Formal Languages for Mechanistic Interpretability

by Abhinav S Menon 2020114001

Formal Languages for Mechanistic Interpretability

Thesis submitted in partial fulfillment
of the requirements for the degree of

*Master of Science in **Computational Linguistics** by Research*

by

Abhinav S Menon
2020114001
abhinav.m@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA
May 2025

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Formal Languages for Mechanistic Interpretability"** by Abhinav Menon, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ Date

_____ Adviser: Prof. Manish Shrivastava

To Amma and Dada.

Acknowledgements

I'm grateful to my advisor, Dr. Manish Shrivastava, whose insights and discussions, both as a research guide and as a professor, have been instrumental in sharpening my intuitions towards deep learning and interpretability.

I also have the deepest gratitude to Dr. Tobias Grosser, Dr. Andrés Goens, Siddharth Bhat, and Arjun Pitchanathan. My collaboration with them, in addition to being one of the most memorable times of my college life, taught me much about how to conduct a research project and work in a team.

I'm also indebted to Dr. Ekdeep Singh Lubana and Dr. David Krueger, with whom I co-authored the work this thesis is based on, and whose guidance and support was crucial in understanding how to do good science and present it effectively.

Throughout my degree, I've been helped out by several seniors from their experience, connections, and guidance, whose efforts I hope to pay forward – Saujas Vaduguru, Kunwar Grover, Mukund Choudhary, Atreya Ghoshal, Shashwat Goel, and Ameya Prabhu.

Finally, on a personal note, I would not be where I am today without the unwavering support of my parents and brother. I'm also grateful to my school friends – Samay, Abhiram, Vidushi, Rohith, Arjun, Dhruv, Advaith, Ananya, Pranav, Romir and Akshath – for sticking together, keeping the memories alive, and always making new ones.

The last five years have been unforgettable, and my college friends – Shashwat, Pratyaksh, Lakshmanan, Nanda, Aneesh, Pranjal, Manav, Raghav, and Priyanshul – are the reason I've reached this point while still taking away memories of trips, hangouts, and movies, rather than deadlines and workloads.

A big thanks also to the people who made my life abroad a time to remember: Anton, Luisa, and Dalia.

All these people, and several more, have in innumerable ways contributed to my growth, well-being and contentedness. I'm eternally grateful to them for their support and sacrifice, and for everything I've learnt from them.

Abstract

Neural models have seen exponential changes, both in terms of scale and deployment, in the years since transformers and large language models were developed. The scale of these models and of their training data have enabled them to reach near- (and in some cases super-) human performances in several tasks. However, this raises concerns of value misalignment and potential misbehaviour of these models in high-stakes situations. This creates the need for a more fine-grained, general, and mathematical understanding of the functioning of these models, with the objective of being able to reliably and generally predict and control their behaviour. This is the central effort of interpretability, a field of study aiming to reduce the heavily overparameterized functions implemented by neural nets to simple, sparse, and abstract causal models.

However, the relative immaturity of the discipline has meant that the rigour of paradigms, techniques, and experiments has not seen consensus. In this thesis, we present a proof of concept that analogy with the natural sciences can form a valuable foundation for achieving the long-term aims of interpretability; in particular, we leverage the reductionist approach to understanding complex systems, and apply it to the study of deep models.

We restrict our scope to models that operate on natural language – or, more generally, text – rather than other modalities like images, audio, or time series. We take inspiration, therefore, from computational linguistics, which in its incipient phases relied on a remarkably expressive reduction of natural language – formal grammars. We exploit this concept to idealize the conditions under which we examine neural language models, and present a study that operationalizes this intuition.

Concretely, we examine the recently popular sparse autoencoder (SAE) method for interpretability. This method centres on using two-layer MLPs with a sparse, overcomplete hidden representation, trained to encode a latent space of a large model, in the hopes that meaningful semantic decompositions of this space arises. We use language models trained on formal grammars, attempt to uncover relevant features using this approach, and try to find properties of the approach that are significant to its usability. Our findings align for the most part with existing conclusions on the properties of SAEs (although these were based mostly on experiments in the image domain) such as their sensitivity to inductive biases and lack of robustness. Most significantly, we note that the features identified by SAEs are rarely causally relevant – ablating them fails to produce the expected effects most of the time. As causality has emerged as a widely agreed upon *sine qua non* among interpretability researchers, this is a major deficiency of the

method. We propose, accordingly, a modification of the pipeline that aims to incentivize the causality of identified features, and demonstrate its efficacy in the same setting of formal grammars.

Overall, we believe that our results demonstrate the potential of importing scientific *modi operandi* into interpretability, and more specifically, the capacity of reductionism to provide useful insights into the functioning of deep models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Neural networks have seen tremendous application as a solution to outstanding problems in several disciplines, including (but not limited to) the natural sciences, medicine, social sciences. One of the fields, however, which has been all but revolutionized by the application of deep learning is natural language processing, in which developments have been so rapid that benchmarks struggle to keep up with technologies, rather than the other way around. The endeavour to create a mathematical model of language has spurred the development of a number of machine learning techniques, including RNNs and their variants, attention, and transformers, and has culminated in the burgeoning effort to build large language models (LLMs) – deep learning systems which display language usage capabilities equal to or surpassing humans' in many respects.

There has, therefore, been an exponential rise in the language abilities of artificial systems, which have progressed from models with 2-3 layers and about 10^8 params [53] to those with 126 layers and 10^{11} params [71]. This rise has enabled a corresponding increase in the utilization of these systems in economically productive activities, like software development, content generation, and translation [24]. However, this phenomenon has led to widespread concerns for multiple reasons. Firstly, the improved capabilities of LLMs enables their expanding involvement in high-stakes environments, like hiring decisions and medical diagnoses. This is symptomatic of a larger tendency to use deep learning systems in critical settings, like self-driving cars and facial recognition systems. Combined with the observation that these models frequently reflect biases in their training data [16, 10] , and consequently in the real world, this implies that such biases can be reinforced by their widespread operation.

Secondly, the scale of these models necessitates training data in quantities that can only be obtained by investing huge amounts of resources into scraping the internet. This has the effect of including private (or otherwise sensitive) data into the training data, and thus of models leaking this information at inference time after deployment. Such breaches may occur inadvertently (in a day-to-day situation under an ordinary prompt) or they may be engineered via jailbreaking or other adversarial methods. This has the potential to create unprecedented security concerns.

Thirdly, in a strict capabilities sense, the failure modes of deep learning systems are poorly understood. Much of their functioning in economically valuable activities relies on empirically observed,

but theoretically unexplained, facts of their operation. For example, the capacity of these models to generalize out-of-distribution (OOD) has not been accounted for by any scientific standards; yet a large part of their functioning in the real world depends on this ability.

Therefore, the need of the hour is to understand, explain and abstract the operation of deep learning models. The above concerns make it clear that a general and rigorous understanding of these systems is the only way to mitigate the risks currently inherent in their deployment. To this end, many distinct strands of research have arisen, each aiming to deal with some subset of the issues, and each employing a distinct set of techniques and methods.

In this thesis, we restrict ourselves to mechanistic interpretability, or the effort to develop theories to describe models' functioning in an algorithmic sense on the basis of their internal representations. We first outline the aims of and progress in the field of interpretability, and then describe the perspective we take towards these aims in terms of scientific approaches.

1.1 Interpretability

Interpretability as a field has been growing since the development of what are now considered 'primitive' deep learning techniques and models; many of these studies and insights, however, can be classified under interpretability only retroactively, as the field has begun to crystallize only in the last few years. We restrict ourselves in this section to interpretability for natural language processing, or more generally, text processing models.

The development of global word representations [45, 52] was the earliest major step towards neural methods for natural language processing. These vector representations proved useful for tasks like sentiment classification; the way information was encoded in these vectors, consequently, became an object of study. One of the classic insights that arose out of this effort was the observation of "linearity" in semantics – the representations of "king" and "queen" had almost the same difference as those of "man" and "woman," Figure 1.1 indicating a simple additive transformation between "male" and "female" semantics. Similar properties were identified for other semantic distinctions.

While this particular property had little practical significance, it had the impact of creating an intuition of the *geometry* of representation spaces – the properties that corresponded to semantic distinctions and manipulations, operations, compositionality, extractability, complexity, and so on. This framework of thinking about representations has persisted and continues to inform much of the work in interpretability and related fields. Many of the concepts dictating results and experiments, in particular, are based on this foundation. It has, consequently, informed many studies in contextual representations, i.e., layer-wise representations of deep transformer-based language models. All the concepts and efforts described below take such contextual representations as their scope.

An important concept in the study of representations is a *feature*, which is generally understood to mean 'a semantically significant direction in the representation space.' A feature, therefore, usually refers to some linear combination of *neurons*, or individual elements of the vector representations. There

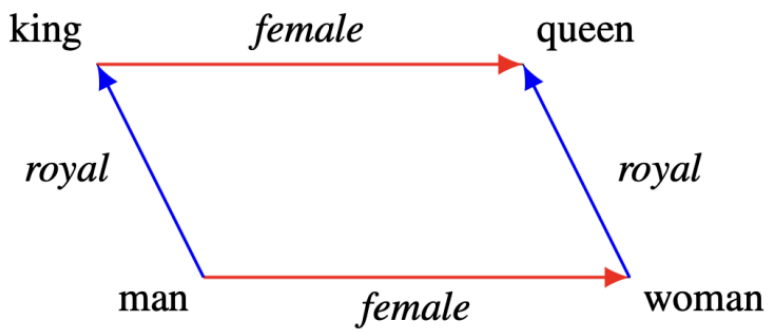

Figure 1.1: The approximate geometric relationship among word vectors for words related by analogy, in this case the concepts of "royal" and "female." Word2vec representations turn out to have a roughly linear relationship among vectors for analogous words. Figure from [19].

also arose a notion of a *basis* in this space for the features, distinct from the canonical neuron basis of one-hot vectors; the relationship between these two bases defines whether or not the feature basis is *privileged*.

The idea that semantically significant information is encoded as a direction in the representation space has many corollaries – for instance, strength of the concept corresponds to the magnitude of the projection along the corresponding direction; polarity of the concept corresponds to the polarity of this projection; and so on. These ideas are representative of a more general hypothesis underlying much of the work in interpretability – the *linear representation hypothesis* (LRH). This essentially postulates that "important" or "significant" semantic information (i.e. the information used for tasks) is linearly (or affinely) separable from vectors.

Another intuition that has been adopted towards the decomposition of representation spaces is of internal world models. Many studies [34, 3, 63] have studied whether models process data superficially, in terms of statistics, or use it to update a continuous model, and the circumstances in which this can happen. The actual information constituting a world model may be very different (a board game position, parsing information, etc.), but probing methods (both linear and nonlinear) have been employed successfully in many cases to establish such a model of the performance of neural networks.

An important recent development in the study of representations and model functioning is the emphasis on *causal relevance*. Many studies initially relied exclusively [57, 3] on simple probing mechanisms, whether linear or nonlinear, to show that certain information is "encoded" within model representations; however, it has been shown [5] that simply being able to extract information from a

representation is not sufficient to claim that it is actually relevant to model functioning. The causal role of this information in computation is, in turn, important for applications of interpretability like model control and privacy-related efforts like unlearning [49, 55, 78]. Therefore, the trend of causal results in interpretability has gained traction in recent years and is now central to the effort.

A landmark example of the applicability of causal results is model editing, which aims to alter the model's behaviour by making it 'believe' something other than what is found in its training data. The most stringent requirement for this is to change the weights of the model and have it behave according to the new specification; methods [41, 42] have been developed leveraging the causal properties of interpretability techniques to achieve this.

Another major focus of interpretability is the property of generalization, related to objectives of control and safety. As no training dataset can feasibly contain all possible future inputs, to some extent, we rely on models' capacity to generalize to unseen inputs at inference time. However, the nature of this generalization is as yet poorly understood and its failure modes are unexplained – this creates a major gap in our understanding of deep learning. One particular type of generalizing behaviour exhibited by models is *in-context learning*, or the ability of models to perform tasks based on information in a prompt and/or some examples. Many studies [43, 38] have explored the conditions for this behaviour to arise and the ways it can be controlled or modified; as in-context learning and generalization form the cornerstones of the utility of LLMs, this is an important part of the effort.

1.2 Reductionism

We have outlined the general concepts underlying the field of interpretability in the previous section. We now describe one particular approach, not particular to any one science, that has been used to great advantage in many disciplines – *reductionism*. We describe the approach and list some of its applications in this section.

Reductionism, briefly, is the idea that a system can be modelled by distilling it down to some "ideal" version of it, and then separately modelling the divergence between this ideal and the true system.

The exact nature of this ideal differs among all the applications of the method, but there are some common properties: it rarely or never exists in the real world; it can sometimes be simulated by extreme conditions; and it is defined by very few parameters. Its existence is solely mathematical, and it is created by abstracting away "distracting factors" or "noise" from a real-world system. The second step in a reductionist approach is to model the divergences caused by introducing the distracting factors into the ideal system. If all the factors have been considered and their effects modelled sufficiently well, we end up with a reliable model of the real system.

