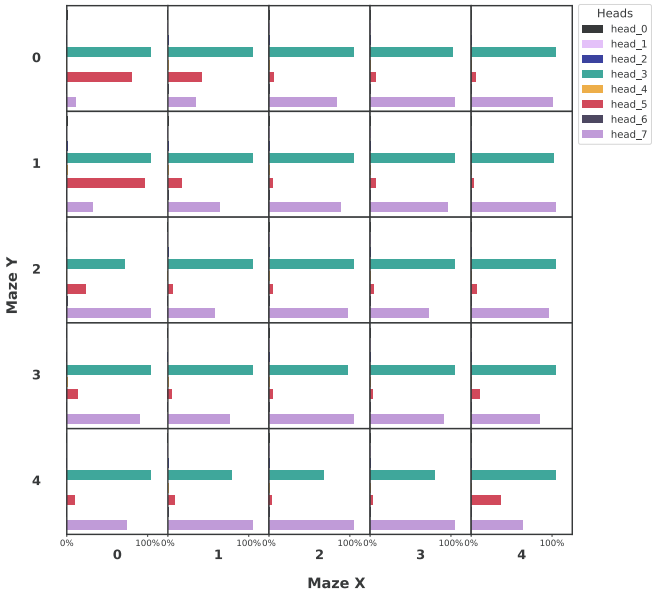
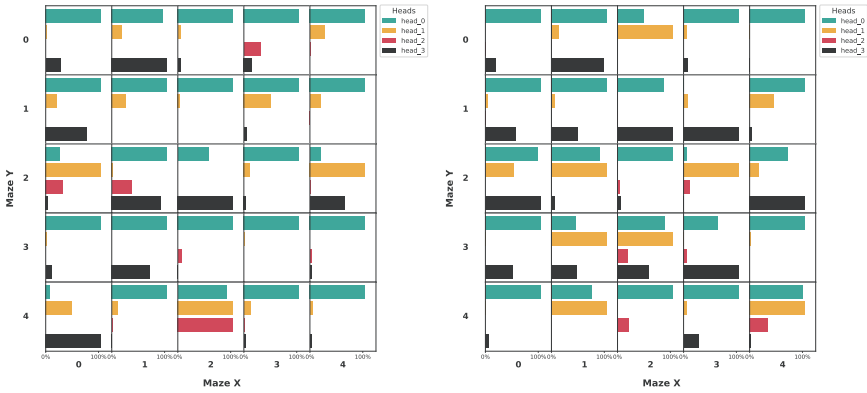


Figure F.14: Effect of patching attention heads on SAE features for each down-connection Stan. These again provide agreement with the OV analyses performed in the main text.



(a) Effect of attention patching on right-connection features.

(b) Effect of attention patching on down-connection features

Figure F.15: Effect of patching attention heads on SAE features for Terry. Whilst we observe notable effects, it is difficult to see a clear pattern — as revealed by the attention analyses, the role of each head in constructing a single connection feature in Terry is harder to understand.

F.6 Computing SAE and OV edge feature similarity

In Figure 9.7a we compute the cosine similarity between SAE edge features and OV circuit edge features.

SAE edge features are formed from a linear combination of the specific edge feature and a "generic edge" feature, with the generic feature coefficient of -0.6 being chosen to maximise cosine similarity.

OV edge features are formed from a weighted sum:

$$\sum_{h,c} a_c^h W_{OV}^h t_c$$

Here, h indexes heads L0H3, L0H5 and L0H7, with $W_{OV}^{\texttt{L0H3}}$, for example, giving the OV matrix of L0H3. c indexes the two cells present in the edge of interest, and t_c is the token embedding of a cell c. The coefficients a_c^h are given by the attention directed by head h to cell c from the `;` context position following the specification of the edge of interest. Data was averaged over 100 examples (see Figure 9.2 for one such example).