Probably the most well-known instance of reductionism in practice is the study of thermodynamics, in modelling the behaviour of matter (usually restricted to fluids, i.e., liquids and gases) under changes in pressure, temperature, and volume. The approach in this case consists in ignoring intermolecular forces and molecular geometries, and therefore defining an ideal of a substance consisting of point-sized,

un-interacting particles. Such a substance is called an "ideal gas" and its state is parametrized by three quantities – pressure, temperature and volume. The experiments of Charles, Boyle, Avogadro and Gay-Lussac all contributed to the mathematical model of such a system, which is encapsulated in the ideal gas law

$$PV = nRT$$

where R is the ideal gas constant.

Further modifications to this model were then made by accounting for the mass and volume of the particles, which are parametrized by constants a and b in the real gas law

$$\left(P + \frac{a}{Vm^2}\right)(Vm - b) = RT$$

The quantities a and b depend on the properties of the particular gas being studied, and are related to other characteristics of the gas, like the critical temperature and critical pressure.

Other modifications of the ideal gas law have also been proposed, like the Redlich-Kwong model, the Berthelot model, the Dieterici model, the Clausius model, the Virial model, the Peng-Robinson model, the Wohl model, the Beattie-Bridgeman model, and the Benedict-Webb-Rubin model. Many of these use the same parameters a and b; some use additional ones, and a few use an entirely different set. This highlights the flexibility and customizability of the reductionist approach, allowing us to leverage different aspects of the system according to requirement and obtain models more accurate for particular use-cases (depending on complexity, the range of system states, the mathematical tractability, and so on).

Another familiar example of reductionism is Newton's laws – or, more generally, Newton's model of mechanics. In essence, Newton's laws hold in cases where real-life forces like friction, air resistance, and material stress are negligible (they are nearly never truly absent); we are therefore enabled to only consider the effects of gravity, and other such simple forces, modelled by compact, elegant laws like the inverse-square law

$$F = \frac{GM_1M_2}{R^2}$$

The real behaviour of many systems is then modelled by other, independent principles for forces like friction and stress, which are then incorporated into the Newtonian model with suitable changes (e.g. the introduction of friction means that the laws of conservation of momentum and mechanical energy do not hold). Additionally, for many cases, we assume the system to be a point mass concentrated at its centre of mass, which is an abstraction ignoring the constitution of the system itself. *A priori*, such an abstraction seems almost preposterous, but it works remarkably for the majority of cases, especially in the descriptions of planetary motion. This highlights another property of reductionism – it has the potential for insight even in cases when the reduction appears simplistic.

We now turn to a simple question, the answer to which forms the rest of this thesis – what does a reductionist perspective imply for interpretability? The answer we propose herein is straightforward –

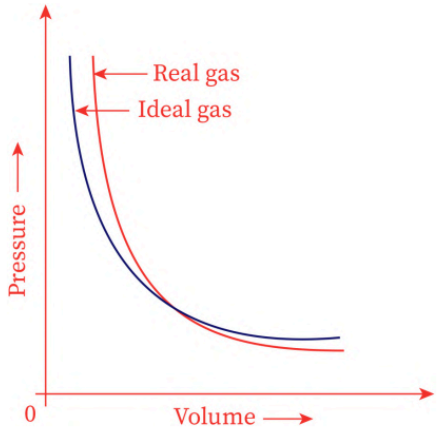

Figure 1.2: The behaviour of real gases compared to the predictions of the ideal gas law $PV = nRT$. The prediction is close to accurate for reasonable ranges of pressure and volume, but diverges more and more at extreme settings of these parameters.

synthetic data. We define data that has a clear, intuitive and interpretable generative model, and we examine the neural networks that model such data. This enables us to establish connections between the nature of the data and the internals of the networks. The expectation, then, is that the insights on the functioning of models derived from such experiments will carry over to the functioning of models in real-world settings, i.e., on natural language or images. In particular, we focus on a proof-of-concept result that allows us to assume this approach can be fruitful.

1.3 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2. Mechanistic interpretability with a focus on synthetic data. We describe the aims and approaches of mechanistic interpretability, a subfield of interpretability, and examine the results that have already been derived by the use of synthetic data.

Chapter 3. Sparse autoencoders (SAEs) and their approach to interpretability. SAEs are a recent development in the effort to understand representation spaces; we describe the intuitions and expectations of the approach in this section.

Chapter 4. SAEs on formal languages. We stress-test the SAE methodology using synthetic data, and outline the problems with SAEs highlighted by this approach.

Chapter 5. Modified SAEs. We propose a modification of the SAE paradigm based on these observations, and show that it achieves the desired results.

Chapter 6. Conclusion. We outline possible future directions of work that the reductionist approach in general, and these results in particular, can form the basis of. We also examine the limitations of the approach and caveats that have to be kept in mind.

Chapter 2

Mechanistic Interpretability and Synthetic Data

In this chapter, we go into more detail on the findings, intuitions and concepts of mechanistic interpretability. We distinguish it from other types of interpretability and explain in detail central hypotheses in the field. This allows us to situate the work on SAEs (Chapter 3) and the implications of our results (Chapter 4).
We also elaborate on the use of synthetic data, both particularly for interpretability and in the study of machine learning. Thus, we gain an understanding of the ways in which such data can be leveraged to obtain insights on the functioning of models, as we do in our work.

2.1 What Is Mechanistic Interpretability?

We have discussed in Chapter 1 the aims and techniques of interpretability. We now describe in some detail a particular taxonomy of interpretability studies, and the position of mechanistic studies in this taxonomy; we then describe some of the key results of these studies that continue to inform efforts in the field.

2.1.1 Types of Interpretability

We outline here the categorization of [6], who define 4 distinct types of interpretability studies. The distinction is based not only on the actual objects of study (inputs/outputs vs. gradients vs. parameters) but also on the perspectives taken by the studies (top down vs. bottom up). These categories are behavioural, attributional, concept-based and mechanistic interpretability. We describe the distinctions among the categories and outline key results where relevant.

Behavioural interpretability centres on only the inputs and outputs of models. In other words, it treats the model as a black box, and is therefore model-agnostic: it does not depend on any particular architecture, paradigm, or even the cognitive nature of the model. Thus, it has distinct advantages over other types of interpretability in certain cases – significant such cases are ChatGPT [51] and GitHub Copilot, which are closed-source proprietary models that are in common use [24] for several sensitive

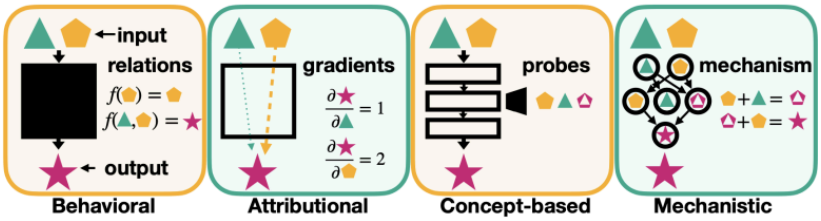

Figure 2.1: Types of interpretability. **Behavioral** analyzes input-output relations; **attributional** quantifies individual input feature influences; **concept-based** identifies high-level representations governing behavior; **mechanistic** uncovers precise causal mechanisms from inputs to outputs. Figure and caption from [6].

applications. As these studies are completely independent of model internals (either model parameters or gradients), they employ a distinct set of techniques, intuitions and axioms from those relevant to us; we therefore point the reader to [6] for a brief description of some studies in this direction.

Attributional interpretability studies, by contrast, do factor the inputs and outputs of models into their considerations, but not exclusively – they focus on the *relationship* between the inputs and the outputs. In other words, they aim to, for instance, isolate what parts of the input are salient to the model's computation. An important application of such methods is bias detection – if it can be identified that, say, a hiring system focuses on the gender of the applicant for a programming role, this constitutes an undesirable bias and it can be mitigated. They are also important for validation; for instance, a medical diagnosis model can be verified to be using only relevant aspects of the input, rather than relying on inaccurate heuristics. Such methods are, consequently, not entirely black-box oriented, since they depend on the numerical nature (continuity) of inputs and outputs; however, they require little to no information about the architecture of the models beyond this, and the wide proliferation of architectures in deep learning means that such methods are valuable in their generality.

One major development in the attributional approach to interpretability is the integrated gradients method [66]. This method defines a consistent, implementation-independent way to compute the dependence of the outputs on particular features of the input. The general attribution problem is defined for a given input and a baseline – we ask what features of the input caused its prediction to change with reference to the baseline. This method is operationalized by taking a path integral of the gradients from the baseline to the given input for each feature.

It satisfies many desirable axioms. For instance, it does not attribute relevance to features that the model does not depend on; it preserves linearity; it completely accounts for the change in the output; and it is

invariant to the actual architecture of the model. This highlights the main features of the attributional interpretability paradigm that we have listed above.

Concept-based interpretability is the most akin to mechanistic interpretability in terms of their assumptions and the objects of study – they both rely on the model's internals. However, concept-based interpretability primarily focuses on the learned representations, exploring the encoding of interpretable concepts and operations in these representations; it emphasizes the connection to the model's end-to-end functioning in any exploration of its intermediate representations. In this sense, it is a top-down approach to interpretability – it is this which sets it apart from mechanistic interpretability (which we discuss below). It examines patterns and correlations in representations, without necessarily hazarding hypotheses of the functioning of *model components*. Concept-based studies have also in recent years moved into causal manipulations and the applications of concept-based interpretability in model control. Several major studies in concept-based interpretability have made remarkable contributions to an intuitive understanding of model functions. Early studies in this direction have centred on probing the learned representations, primarily through supervised auxiliary classifiers, or probes. [5] provides a comprehensive overview of the achievements and shortcomings of this endeavour, which we briefly summarize. One of the most salient features of the approach is that probes have a wide variety of parameters that have to be carefully selected to ensure validity of the effort – the complexity of the probe (linear vs. MLP); the metric by which to measure the probe's success; the baseline values for this metric; and so on. Additionally, they require a predefined ground-truth, generally requiring a large-scale annotated dataset, which makes them sometimes infeasible. Despite these shortcomings, probes have consistently and successfully been used to give evidence for the presence of syntactic and morphological information, like parts of speech, parsing relations, semantic similarity [58, 70, 4], and so on.
Furthermore, initial studies utilizing probing were essentially correlational in nature – they were seen as measuring mutual information between an interpretable ground-truth variable and the representations [54], and made no guesses as to whether the information identified by the probe is actually relevant to model computation. The identification of this research gap led to an effort to improve and evaluate probes causally – [17] show that, for instance, projecting out (i.e. nullifying) the directions identified by probes as containing information may not affect the performance of the model; and [56] show that a similar approach can be used to mitigate bias and improve fairness by eliminating dependence on spurious features. These directions of concept-based interpretability are called *representation engineering*.
Another direction of research in concept-based interpretability has focused on identifying the ways in which language models store and retrieve knowledge in a factual sense. For example, [12] describe a method, called contrast-consistent search, to identify the knowledge encoded in models *distinct from the facts contained in their outputs.* This method relies on the idea that if a model has some internal representation of truth, it must be consistent across logical operations like negation; thus, it finds directions in the latent space that satisfy this constraint. This is an important example of what has come to be known as the linear representation hypothesis (mentioned in Section 1.1) – the method is built on crucial assumptions of linearity's inherence to the representation space, like the idea that probabilities can be

extracted by projections along a certain direction.
In the same vein, [41] aims to causally manipulate the knowledge stored in language models, specifically in a factual recall setting. The method, named ROME, aims to find a way to localize and edit simple relational facts within GPT models, like "Paris is the capital of France." This is achieved by building on the intuition described in [22], which reduces the MLP layers of transformers to memory systems based on key-value association – thus, ROME defines a new key-value pair to 'overwrite' an existing one, thus engineering to model to "believe" a new fact, like "Paris is the capital of Germany."
This work is especially relevant to us as it bridges many aspects of interpretability, in the taxonomy being outlined. We note, firstly, that the study uses attributional interpretability to identify the exact module of the transformer that needs editing, and that it leverages existing facts from prior work on the properties of this module. Fundamentally, moreover, it relies on ideas of linear representation and separability that arose out of concept-based interpretability studies.
Most centrally, the authors ablate on representations to identify the exact hidden state which is causally relevant to a certain fact – this degree of causal analysis being the hallmark of mechanistic studies more than any of the types we have already considered. ROME (and its successor [42]) takes this one step further by proposing a way to modify the weights of the model themselves, in order to control its notion of factuality and truth. While this does not directly leverage mechanistic concepts like circuits, it is still an important unification of the principles of the different categories we have seen.

Mechanistic interpretability has much in common with concept-based interpretability. Most saliently, as we have remarked above, their objects of study coincide – they both examine model internals to obtain an explanation of model functioning. However, the major distinction is in the perspectives; while concept-based interpretability adopts a top-down approach, aiming to find a connection between inputs/outputs and particular parts of the model, mechanistic interpretability aims for a bottom-up understanding of model functioning, integrating all the individual components of a neural network in a complete explanation. In this sense, the objective of mechanistic interpretability is a fully reverse-engineered model – a model whose operation can be completely expressed as an abstract set of computations defined in human-understandable terms. The focus that mechanistic interpretability places on submodules of neural networks and their roles in computation means that it takes the causal efforts of concept-based interpretability one step further; mechanistic works completely eschew correlational approaches in favour of interventional observations in identifying model operations.
We describe key concepts and theories of mechanistic interpretability, and associated works, in the following sections.

2.1.2 Concepts and Hypotheses

One of the central, and oldest, concepts that form the basis of mechanistic studies of interpretability is the notion of *features*, the fundamental units of computation in neural networks. The exact operationalization of this concept in terms of model internals, however, has not seen consensus by the community.

Here, we discuss some ideas of what features could be mechanized as, associated concepts that arise from these ideas, and what these theories imply for our understanding of the functioning of deep models.

The most natural starting place is, perhaps, the neuron as a fundamental unit of computation. Neural network architectures have been built from the ground up on the intuition that each neuron carries out some form of semantically significant computation, and while their *a priori* interpretability value has dropped in the age of large-scale transformers, the idea that they could each be independently meaningful has been explored by several works.

[18], a work which laid out many of the ideas and hypotheses we go into here, characterize this notion by distinguishing *privileged* and *unprivileged* bases. This rests on a conceptualization of the representation space in a linear-algebraic sense, where each neuron represents a one-hot vector and the set of neurons then represents a "canonical" or "predetermined" basis set for that space. If, then, these neurons correspond to meaningful steps of computation, the basis is considered 'privileged' from an interpretability point of view. We consider the implications of unprivileged bases below. Mathematically speaking, the use of element-wise nonlinearities (like ReLU, sigmoid, and tanh) provides ground for thinking of representation spaces in terms of neurons and the privileged basis, because the very definition of these operations depends on the basis being used.

$$\text{ReLU}(x) = \max(0, x)$$
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Many studies, then, have examined the implications of the idea of computationally meaningful neurons. [46], for instance, devise a method to explain neurons in deep models by searching through a well-defined, compositional, explanation space, which can be defined for both image- and text-oriented tasks. Their results show that many neurons in image classification and NLI models correspond to shallow statistical heuristics that originate in dataset biases, and they use this information to design adversarial samples that highlight models' blind spots.

Another neuron-oriented study is [9], in which the authors aim to scale and generalize the pipeline of neuron explanation. To this end, they leverage the (at the time) recently developed capabilities of LLMs themselves – they use one model to find the common factor among a set of "activating samples" for a certain neuron (pieces of input on which the neuron has a high value), and then use another model to generate new samples based on this, and verify that all the newly generated samples also activate the neuron. The idea is to ensure that the explanation finally generated is both complete (everything that activates the neuron fits into it) and sound (everything that fits into it activates the neuron). This proposed pipeline, with minor modifications, has formed the basis of automated correlational explanations for model components (not just neurons; see below) for many works [20, 26, 11, 80] since.

However, it has been observed in multiple studies that neurons are not *monosemantic*, i.e., they appear to correspond sometimes to unrelated, independent aspects of the inputs. An important example of this is

[50], which examines neurons in image models. This work relies on explanations defined by maximally activating samples, or images in the dataset that activate a given neuron the most. The authors then find that many neurons in large vision models, like InceptionV1 [68] are polysemantic. For example, one neuron represents the fronts of cars and cat faces. [46], cited above, also finds that 31% of the neurons in the image classification model they inspect are polysemantic. In the case of language, [23] shows that neurons in transformer-based LMs also employ superposition, especially in early layers, to a great extent. They use probes trained with a sparsity constraint to identify and explain neurons, and find that those in early layers frequently represent collections of unrelated local lexical heuristics.
[18] explores the phenomenon of superposition from a mathematical perspective, examining small models trained on an autoencoding task. They show that adding a nonlinearity to a network enables it to represent orthogonal input features in non-orthogonal directions in a hidden space, creating a new understanding of polysemanticity – it comes from *superposition* in the representation space. They also present an intuition of the connection between correlations in the dataset and the directions that features are represented in – correlated features are orthogonal directions, while anticorrelated features are antipodal directions. Notably, this work also explores the idea of privileged and unprivileged bases, showing that a hidden state with a nonlinearity preceding it has a privileged basis due to the elementwise nature of the operation.
[50] also explores the phenomenon of superposition from an "efficiency of storage" point of view. They remark that a monosemantic neuron in later layers "spreads out" its information over unrelated neurons in order to "conserve" the space that this information would otherwise take up if it had a dedicated neuron.

An intuition towards superposition that has seen expression in multiple works [11, 18] is to hypothesize a "virtual" disentangled model, which represents the model that "could have" been learnt if it had enough space for all the information. The model that is actually learnt, by contrast, is defined by collapsing multiple features of this virtual model into single features through heuristics and correlations (i.e., shallow statistics). This virtual model has been called a "disentangled" model, because in comparison with the actual learned model, it disentangles polysemantic neurons into monosemantic, interpretable neurons.

In the same vein, therefore, there has been a concerted effort in recent years to link information-theoretic concepts of disentanglement to interpretability, and create disentangled representations in neural networks. A major thread of this effort is the sparse autoencoder approach [11], which involves training a simple two-layer model *on the hidden representations of a model*, on an autoencoding task (predicting back the input). The hidden layer of this autoencoder then represents a kind of decomposition, in affine terms, of the representations in question. An additional constraint of sparsity implemented during the training of the autoencoder ensures that a small number of vectors is learnt for each sample. We examine this method in greater detail in Chapter 3, and the limitations in it that our work reveals in Chapter 4.

An important hypothesis underlying much of the effort to disentangle models, or more generally to solve superposition, is the *linear representation hypothesis*. This hypothesis essentially suggests that features correspond to directions, or unit vectors, in the representation space, and these directions are the interpretable units of model functioning. These directions are also frequently operationalized as *linear combinations of neurons*. The intensity and polarity of concepts for a certain representation easily translate to the magnitudes and signs of the vectors in certain directions (i.e. their projections). The combination of concepts, under this framework happens additively, preserving an essentially linear 'semantic space.' Furthermore, superposition is conveniently explained by the misalignment of these directions with the neuron basis. The additive properties of word vectors that we have noted in Chapter 1 [45] were the earliest piece of evidence towards this picture of model interpretability, and there have also been successful studies utilizing linear probes [2]. Furthermore, the recent efforts towards *model steering* – controlling a model by adding a vector to its internal representations – have shown promise [73, 35], lending more credence to the theory. [11] also describes the influence of the linear representation hypothesis on the SAE effort.

Lastly, we examine *circuits*, a notion central to the causal efforts of interpretability. Essentially, quoting [50], a circuit is a connection among features that implement "meaningful algorithms corresponding to facts about the world." That is, circuits are the lowest levels of computation that use features as their primitives, much like, say, assembly uses memory locations, and probability theory uses random variables. What is this "connection" among features? It is represented in models by weights, which form the bridge between representations. In this framework, therefore, [50] outlines several circuits in vision models based on intuitive ways to identify objects – for example, the feature to identify cars builds on features that look for windows near the top of the image, wheels near the bottom, and so on. Mathematically speaking, therefore, circuits are *subgraphs of the computational graph* of a neural model. [6] describes circuits as the computational primitive of deep models, in the way that features are the representational primitive. [44] present a theoretical model of learning dynamics organized around a generalized conception of circuits.

2.2 Synthetic Data

Having described in detail the background and progress of mechanistic interpretability, we now turn to the other main pillar of this thesis: the use of *synthetic data*. In this section, therefore, we examine the various ways in which synthetic data has contributed to the effort of interpretability. Some of these studies are not mechanistic in the ways described above, but they are nevertheless important for an understanding of the possibilities of this direction.

We define synthetic data as data generated computationally, based on a fixed, rule-based algorithm. This can be either deterministic (like the modular addition example below) or probabilistic (like the use of sprites in vision below). Thus, it can represent data across domains and modalities, and we will see evidence of this variation in this section.

This section is divided into two parts. The first is organized as a list of case studies, in which we examine some insightful works that leverage synthetic data in careful ways to present evidence for a certain theory of model functioning. In the second section, we examine a specific type of synthetic data that is relevant to us – formal languages – and the ways it has been and can be used in interpretability.

2.2.1 Works Using Synthetic Data

We first consider a work that we have already cited above. [18] uses entirely synthetic data to provide, as we have seen, a rich array of theoretical support for the superposition hypothesis and possible explanations. The data in this case is simple 5-element vectors, randomly generated, and the task is a straightforward autoencoding task. The scope for the insight, despite the apparent simplicity of the data and task, comes from the architecture – the models are restricted to a two-element hidden space, and the loss function assigns an importance score to each feature in the reconstruction – thus, the authors examine the impact of the importance of a feature in connection with how it is encoded in the hidden space.

The key feature of this study, then, is how it manages to *isolate* aspects of neural models that are responsible for certain types of functioning. For instance, one of the main results is that a non-ReLU model does not exhibit superposition, but once the ReLU in the hidden layer is added, it is able to encode features in superposition, and does so in varied, complex ways. Thus, the fundamental role of the nonlinearity in enabling superposition is empirically substantiated by this study, in a way that an experiment with other confounding factors (like deeper models or more complicated data) would not be able to categorically identify.

Another important study that leverages toy models – small models trained on simple tasks, often via synthetic data – is [48]. In this work, the authors train a one-layer ReLU transformer to perform modular addition, in which the input consists of two numbers a and b, and the output is $(a+b) \mod p$, where p is fixed. This work is significant in that the authors completely reverse engineer the model that they train on this task, i.e., they are able to outline a formal algorithm to achieve the task and show how the model implements this algorithm through evidence (both correlational and causal). The algorithm identified is based on trigonometric identities, in a way that leverages their periodicity to produce the intended modularity, which the authors call *Fourier multiplication*.

The authors present plenty of evidence in support of this claim. The observations that prompt this intuition are mainly correlational in nature – for instance, the embedding matrix appears to map inputs to their sin or cos values for various frequencies (dubbed *key frequencies*), and MLP layers appear to compute sums and products in simple ways. However, once the algorithm is identified, causal evidence, obtained by ablations, is used to complete the picture – individual components of the model are replaced by what we predict based on the Fourier multiplication algorithm, and this does not harm (indeed, sometimes improves) model performance.

We now consider a more complex form of synthetic data. [34] consider the question of whether neural networks form meaningful "world models" within their representations, or just memorize a collection

of surface statistics, a dichotomy which has long plagued the study of deep learning models. They operationalize this distinction through a unique form of synthetic data – representations of the board game Othello. They train a model to predict the next move given a textual representation of the current board state, and examine the nature of the model's internal representations. Specifically, they ask whether a representation of the current state of the board can be uncovered from the internal representations, or moves are statistically predicted simply based on the previous string of moves with no deeper process behind it.

The authors find that the model does in fact have an internal representation of the board state. This is revealed through probes that search for information corresponding to the state of each square on the board, which can only exist if there is a continually updated "state representation," and would not be found if the model relied on simple surface heuristics. This representation is shown to be causally relevant to the predictions of the model through interventions that alter the board state in predictable ways.

Notably, although the board representation appeared first to be encoded nonlinearly, a later work [47] showed that it was in fact linear. The distinction lay in the way the probes were trained – the initial work trained probes to predict whether a "black" or a "white" piece was in a particular position on the board, but the representation actually switched its perspective of the board at every step. Thus, rather than consider each piece as black or white, the distinction was really "mine" or "not mine." This subtle but important change to the probing framework made it clear that the representation was in fact linear. These studies highlight a significant limitation of the probing approach to interpretability – the need for *a priori* gold labels.

[63] also examine a similar question through a related approach – they consider transformer models trained to solve mazes. These mazes are represented textually, as with [34], and the model produces tokens that correspond to path following. The authors find, in this case as well, that *each token* contains a faithful encoding of the topology of the maze, and the possible paths that exist. Furthermore, the authors present an understanding of attention heads relevant to the path-finding algorithm, thus identifying a partial circuit for the task.

In the image domain, one example of synthetic data is [36]. This work aims to survey recent progress towards learning disentangled representations in an unsupervised manner, which (the authors show) was based on unsound or implicit assumptions. In addition to a result showing the theoretical unattainability of true disentanglement, the authors support their claims with an experimental study on synthetic data consisting of objects with deterministically or stochastically defined colours, shapes, and backgrounds. They use this dataset to establish the dominant role of inductive biases in the disentanglement of internal representations, and measure the agreement among several existing metrics for disentanglement.

This study exemplifies an important use case of synthetic data, one which we will leverage more in Chapters 4 and 5 – as a *testbed* to verify wide-ranging empirical claims. As it is easy to generate clean and interpretable synthetic data on large scales, it becomes an attractive way to test hypotheses and stress-test results on general model behaviour.

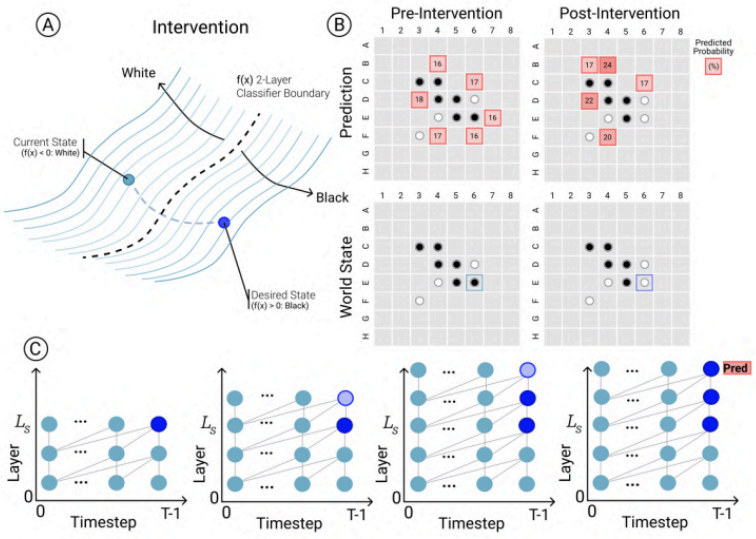

Figure 2.2: A schematic of the interventions performed by [34]. **(A)** shows how they modify the board state so that the it crosses the decision boundary from *white* to *black*; **(B)** shows the effect of the intervention on the prediction of the model, corresponding to the different predictions of white and black tiles; and **(C)** shows the way in which the intervention is carried out in all layers from L_S (where the information is found) until the end. Figure reproduced from [34].

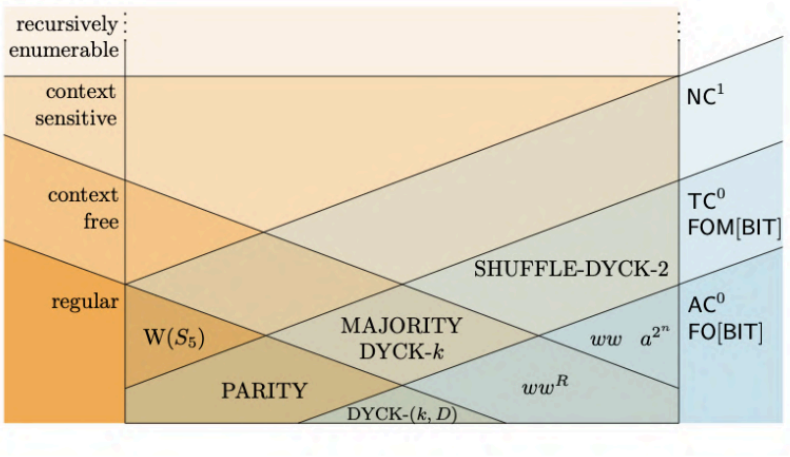

Figure 2.3: The relationship among the categories defined by [65] and the Chomsky hierarchy, with some common languages and their places shown. Figure reproduced from [65].

2.2.2 Formal Languages

We now examine the use of formal languages, a specific type of synthetic data, in interpretability. Note that, in a computational sense, there is no definitional distinction between formal languages and any type of synthetic textual data, not even board game configurations or modular addition; however, we use the term in an informal sense to refer to data generated through some established grammatical framework, usually PCFGs. Additionally, we consider the data a formal language if it is generated deterministically but resembles natural language in some form.

Formal languages have not seen extensive use in interpretability *per se* until recently. In fact, most studies that have used formal languages in machine learning have aimed to characterize the *expressivity*, or limitations, of these models [65].

For instance, [8] formalize the difference in capabilities between transformers and LSTMs, the most popular method of sequence processing in the pre-transformer era, through a well-defined category of "counter languages." They also use formal languages to isolate the role of positional encoding and self-attention within transformers. There have also been attempts to extract formal grammars from neural networks in a scalable manner, aiming to recast their functioning in terms of well-understood formal frameworks like CFGs [79] or RNNs [76].

Within interpretability, however, there has been a growing interest in the potential of formal languages. For example, [3] explore the mechanisms that transformers – both encoding and decoding models – use

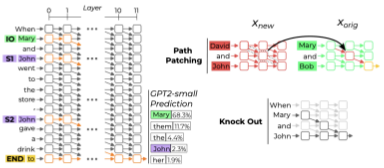

Figure 2.4: The circuit for indirect object identification in GPT2-small. The left shows the circuit itself, implemented using attention modules at each layers, which move information towards the end; and the right shows the interventions used to discover and validate the circuit. Figure reproduced from [75].

to parse formal languages, specifically CFGs, and show that the hidden states and attention patterns correspond to well-known formal parsing algorithms. They also use this setup to characterize the role of positional encoding and noisy input data.

[38] is another work that uses formal languages to understand models better, in this case examining the training dynamics of transformers. We have seen above some evidence that transformers trained on language modelling learn to define a "state representation," or a "world model," in their internal representations, which is used for specific downstream tasks; this work, then, models the *emergence* of these representations – their occurrence after a certain point in training – through a *percolation model*. Essentially, the authors study a transformer trained on a context-sensitive formal language, and characterize the shift in the model's performance on narrow tasks as a phase transition reflecting the changed structure of internal representations.

Another task that is akin to formal languages, but in many ways straddles the boundary between formal and natural languages, is *indirect object identification*. This consists of completing sentences like "John and Mary went to the bookstore; John gave a book to _____." As this kind of data can be deterministically generated, and conforms to a more or less rigid set of templates, we consider it here. Essentially, we consider it a type of synthetic data with the appearance of natural language, and so we examine it under the same heading as formal languages, but it is not really one. [75] use this task as a setup to study the functioning of GPT-2 small, and in fact identify a complete explanation in the form of a circuit that is used to carry out this task. They provide correlational and causal evidence for each component of the circuit, and evaluate their explanation on faithfulness, completeness, and minimality.

Thus, in conclusion, we have situated mechanistic interpretability within the wider study of interpretability of deep models, and we have outlined important concepts, hypotheses, and insights that have defined this field in recent years. We have also laid the foundation for an understanding of the role of synthetic data in interpretability, which the main work of this thesis (Chapter 4) is based on. In the following chapter, we conclude our survey of the field with an in-depth look at sparse autoencoders.

Chapter 3

Sparse Autoencoders

In this section, we conclude our review of the field by examining in detail the context, results, metrics, and insights of the sparse autoencoder (SAE) paradigm of interpretability. First, we discuss the foundations and the specification of SAEs. We then move on to the disentanglement perspective of SAEs, and discuss existing work on the feasibility of the approach from this perspective.
Thirdly, outline some of the key results on the inductive biases of SAEs, which continue to inform much of the work in the area. This includes qualitative results on the nature of the features, scaling laws, and other such observations on the method. Next, we consider the metrics by which SAEs are ranked and evaluated that are common in the field.
We conclude by an examination of a small set of features reported in some studies, and the causal nature of these features.

3.1 What Are SAEs?

3.1.1 Architecture

As we have seen in Chapter 2, SAEs are a method of interpretability centred on *disentanglement*, based on the linear representation hypothesis. The basic idea of SAEs is that internal representations of deep models can be analyzed as the sum of a fixed set of *atoms*, which are themselves vectors in the representation space, and that these atoms can be identified in an unsupervised manner. Furthermore, these atoms are expected to be *interpretable*. Note that these atoms are, ideally, *features* in the sense that we have already seen.

3.1.1.1 Reconstruction

This is operationalized via an autoencoder setup, which we have already seen (e.g. in [18] in Chapter 2). An autoencoder is a model trained to replicate its input, and, as we saw with [18], the value of an autoencoder lies not in its actual output but in the intermediate representations that it learns. In the SAE paradigm, we use autoencoders consisting of a single hidden layer, which is set to have a much

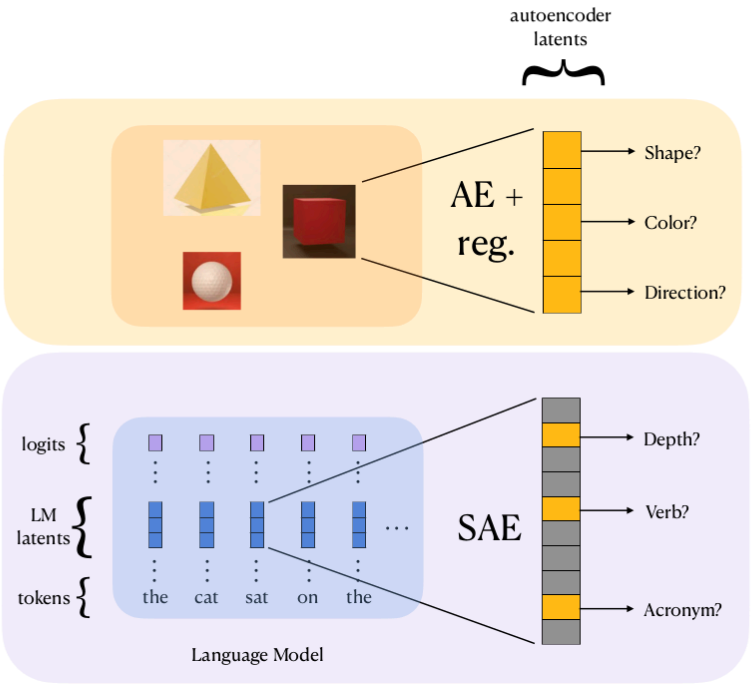

Figure 3.1: The autoencoder paradigm for interpretability. Autoencoders have formed the basis of approaches to disentanglement in the vision domain [25]. Utilizing synthetic testbeds, prior work has shown several limitations in this general pipeline [36, 72]. While SAEs have similarly been used to disentangle hidden representations of language models for interpretability, we aim to perform a similar study as ones in vision to understand the (in)abilities of SAEs.

higher dimension than the input/output vectors – it is intended to be an *overcomplete basis*. The intuition is that the analyzed representation space uses superposition, and therefore actually encodes many more features than what its dimension suggests.
Thus, the basic problem we are aiming to solve here is *reconstruction*; for a given vector, we must find coefficients of a linear combination of a fixed set of atoms to recreate the input.

3.1.1.2 Sparsity

However, an immediately visible issue with this approach is that it quickly becomes possible to fall into a trivial solution of a set of one-hot vectors, which can then naturally be combined to reconstruct any vector. This gives us no information about the *semantic space* of the representations, but rather becomes about the geometric space that they inhabit.
The workaround for this is to introduce sparsity – each vector is restricted to be decomposed into a fixed number of atoms. Thus, it becomes a problem not simply of reconstruction, but of economy: the model needs to learn a set of features such that as few as possible suffice to reconstitute the input.

This sparsity is operationalized in several ways. Here, we discuss two common methods of introducing sparsity, both of which have been relatively popular in prior work. The first is a standard regularization method, which consists of adding a term to the loss, which when minimized is intended to achieve sparsity. This is the method adopted by [14, 11], who add to the loss a term equal to the L_1-norm of the hidden representation of the vector; the L_1 norm is selected due to its tendency not to blur, or average, the way L_2-norm does.
Another method that was popularized by [21] is a *top-k activation*. In this method, the ReLU commonly placed after the first linear layer (the encoder) is followed by a top-k activation, which selects the k largest elements of the vector and zeroes out all other elements. As this comes after a ReLU layer, the activations are all initially nonnegative, and we end up with a k-sparse set of coefficients for a convex combination of atoms.

3.1.2 Feature Identification

We have seen, then, how SAEs learn to decompose the representation space into a set of atoms. These atoms are represented by the columns of the matrix that forms the second half of the SAE, i.e., the decoder. To see how this interpretation is valid, let the reconstruction $\hat{x}$ be generated from the hidden representation h through the linear transform D as

$$\hat{x} = h \cdot D.$$

By the properties of matrix multiplication, this is equivalent to a linear combination of the columns of D where the coefficients are defined by the elements of h.

However, in what sense are these atoms interpretable? What do they represent? This part of the paradigm is operationalized by *feature identification pipelines*, which are of various types. We briefly describe some of these ways here.

[11] uses a feature identification pipeline that starts with manual inspection of each feature. This involves examining *positive samples* (sequences and tokens on which this feature activates) and *negative samples* (those on which it does not activate) and hypothesizing an interpretable description that accurately predicts both types of samples. The need for both samples of both types comes from a requirement that the explanation be not only complete (it should correspond to samples which do activate the feature) but also sound (it should *not* correspond to samples which do not activate the feature).

However, this method suffers from its labour-intensive nature. [9] introduce a more scalable method – *automated interpretability*. This pipeline consists of two LLMs, which we refer to as an *explainer* and a *simulator*. For a given feature, the explainer model takes samples that activate it (positive samples in the terminology we use above) and hypothesizes a description that accounts for all these features. Now, this explanation can be reasonably expected to be complete, but it may not be sound; in order to ensure this, we use the simulator model to "estimate" activations on multiple sequences based on the hypothesized description given by the explainer. If the explanation is sound as well, the simulator model guesses zero activation for the truly unrelated samples, and we can then use this score as a measure of how good the feature is.

Note that these, and related methods, are *unsupervised* methods for finding feature explanations – they do not have a predecided set of interpretations to select from. They must generate complete natural-language descriptions based on data. In Section 3.4 below, we describe some ways of finding and evaluating features under both unsupervised and supervised settings.

3.2 Disentanglement and Interpretability

We have seen the theoretic and implementational foundations of the SAE paradigm. Now, we examine in more detail the implications of SAEs as a *disentanglement* method, in view of the extensive literature on this topic. We first examine some studies on what it means for disentangled models to be identifiable, what results have been put forward in this direction, and what these results mean for SAEs. We then present two case studies for how SAE features have formed the basis for insights in the functioning of deep models on a larger scale.

3.2.1 Disentanglement

[36], working in the image domain, carry out a wide-ranging empirical and theoretical study of VAEs on synthetic image data, and prove a non-identifiability result on the learned latent representations. In other words, infinitely many possibilities for disentangled latent representations exist if only a data reconstruction objective is used, and so the disentanglement of the actual representations learned is

extremely sensitive to the inductive biases of the autoencoder being used. Thus, no guarantees about the interpretability or task-specific usefulness of the learned representations can be assumed. [36] also note that identifiability results are in general not obtainable for the case of a nonlinear data-generating process [28, 27, 29]. Furthermore, [72] observe that real-world data can only be generated from highly correlated latents, possibly with a complex causal relationship. They prove also that disentangled representations do not represent an optimum in this case, and so entangled representations are learned. However, they also note that supervision can be leveraged to achieve true disentanglement – auxiliary data linking priors to observations can be used for weak supervision during training to resolve latent correlations. Later works outlined settings in which identifiability results can in fact be proven. For example, [32] propose a bi-level optimization problem, where the representations are optimized for reconstruction as well as performance on downstream tasks via sparse classifiers, and [64] demonstrate how to achieve disentanglement with interventional data (data from observations with individual latents ablated).

3.2.2 Interpretability

Even though disentanglement has been shown to theoretically be unidentifiable, empirically, SAEs features have been applied in 'downstream' theories of the functioning of deep models, highlighting the fact that they could be the source of many useful insights if they can be made theoretically more robust. For instance, [15] train SAEs on the residual stream of a large language model (Llama-3-70B, to be specific) that performs reinforcement learning problems in-context. These SAEs reveal features that closely correspond to the errors in temporal difference learning [67], an iterative reinforcement learning method. These representations are also causally implicated in the computation of Q-values, showing that the model indeed implements this RL method to achieve in-context learning.
[33] is another work that exemplifies the potential of SAE features in downstream results. This work examines the *feature universality hypothesis*, which suggests that large models trained on large natural language datasets are converging towards the same feature sets. They explore this by comparing the decomposed representation spaces of SAEs of each model, rather than the models themselves, in order to avoid the challenges posed by polysemanticity and superposition. The high degree of similarity that these spaces show, then, presents new evidence towards the hypothesis.

3.3 Scaling Laws and Inductive Biases

Another aspect of SAEs that has received much attention in the literature is the inductive biases of the paradigm. As SAEs are defined and implemented themselves using deep learning techniques, they beg the same question that pervades much of deep learning architectural research – what does the nature of the model mean for its behaviour? In other words, what are the inductive biases of the SAE paradigm? For example, [36] highlight the importance of drawing attention to the inductive biases of autoencoders while using them to achieve disentanglement. They show that the objective function and random seed

together are responsible for roughly 80% of the performance of VAE encoders, demonstrating the lack of robustness in the method.

In a related vein, much work has been done on the effect of hyperparameter selection on the effectiveness of SAEs. In particular, there are many empirical results on the relationship between the dictionary size (i.e., the latent size of the autoencoder) and the features learned. For example, [60] (working with numerical data) note that recovering the ground-truth features requires a dictionary size of 1-8× the size of the input, and that if sparsity is enforced by L_1-regularization, then larger dictionary sizes need larger penalties.

Other studies have found that "dead" features (i.e., features that don't activate on any sample) begin to occur from a dictionary size of about 4x (Cunningham et al., 2023), that a single feature in small SAEs "splits" into several features (whose union represents the former feature) in larger SAEs [39, 11], and that several features are simply the same token in various contexts, like a physics "the" and a mathematics "the" [11].

3.4 Metrics

As with any interpretability (or even ordinary deep learning) paradigm, the discussion on metrics has not been ignored for SAEs. Many aspects of SAEs have been identified as important for evaluation, and many metrics exist for each of these.

Mainly, they can be classified along two axes: interpretability vs. disentanglement, and supervised vs. unsupervised. Supervised metrics require some ground-truth dictionary of features to evaluate against, which is generally assumed to be human-interpretable. Therefore, interpretability and disentanglement are tied together in these metrics.

For example, BetaVAE uses the accuracy of a classifier trained to predict ground-truth features from learned ones [36, 59]; consistency and restrictiveness measure the sensitivity of ground-truth features to learned ones [61]; and maximum mean cosine similarity (MMCS) maps the two sets of features using the cosine similarity [60].

However, when no ground-truth is available, a feature set may be disentangled but not interpretable, or vice versa. Thus, unsupervised metrics evaluate interpretability and disentanglement separately. Examples of metrics for interpretability are controllability, which evaluates how 'easy' it is to control the model output by intervening on a feature set [39], and next logit attribution, where the causal role of the feature in the model's final logit output is examined [11]. Notably, [39] find that SAEs trained on task-specific data (indirect object identification in their case study) learn meaningful directions, while those trained on full data perform on par with those that have random directions kept frozen through training.

	Interpretability	Disentanglement
Supervised (requires FoV)	**MMCS** **BetaVAE, FactorVAE** MIG Modularity DCI **SAP Score** 3CharM **Consistency/restrictiveness** **MCC**	
Unsupervised	**Logit attribution** **Explanations** **Ablations** **Controllability**	**Necessity/sufficiency** **Faithfulness/completeness**

Figure 3.2: The metrics developed in various papers to evaluate SAEs for interpretability and disentanglement. The names in bold represent metrics applicable to SAEs trained on text models.

3.5 Correlational and Causal Features

The research on SAEs, however, despite its scope and depth, has not yet settled on its relationship with correlational interpretability. Many results and insights on SAEs rely on correlational observations with little causal backing, but the very nature of causal evidence renders it liable to cherrypicking.

It is undeniable that a number of studies demonstrate the causal effects of SAE features. For example, [11] show that the features representing Arabic-language text (in a one-layer transformer) can be clamped to a high value, increasing the probability of generating Arabic text. [69] scale up this work to analyze Claude's representations, identifying a feature representing the Golden Gate Bridge in San Francisco, with a causal effect on how prominent this monument is in the model's output. Notably, [40] use SAE directions to identify circuits in models (across layers) responsible for specific tasks, like subject-verb agreement across relative clauses.

However, we explore the general property of causality in SAE features in our work Chapters 4 and 5. The bulk of features have no causal effects on the model computation (using standard SAE protocols). We claim similar results can be easily demonstrated in natural settings as well—for instance, [13] report results qualitatively similar to ours in a restricted natural language setting as well.

In summary, therefore, we have examined in depth the foundations, insights, and results of the sparse autoencoder paradigm. To some extent, we have aimed to highlight its particular assumptions and make them explicit, and draw attention to the outstanding research gaps in the field. These are the very gaps which we aim to address through our experiments and proposals, which we outline in detail in the next two chapters.

Chapter 4

A Formal Testbed for Evaluating SAEs

We have seen in the previous chapters the development of interpretability in general, and mechanistic interpretability in particular, and the wide variety of approaches and techniques that have been leveraged towards this progress. We have also seen the application of synthetic data in the context of this development.
The evolution of the SAE-based interpretability paradigm has also been influenced by the use of synthetic data [36, 72, 32, 37, 31], from both empirical and theoretical perspectives, However, these studies have been confined thus far to the vision domain, and the lack of corresponding results for SAEs in NLP interpretability is conspicuous.

Aiming to provide proof-of-concept for the application of reductionism in NLP interpretability, through the abstraction of formal languages, we outline in this chapter and the next a synthetic testbed to stress-test the SAE paradigm in this domain. We first describe the SAE variants and the formal languages that we focus on, and then we lay out results in evaluations corresponding to robustness, interpretability, and causality. These results are both qualitative and quantitative in nature, and line up with prior results in vision.

4.1 Setup

In this section, we describe all aspects of our setup. First, we describe the data generation process (including the formal grammars and their specification) and specify the models *on which* we run our studies (transformer-based language models). We then list the variants of the SAE architecture that we compare and their specifications.
Lastly, we describe the experimental setup for the three classes of results that we present: particular interpretable features, robustness to hyperparameters, and causal validity.

4.1.1 Data and Models

4.1.1.1 Data Generation

The formal languages we work with are probabilistic context-free grammars (PCFGs), which are generated by starting with a fixed 'start' symbol, and probabilistically replacing nonterminals according to production rules. We work with three PCFGs, intended to represent levels of complexity (in parsing and generation).

In order of increasing complexity, the languages we consider are `Dyck-2` (the language of all matched bracket sequences with two types of brackets), `Expr` (a language of prefix arithmetic expressions), and `English` (a simple fragment of English syntax with only subject-verb-object constructions). We describe below the actual grammars and outline parsing algorithms for each of these languages.

`Dyck-2` Given n types of brackets (that is, $2n$ symbols consisting of n opening and the matching n closing brackets), the `Dyck-n` consists of all the valid sequences of brackets. Algorithmically, a string belonging to `Dyck-n` can be parsed by maintaining a stack of opening brackets, and popping the topmost one when the corresponding closing bracket is encountered. If a closing bracket that does not match the topmost opening bracket is encountered, the string is rejected. The production rules that express the generation of strings from `Dyck-2`, then, are as follows.

$$
\begin{aligned}
S &\to S\,S \mid B_1 \mid B_2 \\
B_1 &\to \texttt{(}\,S\,\texttt{)} \mid \texttt{(}\,\texttt{)} \\
B_2 &\to \texttt{[}\,S\,\texttt{]} \mid \texttt{[}\,\texttt{]}
\end{aligned}
$$

`Expr` The Expr language is the set of prefix arithmetic expressions, in which operands are single digits from 0 to 9, and operators may be unary, binary or ternary. There are three operators of each type. Note that it is possible to view the vocabulary as being organized according to arity, or number of arguments needed. Thus digits are symbols of arity 0; unary operators of arity 1; binary operators of arity 2; and ternary operators of arity 3.

Given this arity-based classification of the vocabulary, it is easy to see that parsing consists of simply maintaining a counter that keeps track of how many more expressions are needed to complete the sequence – this counter starts at 1 (as the whole sequence, which is pending, represents one expression), and is incremented by $n - 1$ when we encounter a token of arity n.

Thus, for instance, if 3 more expressions are needed to complete the sequence and a binary operator is encountered, we now need 4 more expressions – two to complete the binary operator, and two more to satisfy the original three. Hence, the binary operator changes the counter value from 3 to 4, i.e., increments it by $2 - 1 = 1$. Parsing succeeds if this value reaches 0 at the end of the string, and fails if it becomes negative at any point during the parse.

$$
\begin{aligned}
S &\to O \mid D \\
D &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
O &\to U\,S \mid B\,S\,S \mid T\,S\,S\,S \\
U &\to \mathrm{un}_1 \mid \mathrm{un}_2 \mid \mathrm{un}_3 \\
B &\to \mathrm{bin}_1 \mid \mathrm{bin}_2 \mid \mathrm{bin}_3 \\
T &\to \mathrm{tern}_1 \mid \mathrm{tern}_2 \mid \mathrm{tern}_3
\end{aligned}
$$

English We define a simple fragment of English, intended to capture major parts of speech and bridge the gap between parsing languages like `Dyck-2` and `Expr` above, and natural language parsing. We retain the two most common sentence constructions, but ignore more complicated syntactic features like agreement and relative clauses, and morphological features, like conjugations and declensions. This grammar can be parsed using any standard CFG parsing algorithm, like Earley or CKY parsing [62].

$$
\begin{aligned}
S &\to \mathrm{NP\,VP} \\
\mathrm{NP} &\to \mathrm{Pro} \mid \mathrm{N} \mid \mathrm{NP\ Conj\ NP} \mid \mathrm{Adj\,N} \\
\mathrm{VP} &\to \mathrm{V} \mid \mathrm{V\,NP} \mid \mathrm{VP\ Conj\ VP} \mid \mathrm{VP\,Adv}
\end{aligned}
$$

4.1.1.2 Models

We train Transformers [30] via the standard autoregressive language modeling pipeline on each of the above languages. The models have 128-dimensional embeddings, with 4 attention heads and MLPs, and 2 blocks. In all cases, the models achieve more than 99% validity, i.e., under stochastic decoding, the strings generated belong to the language more than 99% of the time.

We use an online data generation process that randomly generates a sequence from a given PCFG at every iteration. The models are trained for 70,000 steps under this process.

4.1.2 SAE Architecture and Variants

Broadly, we use the conventional SAE architecture, which consists of two linear layers (an encoder and a decoder transform) with an activation function in between. More explicitly, if the sizes of the input and hidden state are d and h respectively, then our SAEs implement functions of the type

$$\mathrm{SAE}(x) = W_{\mathrm{dec}}(\sigma(W_{\mathrm{enc}}(x) + b_{\mathrm{enc}})) + b_{\mathrm{dec}},$$

where $W_{\mathrm{enc}} \in R^{h \times d}$, $W_{\mathrm{dec}} \in R^{d \times h}$, and $b_{\mathrm{enc}} \in R^h$, $b_{\mathrm{dec}} \in R^d$. We pick h from $\{d, 2d, 4d, 8d\}$ and $\sigma = \mathrm{ReLU}$ as the activation function. We refer to the encoder transform (up to and including the nonlinearity) as E, the decoder as D, and the hidden representation (the output of the encoder) as a

latent. When these latents have interpretable explanations, identified correlationally (see Section 4.2.1 below), we also refer to them as features.

Training is performed by optimizing on the MSE between x and SAE(x), where the input x to the SAEs is the output of the first block of the transformer model. The data is drawn from the same online generation process used to train the transformer model.

Sparsity is enforced via two main methods. The first (and most common in existing work) is an L_1-regularization term added to the loss, which encourages latent representations to have low L_1-norm. The hyperparameter for this method is the weight of the regularization term.
We also use, following [21], top-k regularization—at the SAE hidden layer, after the activation, we select the highest k latents, and zero out the rest. Effectively, we force the L_0-norm of the latents to be at most k, i.e., the hyperparameter is k itself. We examine the effects of hyperparameters on the performance of SAEs in Sec. 4.3.2 below.

We analyze variants of the architecture above by altering the normalization method and pre-bias. There are three parts of the operation of the SAE that can be normalized – the input, the reconstruction, and the decoder directions. This yields four architecture variants: no normalization (I), input and decoder with pre-bias (II), input alone (III), and input and reconstruction (IV). By pre-bias, we refer to the addition of a learnable vector $b_{\text{pre}} \in R^d$ to the input before applying the encoder, which is then subtracted after applying the decoder.

4.2 Experiments

Having established the models we study and the training paradigm for both these models and the SAEs, we now outline how we examine the SAEs and our methods of arriving at our conclusions. These consist of three types of results: interpretability of features, robustness of SAEs and causal validity.

4.2.1 Interpretability

Our interpretability pipeline is essentially qualitative. We analyze individual features by examining visualizations of their activations on certain tokens and samples, hypothesizing explanations, and then measuring correlations between our predicted activations and the ground-truth activations of the feature.

The manual inspection is based on an *a priori* notion of the latent generative factors inherent in each formal language, which we briefly outline here. In `Dyck-2`, we expect some computation to be based on the depth of the current token (i.e., the number of unclosed brackets preceding it). In `Expr`, we expect a counter that keeps track of the number of expressions yet to be generated. In `English`, we expect features corresponding to each part of speech in the grammar. These are all variables both necessary and sufficient for the data-generation process. We note, however, that these are not the only possibilities – each language may have several equivalent reformulations, but we believe these are the most intuitive

and suitable for the autoregressive decoding paradigm. The formulation of the grammars presented above is intended to clarify the central nature of these features to the data generation process.

This pipeline suffices to establish the correlational validity of the features. In general, we only examine SAEs which achieve near-perfect reconstruction (see Table 4.2 below), and we only consider features with Pearson correlation above about 0.8.

4.2.2 Robustness

This section of our experiments simply consists of recording the reconstruction losses of SAEs of each of the four types we describe above (Section 4.1.2). Additionally, we discuss qualitative observations on the frequency of interpretable features (as opposed to uninterpretable ones).

4.2.3 Causality

Given a trained SAE : $x \to D(E(x))$, we examine the causality of the latents by defining a reconstructed run of the transformer. Let the model's two layers be L_0 and L_1; then a normal run is logits $= W_{\text{LM}} \cdot L_1(L_0(t))$, where $t \in R^{s \times d}$ represents the embeddings of a sequence of s tokens, and $W_{\text{LM}} \in R^{d \times |V|}$ is the projection of the final representations into the logit space. The prediction of the next token is given, therefore, by softmax(logits). We then define a reconstructed run as logits $= W_{\text{LM}} \cdot L_1(D(E(L_0(t))))$; i.e., we interrupt the run after the first layer, pass the output through the SAE, and resume the run using the reconstruction of the SAE. For ease of notation, we partition the reconstructed run into two functions, which chained together give the complete run:

$$f := E \circ L_0$$

$$g := W_{\text{LM}} \circ L_1 \circ D.$$

Thus, the latent of a token x is given by $f(x)$ and the logit distribution of a token with latent l is given by $g(l)$. To study the causality of the representations, then, we intervene on the latent representation, between E and D (in other words, between f and g). We intervene on the specific elements of the latent that correlate with interpretable features of the input tokens; for instance, in the English grammars, we search for latents correlating with each part of speech in the grammar.

Our interventions consist of 'clamping' latents to some fixed value. In other words, for each token, we run the forward pass and extract the latent representation; we set certain elements to a clamp value, leaving the rest unchanged – this value is then passed through the decoder D and the rest of the forward pass.

In other words, consider a computational graph of the language model. The node corresponding to the layer that is being examined, i.e., the first hidden layer, is replaced by three nodes:

1. the actual representations generated by the previous layer, which form the input to the SAE;

2. the hidden layer of the SAE, which consists of the features identified by the SAE; and

3. the output, or the reconstruction, of the SAE.

The last node feeds back into the larger model, where the original input should have been used. Thus, we effectively replace the model activations with SAE reconstructions of those activations. We then intervene on node (ii) above – we select a single element in the SAE representation, set it to our value, and recompute the modified reconstruction, which is then used in the rest of the LM's forward pass. The exact intervention that we carry out is defined in terms of the maximum value $v_{\max}$ that the feature attains across a sample of 1280 sequences. We then select the values for the intervention by spacing 10 intervals across the range $[-v_{\max}, v_{\max}]$, i.e., the intervention values are

$$v_i = -v_{\max} + \frac{i}{10} \cdot 2v_{\max}, \forall i \in \{0, 1, \ldots, 10\}.$$

The principle of using $v_{\max}$ as the baseline for our interventions is in line with [69]. Note that the above leads to 11 possible values for the interventions.

4.3 Results

Now, we are ready to examine the results of the above experimental pipeline. As stated above, the results are classed into three main categories: the interpretability of the features (the nature of the features found, visualizations of their activation, and what conclusions can be drawn from their existence); the robustness of SAEs to hyperparameters (the different performances of different SAE architectures, both in terms of interpretability and reconstruction); and the causal validity of the features (whether the features we find have causal relevance). We discuss each set of results below.

4.3.1 Interpretability: What Features Are Found?

Qualitatively, we observe that top-k-regularized SAEs tend to have more interpretable features than L_1-regularized ones. The former results in interpretable features across all languages—we find features representing fundamental aspects of the corresponding grammars, as discussed next briefly.

In the case of `Dyck-2`, we expect the depth (the number of brackets yet to be closed) to be represented. We find a feature that thresholds the depth of tokens, i.e., it activates on tokens with depth above a certain threshold depth D. Usually, D is greater than the mean; for instance, when mean depth is 8.3, we find D = 11. We also find features that match corresponding pairs of opening and closing brackets, usually at lower depths (Figure 4.1). These features take values from only a few small ranges, and their values at an opening bracket determine those they take at the matching closing bracket. These features, even though they are binary, indicate that the token representations do maintain some form of counter (which aligns with claims from prior work, like [8]).

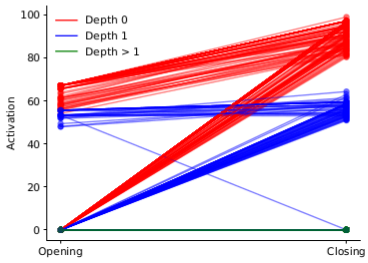

Figure 4.1: A feature matching corresponding opening and closing brackets. Each line represents a pair of brackets, and joins the opening bracket's activation (left) to the closing bracket's (right). We note that the depth and opening activation are sufficient to determine the closing activation, and that the opening and closing activations are sufficient to determine the depth.

In `Expr`, the analogue of depth is a counter that indicates how many more expressions are needed to complete the sequence, as described above in Section 4.1.1.1. We find a feature that activates exactly when this counter's value is 1, i.e., when exactly one expression is required (Figure 4.2). This provides strong evidence of a counter process being implemented.

An inference we can make from the `Expr` features is that there is an implicit "type" feature in the representations, distinguishing operators of different valences (arities). A natural place to look for this is the parts of speech in our English grammar, and we find that k-regularized SAEs do contain features corresponding to each part of speech. For example, we illustrate the 'adjective' feature in Figure 4.3.

Further, we note a scaling relation in the performance of the top-k-regularized SAEs with hidden dimension size. The reconstruction loss they achieve (after a fixed number of iterations) decreases according to a power law with the size of the autoencoder's hidden layer; similar results were seen by [60], relating the reconstruction loss to the L_1 penalty coefficient, indicating our setup captures part of the phenomenology observed with complex settings.

We list out other features we identified in various SAEs in Table 4.1 below. For each feature, we give the explanation, the language, the correlation between the explanation and the activation, and the hyperparameters for the SAE in which the feature was found. As all these SAEs are trained without `pre_bias` and without normalization, we only mention the other two hyperparameters: expansion factor (the ratio between the hidden size and the input size) and, according as the SAE is trained with L_1- or top-k- regularization, the L_1 coefficient or k.

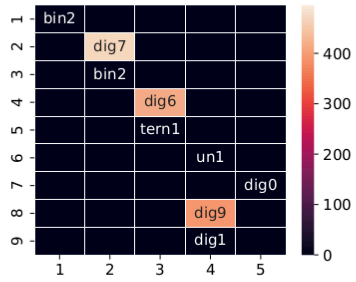

Figure 4.2: A feature that activates when exactly one more expression is required. Here, the x-axis is token depth, and the y-axis is token index. The lines connect the operators to their operands.

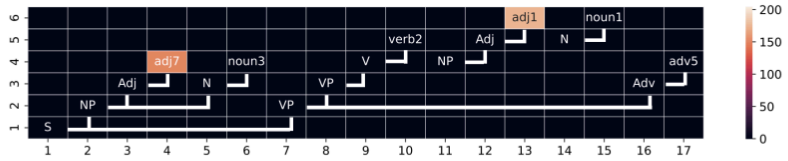

Figure 4.3: A feature that activates only on adjectives, at any position. Here, depth is represented by the y-axis and position by the x-axis; the lines connect nonterminals to their productions (see App. **??** for the production rules). The cell color represents the activation magnitude.

Explanation	Language	Settings	Correlation
One more expression required to complete the sequence.	Expr	$(2, k = 16)$	0.972
Last token.	Expr	$(8, \alpha = 10^{-3})$	0.984
Verbs.	English	$(8, k = 128)$	0.964
Adjectives.	English	$(8, k = 128)$	0.985
Adverbs.	English	$(8, k = 128)$	0.999
Conjunctions.	English	$(8, k = 128)$	0.932
Empty stack of type 0.	Dyck	$(8, k = 128)$	0.925
Empty stack of type 1.	Dyck	$(8, k = 128)$	0.814
Stack depth 11 or more.	Dyck	$(8, k = 128)$	0.924
All brackets at depth 0, and the first opening and all closing brackets at depth 1.	Dyck	$(8, k = 128)$	0.912

Table 4.1: Features identified by SAEs. We give a description of each feature, with the language it is found in, and the correlation (Pearson coefficient) between the activations and the explanation. All the SAEs are trained without `pre_bias` or normalization; we therefore identify them by their expansion factor and regularization factor (α in the case of L_1- and k in the case of top-k regularization).

Language	L_1				top-k			
	I	**II**	**III**	**IV**	**I**	**II**	**III**	**IV**
Dyck-2	0.01	0.0	0.01	0.0	49.27	3.48	50.02	0.06
Expr	33.31	6.50	0.06	0.49	99.88	69.14	99.76	0.0
English	0.29	1.13	0.01	0.01	92.46	50.12	80.79	0.37

Table 4.2: Sensitivity to Hyperparameters. Reconstruction Accuracy (%) averaged over regularization values and hidden size. We present the accuracies for SAEs with no normalization (I); with inputs and decoder normalized, and pre-bias (II); with inputs normalized (III); and with inputs and reconstructions normalized (IV). The reconstruction capabilities of the models shows high variance across hyperparameter settings.

4.3.2 Robustness: What Do SAEs Depend on?

As noted before, we find top-k-regularized SAEs consistently yield more interpretable features than L_1-regularized SAEs. We also measure the reconstruction accuracy, or the percentage of valid generations of the model after the latents are substituted with the SAE reconstructions. Table 4.2 shows the average results for each combination of regularization, pre-bias, and normalization strategy (averaged across regularization values and hidden sizes). We see that top-k-regularized SAEs usually (but not always) outperform L_1-regularized ones if all other settings are kept the same. We also note from this table the high sensitivity to hyperparameter settings that SAEs exhibit: no clear trend is visible across languages, regularization methods, or normalization settings – in line with [36]'s findings.

4.3.3 Causality: Do These Features Matter?

An interesting aspect of the features we identify is that despite strongly correlating with the discussed explanation (see Table 4.1), they do not have the expected causal effects. In this section, we outline our findings in causal experiments on the features described above.

In the case of `Expr`, we have seen above that there is a counter feature that activates when exactly one expression is left to complete the sequence. We pass an incomplete sequence to the model, and intervene on this feature by clamping it to one of $\{-v_{\max}, 0, v_{\max}\}$, where $v_{\max}$ is the maximum absolute value of the latent. We then examine the generations that result in each case. We expect that high clamp values will cause the model to generate only one more expression (even if more may be required) and low values will cause it to generate more than one, or perhaps end the sequence immediately (even if exactly one is needed); or vice versa, since we cannot be certain of the direction to apply intervention in. Table 4.3 presents the results of this experiment. For each input sample, we mention the number of expressions required to complete it, and the number of expressions actually generated by the model

under each intervention. We do not see the expected—or indeed any—causal effect of intervening on this feature – it appears that this feature is only correlational in nature.

Input Sequence	#Required	Clamp $-v_{\max}$	Clamp 0	Clamp $v_{\max}$
`un2 tern1 dig8 bin0 tern2 bin2 bin2 dig6 dig7 dig5`	4	4	4	4
`bin1 un1 tern1 bin2 dig0 dig7 dig0 bin1 dig3 dig7`	1	1	1	1
`un0 tern1 dig9 dig7 un0 tern1 un2 bin2 dig6 dig6`	2	2	2	2
`tern1 dig1 dig7 tern1 tern1 dig7 tern1 tern1 un0 un1`	8	8	8	8

Table 4.3: Behavior of the Expr model under interventions. The 'clamp' columns indicate the number of expressions generated by the model after being prompted by the input sequence and an intervention defined by the clamp value. We see that there is no effect of the intervention on the behavior of the model.

Similarly, we examine our `English` grammar. Here, we examine the effects of intervening on the part-of-speech features described above. Note that for a feature correlating with, say, adjectives, we can expect that if it has a causal effect, it must control the next-token distribution for that part-of-speech (as this is the only task the model is trained on). We thus apply our causality protocol to the feature correlating with adjectives. The results of this experiment are shown in Figure 4.4. In this case, as in `Expr`, we do not see a causal effect of intervening on this feature.

In conclusion, therefore, Inspired by studies in vision [36, 37], we propose a minimalistic setup to assess challenges in the use of SAEs for NLP interpretability. We validate our setup by identifying semantically meaningful features and demonstrating the sensitivity of said features' extraction to inductive biases. We further demonstrate a lack of causality, i.e., interventions on these features do not yield intended effects. We expect these results to bear out at scale as well, e.g., we find it likely that features identified by SAEs will not always be causally relevant to model computation. While previous works, like [40], have successfully built upon SAE features to identify circuits, we believe our results demonstrate the importance of embedding causality into the feature identification process as a first-class citizen.

As a step towards this integration, in the next chapter, we propose a modified objective for training SAEs that exploits the structure of our data, and find that it succeeds in identifying features with predictable causal function.

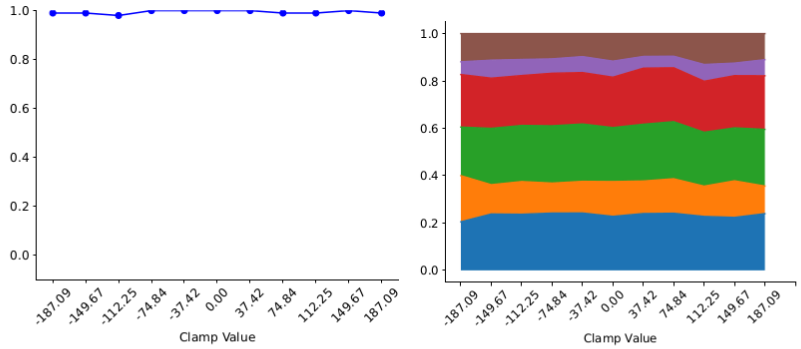

Figure 4.4: Behavior of the English model under interventions. We intervene on the model by replacing its hidden representations with the SAE's reconstructions, where an SAE latent (specifically, one corresponding to adjectives) is clamped to a fixed value. These values are selected at uniform intervals from $[-v_{\max}, v_{\max}]$, where $v_{\max}$ is the maximum value taken by that latent (in line with Templeton *et al.* [69]). For each value (x-axis), we plot the fraction of each part of speech (nouns, pronouns, adjectives, verbs, adverbs, and conjunctions) in the output (**left**) and the fraction of outputs that are grammatical (**right**). We see interventions yield essentially no visible effects.

Chapter 5

A Proposal for Causally Incentivized SAEs

In Chapter 4, we discussed the gaps in the SAE methodology that were identified through a reductionist approach centred on formal languages. Here, we continue the discussion on these gaps and present actionable future directions.

In particular, we focus on two takeaways of the results presented in Chapter 4:

1. that SAEs do not necessarily identify causal features; and

2. that a greater focus on the inductive biases of the SAE architecture is important.

Motivated by this, we propose and perform preliminary investigations for an approach that promotes learning of causally relevant features in our formal language setting. Specifically, we propose a modified SAE training protocol that incentivizes the identified features to be causal in nature, and examine its performance quantitatively and qualitatively in comparison with the SAEs we have already seen. Additionally, we address the second point above by drawing theoretical connections between the qualitative behaviour of this type of SAE and its architectural formulation.

In the remainder of this chapter, we first motivate the exact formulation of this modification through intuitions that partially come from prior work in vision and are partially assumed from the nature of the data. We then describe the modification mathematically.
Finally, we examine the results of the method, and compare it to the SAEs we have already seen in the previous chapter.

5.1 Motivation

Observations corresponding to ours on the correlational nature of SAE features have also been made in vision, where it has been noted that mere data reconstruction pipelines (e.g., based on autoencoders) can fail to disentangle the latent factors of a generative model [36]. However, later work on the topic demonstrated that correlations in the data-generating process (e.g., correlation in nearby frames of a video) can be leveraged to obtain disentangled representations [37, 31]. As this method relies on

supervision via other data samples, rather than a ground truth, it is described as weak supervision. A warranted question then is whether weak supervision can be elicited in language modeling scenarios too.

Motivated by the above, we take inspiration from [31]'s use of temporal correlations in video data as weak supervision. We argue the temporal and sequential nature of language can be exploited in a similar manner as well: more precisely, we can consider tokens belonging to the same sequence to share latent factors, and therefore pair up tokens within sequences in a similar way as image pairs in vision. We next operationalize this idea in our setup (Section 5.2) and then analyze the features obtained by SAEs trained with this method (Section 5.3.1). We discuss in Section 5.3.2 the connection between the inductive biases of the proposed pipeline and the nature of identified features.

We note that the proposed protocol and results should be deemed a proof-of-concept: our goal is to make a stronger connection to prior literature on autoencoder-based approaches to interpretability, hence eliciting (in)abilities of such protocols.

5.2 Definition

We propose an additional causal regularization term in the loss to incentivize causality. Thus, we now have a loss function given by

$$L = L_{\text{recon}} + \alpha L_{\text{sparse}} + \beta L_{\text{caus}}.$$

In order to define L_{caus}, note that what we want is for interventions on the latents to lead to interpretable changes in the model output. However, we do not want the changes to be arbitrary, i.e., we cannot simply try to maximize the change caused by an ablation – then the incentive for disentanglement is harmed. We therefore introduce weak supervision, motivated by the 'match pairing' approach of [37, 31].

Specifically, we try to make the run (forward pass) of the model on a given token t_1 similar to its run on a different, albeit related, token t_2. More precisely, given a single token t_1, we try to intervene on its latent representation l_1 to cause its logit distribution to resemble that of t_2. We operationalize this by a simple interpolation technique – we intervene on l_1 during the reconstructed run (as defined in Section 4.2.3) of the model on t_1, replacing it with $\lambda \cdot l_1 + (1 - \lambda) \cdot l_2$ for some $\lambda \in [0, 1)$. To define what to compare the output of the model on this corrupted run to be, we follow our intuition of using the run on t_2 as a form of supervision and compare the model output to the interpolation of the outputs of t_1 and t_2 (Figure 5.1) This is similar to prior work that aims to improve the robustness of representations and obtain smoother decision boundaries by training on interpolations of inputs [74]. Another way to motivate this operationalization is that we would like to force the model to change its output based on specific, surgical changes in the latents that happen as tokens evolve over time.

More formally, let $a \leftrightarrow b$ denote the interpolation $\lambda a + (1 - \lambda)b$ of vectors a and b by a scalar λ. Then we define the causal loss term for a token t_1, given a 'baseline' token t_2, as

$$L_{\text{caus}}(t_1, t_2) := d\left(g(l_1 \leftrightarrow l_2), g(l_1) \leftrightarrow g(l2)\right),$$

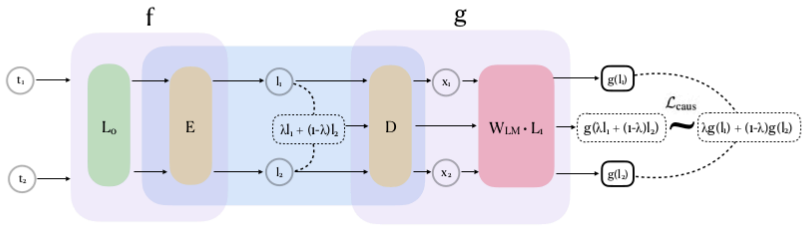

Figure 5.1: Operationalizing the causal regularization term. A clean run of the model consists of applying L_0, followed by L_1 and W_{LM}. We define a *reconstructed run*, which introduces an SAE between these two layers. The SAE consists of an encoder E and a decoder D. We denote the input embeddings as t, SAE latents as l, and reconstructed model activations as x. Furthermore, we group the first part of the reconstructed run as $f := E \circ L_0$, and the second part as $g := W_{\text{LM}} \circ L_1 \circ D$. Given two tokens t_1 and t_2, we interpolate between the latents l_1 and l_2 (indicated by a dotted line) and pass this as input to g; our causal loss $\mathcal{L}_{\text{caus}}$ is then given by the difference between the interpolation of the outputs, and the output of the interpolation.

where d denotes MSE. To pick t_2 in the pipeline above, we aim for as minimal an intervention as possible; this is to avoid the latent being completely overwritten, and to ensure that even small, surgical changes in the vicinity of the latent have causal effects. Thus we simply find the token with the latent nearest (by L_2 distance) to l_1. This is done at the sequence level; thus, for each sequence, we pair each token with the one nearest to it in the latent space, and find L_{caus} for each of these pairs. For λ, we simply sample from the uniform distribution over $[0, 1]$ every iteration – this incentivizes causal effects for a wide range of interventions.

5.3 Results

5.3.1 Nature of Features

In the case of English, as in the SAEs trained without this loss in the previous chapter, we identify a number of features correlating with parts of speech—specifically, adjectives, verbs and adverbs. Now, we predict that if these features have a monotonic causal function, they shift the output logit distribution towards the one predicted by the corresponding part of speech.

For example, a feature correlating with adjectives should promote the probability of nouns being predicted next, as nouns are the only PoS allowed to come after adjectives (see Section 4.1.1.1 for the exact

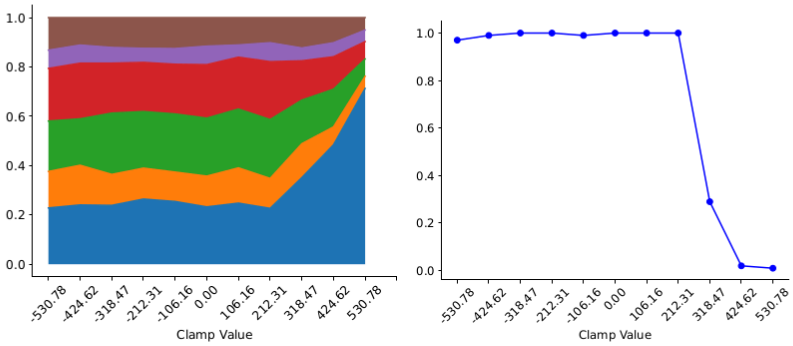

Figure 5.2: Behavior of the English model under interventions. We intervene on the model, replacing its hidden representations with the SAE's reconstructions, clamping a single latent (in this case, the one corresponding to adjectives) to fixed value. These fixed values are selected at uniform intervals from $[-v_{\max}, v_{\max}]$, where $v_{\max}$ is the maximum value taken by that latent (in line with Templeton *et al.* [69]). For each value of the clamp (x-axis), we plot the fraction of each part of speech (nouns, pronouns, adjectives, verbs, adverbs, and conjunctions) in the generated text (**left**) and the fraction of generations that are grammatical (**right**). We see that the SAEs trained with causal regularization have a predictable causal function.

grammar). In the case of verbs, anything may appear as the next token except another verb; thus we expect the probability of verbs to be downweighted by this feature. Figure 5.2 presents the effect of these interventions on the PoS distribution across sentences, along with the grammaticality of generations. The effect is as we hypothesized.

Similarly, we can make predictions for the other parts of speech for which features are found:

1. **Adjectives.** As noted in Section 5.3.1, nouns are the only tokens that can come after adjectives. Therefore we expect the feature correlating with adjectives, if it has a causal effect, to upweight the probability of nouns being predicted.

2. **Verbs.** We have also seen in Section 5.3.1 that any part of speech can follow a verb, except another verb. Thus, we expect the probability of verbs to be reduced by high values of verb-correlated features.

3. **Adverbs.** The grammar (Section 4.1.1.1) shows that adverbs can only be followed by conjunctions, more adverbs, or the <EOS> token. We then expect the former two types of tokens to be upweighted, and the other parts of speech dispreferred.

We note, importantly, that as soon as the distribution significantly shifts away from the uncorrupted distribution, the fraction of grammatical generations drops drastically. This supports, as pointed out in [72, 1], the difficulty of learning disentangled representations from correlated data (in this case, with respect to PoS distribution and grammaticality).
In fact, recent work [7, 77] observes that intervention-based methods (in comparison with probing-based methods) suffer from the drawback of damaging models' output quality in terms of coherence, creating a control-capabilities tradeoff. This indicates that the problem is possibly more general than simple semantic disentanglement, and this is a valuable avenue for future research.

We note, furthermore, that this is another validation of our setup (like the scaling law described in Section 4.3.1), as it demonstrates behaviours also observed in more complex, natural language settings.

5.3.2 Inductive Biases

As we remarked before, [36] demonstrated that finding latents underlying the data generating process is generally an ill-defined problem with multiple viable solutions. The authors thus argue that to the extent an approach achieves disentangled features, it is an inductive bias of the training pipeline. For example, their results show that a good deal of the variance in performance across paradigms (37%) can be accounted for by the effect of the objective function alone. In this section, therefore, we investigate the inductive biases of our approach.

We noted above that our proposed modification to the SAE pipeline led to only a subset of features that the vanilla SAEs identified, i.e., nouns, verbs, adverbs in the English grammar. In the spirit of the argument above, then, we attempt to sketch a theoretical connection between the actual form of our training objective and this behaviour. This is presented through two forms of evidence – qualitative and quantitative. We first examine the grammars of the three formal languages we work with and establish what is unique about these three features, and connect this property of the features to an architectural artefact of the transformer model we study. We then substantiate this qualitative analysis through interventional quantitative evidence, isolating the attribution process.

5.3.2.1 Intuitive Sketch

First, we reexamine the causal loss outlined above (Section 5.2). As $\leftrightarrow$ represents a convex combination for all $\lambda \in [0, 1]$, it effectively incentivizes the function g to be distributive over convex combinations (i.e., over addition and scalar multiplication). In other words, it is an incentive for g to be a linear function. This is of course not possible in full generality (otherwise we would not need nonlinear models), but can work for features that have an approximately linear mapping to the output space.

Before we examine this further, it is useful to lay out certain perspectives of the functioning of transformers. [18] presents a framework in which the transformers centre their computations around what they call a *residual stream*, i.e., the embedding space of the token sequence. This space is considered to be shared by all the layers, in continuation to the linear representation hypothesis, as each layer only implements *residual connections*:

$$h_n = h_{n-1} + L(h_n);$$

i.e., each layer only "adds information" to the residual stream, and semantically, the space stays the same. While the strong version of this perspective is certainly dependent on the LRH, the aspects of it important to us are independent of it – based on this intuition, the authors characterize the effect of such a residual computation as "writing to" the residual stream. In particular, each module that adds something to the stream in this way "writes to" some subspace of the stream (very rarely to all dimensions of it). Similarly, modules that take input from this stream are "reading from" it, or more precisely, from some subspace of it.

The form of transformers that we work with implements each layer with a residual attention module, followed by a residual MLP. Mathematically:

$$x \mapsto x + \text{attn}(x) + \text{MLP}(x + \text{attn}(x)).$$

With this perspective in mind, consider the computational graph that g represents [FIG]. Then, we consider the SAE decoder D as writing to a subspace of $x \in \mathbb{R}^d$, which each of $\text{attn}(x)$ and $\text{MLP}(x)$ then read from. Now, note that the attention module involves two softmax operations, and is therefore a highly nonlinear operation. The MLP, by contrast, is a simple linear-GeLU-linear operation, and thus involves only one nonlinearity (which is in fact linear in the range $\mathbb{R}^+$).

Thus, we claim that the more D writes to the MLP's input subspace, the 'more linear' it is; and the more it writes to the attention's input subspace the 'more nonlinear' it is. In other words, the 'linearity incentive' of the causal loss manifests itself as a preference for D to have more causal effect on the MLP than on the attention module.

To connect this to the parts of speech we have seen above, we note that it is exactly these parts of speech whose next-token distribution is static – adjectives, verbs, and adverbs. In other words, regardless of where they appear in the sequence, the next-token distribution is fixed by the part of speech, i.e., these tokens do not require attention for their next-token distribution. Consider the `English` grammar

$$\begin{aligned} S &\to \text{NP VP} \\ \text{NP} &\to \text{Pro} \mid \text{N} \mid \text{NP Conj NP} \mid \text{Adj N} \\ \text{VP} &\to \text{V} \mid \text{V NP} \mid \text{VP Conj VP} \mid \text{VP Adv} \end{aligned}$$

Note that adjectives, wherever they occur (in the "subject" or "object" NPs, or at any level of nesting within an NP with multiple nouns), must be followed by nouns. They cannot be followed by any other

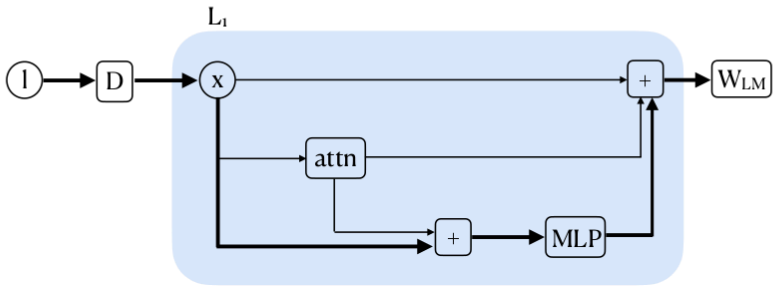

Figure 5.3: The computational graph of the second layer L_1 of the transformer model. Starting from the SAE latent l, the decoder D produces a reconstruction of the model activation x; the blue box then represents L_1, whose output is projected by W_{LM} into the logit space. The boldface arrows represent paths through the graph that involve fewer nonlinearities; our causal loss term incentivizes D to write to the subspace of x that is read by the modules in these paths.

part of speech. Thus the next-token distribution of any adjective token assigns all the probability mass to the nouns in the vocabulary.
Similarly, verbs only occur in one position (syntactically speaking) – within a verb phrase. As such, they can only ever be followed by conjunctions, adverbs (extending the VP), adjectives, nouns or pronouns (beginning the object NP); and the distribution among these five parts of speech is constant, defined by the probabilities in the grammar.
Finally, adverbs can only be followed by conjunctions (extending the VP) or other adverbs. As before, the distribution among these options is constant.
By contrast, the tokens that can follow a noun or pronoun depend on whether or not it is a subject (only conjunctions or verbs) or an object (only conjunctions, adverbs, or <EOS>). Similarly, the tokens that can follow a conjunction depend on whether it connects two noun phrases (only nouns, adjectives, or pronouns) or verb phrases (only verbs).

Putting our subspace intuitions together with the fact that nouns, verbs and adverbs do not require attention, we find a plausible answer – the objective creates a signal to promote learning of features that have causal effects on the 'more linear' modules (the MLP), while features that affect the attention module are dispreferred.

This also explains why our causal loss does not find features in the `Dyck-2` and `Expr` models; these languages rely on stacks or counters for generation (which are implemented by attention, as shown by [8], rather than vocabulary complexity, and so the attention module carries more of the computational

weight. We find further evidence for this in the fact that SAEs trained on these languages without causal loss do not identify any interpretable features other than those computed by attention.

5.3.2.2 Quantitative Evidence

Now, we present interventional, numerical evidence for the intuition sketched out above.
First, we re-examine the definition of the function L_1 (the second layer of the model). Ignoring LayerNorm and dropout, this function is simply

$$x \mapsto x + \text{attn}(x) + \text{MLP}(x + \text{attn}(x)).$$

Thus, when the input x is corrupted to, say $x+\Delta x$, each of these terms lends a corresponding corrupted term the output of L_1 (and therefore of g). Now, we examine the effect of adding each of these difference terms (separately) to $L_1(x)$. If, as we hypothesize, the MLP is the most significantly affected module, then we expect that

$$L_1(x) \approx L_1(x) + \Delta x \tag{5.1}$$

$$L_1(x) \approx L_1(x) + \Delta\,\text{attn}(x) \tag{5.2}$$

$$L_1(x + \Delta x) \approx L_1(x) + \Delta\,\text{MLP}(x + \text{attn}(x)); \tag{5.3}$$

In other words, the effect of the corruption should be replicated by corrupting only the MLP output; and corrupting either of the other two terms should have no effect.
Correspondingly, we can carry this analysis to the input of the MLP module. Here, we expect that the difference in the MLP module's output can be achieved by corrupting only x, and not $\text{attn}(x)$ at all. To make this precise, we define

$$\begin{aligned}\Delta_x\,&\text{MLP}(x + \text{attn}(x)) \\ &= \text{MLP}(x + \Delta x + \text{attn}(x)) \\ \Delta_{\text{attn}}\,&\text{MLP}(x + \text{attn}(x)) \\ &= \text{MLP}(x + \text{attn}(x + \Delta x)).\end{aligned}$$

Then we expect, in line with the previous case, that

$$L_1(x) \approx L_1(x) + \Delta_{\text{attn}}\,\text{MLP}(x + \text{attn}(x)) \tag{5.4}$$

$$L_1(x + \Delta x) \approx L_1(x) + \Delta_x\,\text{MLP}(x + \text{attn}(x)); \tag{5.5}$$

Applied Corruption	Divergence from Clean Run	Divergence from Corrupted Run
Δx	10^{-3}	1.37
$\Delta\,\text{attn}(x)$	$9.1 \cdot 10^{-5}$	1.38
$\Delta\,\text{MLP}(x + \text{attn}(x))$	1.29	$6.5 \cdot 10^{-3}$
$\Delta_x\,\text{MLP}(x + \text{attn}(x))$	1.44	0.04
$\Delta_{\text{attn}}\,\text{MLP}(x + \text{attn}(x))$	$4.8 \cdot 10^{-3}$	1.53

Table 5.1: KL divergence between partially corrupted runs, clean runs, and corrupted runs.

In other words, the corruption of the attention module does not affect the output of the MLP module. Note that in this setting, the corruption is only applied to the input of the MLP model; the outside $\text{attn}(x)$ term remains as is.

We provide in the Table 5.1 below the KL divergence between expressions (5.1), (5.2), (5.3), (5.4) and (5.5) above, and the distributions corresponding to $L_1(x)$ and $L_1(x + \Delta x)$. The results are as we expect from the qualitative explanation in the previous subsection. Here, the corruption consists of clamping a feature correlating with adjectives to its maximum value.

We have already noted that we were unable to identify interpretable features in the case of causal SAEs trained on `Dyck-2` and `Expr` models. We explain this, in line with the above, by observing that generation in these languages necessarily requires long-distance memory, in the form of a stack or a counter, which is a function taken up by attention in a transformer model [8].
Thus the MLP plays much less of a role in these languages (given the lack of diversity in the vocabulary), and attention is much more important. Since our causal loss avoids identifying features that attention relies on, this explains the lack of features in these two languages.

Overall, in line with prior work on autoencoders-based interpretability [36], our results highlight the importance of recognizing the inductive biases of any SAE paradigm for interpretability. We expect that any method aiming to integrate causality into feature identification will similarly have inductive biases, and no single method will overcome all limitations of the paradigm.

Chapter 6

Conclusions, Limitations and Future Work

In this chapter, we aim to wrap up the results, intuitions and methods that we have presented, and present them in the context of the prior work in this topic. We also highlight limitations of work that must be kept in mind when using these conclusions. Finally, we outline some things that we believe the field should focus on going forward.

6.1 Conclusion

In conclusion, we present our work with two main angles in mind: the potential of reductionism as a scientific approach to mechanistic interpretability, and a stress-test revealing the most pressing issues with the sparse autoencoder methodology. We summarize our perspectives on these two aspects of our work below.

6.1.1 Reductionism

We present our work as a validation of the idea that reductionism and simplification of the setting can lead to useful and valuable insights into problems in mechanistic interpretability.

Concretely, we focus on the valuation of the SAE pipeline. The development of SAEs as the latest big step in interpretability has shown great promise as an unsupervised paradigm for feature decomposition, and its potential for being automated through the use of large language models has been explored to a great extent. The features identified by SAEs have formed the basis for identifiable circuits and problem-solving mechanisms, and at times new insights into the ways models work. The causal nature of the features, which has recently become a central concern, has also been demonstrated in several works.

However, in-depth theoretical and empirical investigations into the ways in which SAEs identify features have revealed a fundamental reliance on inductive biases, and a lack of robustness, that have not been adequately addressed in works that adopt the approach. These studies have been largely confined to the image domain, however.

Some of these studies leverage synthetic data, for which we have complete information of the data-generating process and the latent factors, which make it possible to evaluate the performance of the SAEs with more certainty than in the case of real-world data. Therefore, we build on this idea and create synthetic text data – formal grammars – to examine SAEs in this domain.
Our results line up with much previous work, and point to the additional evidence that SAE features are not necessarily causal; as causality was not a focus in image-centric interpretability studies, this is a valuable new insight that needs to be kept in mind for future work involving and building on SAEs.

6.1.2 Inductive Biases

Another angle of our work that we wish to present is that our results show that SAEs are deeply sensitive to inductive biases, which has been suggested in prior work as well. We make the case that this tendency of the model is one that needs to be understood better for us to make the most use of the pipeline as a whole. The lack of causality in our preliminary results acts as evidence towards this claim as well – if we don't understand the architecture and the loss design well enough, we can't be confident that the features obtained behave as we would like them to.

We present, therefore, a first step in this direction: a modification of the objective to incentivize causality. We implement and test this modified methodology, in addition to an extensive theoretical examination of it. In the spirit of our exhortation to have a better understanding of the inductive biases of the pipeline, we hypothesize a connection between the objective function we implement and the nature of the features learnt by our model. While they are more likely to have causal function, as intended, they are more restricted due to this inductive bias.
We also draw attention to the fact that the specific properties of the formal languages involved (in particular, the static/dynamic nature of their next-token distributions) formed the key to identifying this inductive bias. This, we believe, additionally buttresses our proposal for reductionism for mechanistic interpretability.

6.2 Limitations

Stemming from the basis of our work in the tradition of synthetic data as an instrument to explore machine learning models are many important limitations of it. Particularly, the limited vocabulary size and the relative simplicity of the data mean that the distribution is by no means comparable to real-world data, especially in natural language, where no comprehensive enumeration of the factors of generation can be made.

Therefore, it is important to keep in mind that any behaviours exhibited by models in this setting cannot be *a priori* generalized to natural language – it must first be verified that the behaviours do correspond to the behaviour of the model in more complicated, realistic settings.
An interesting case where this does not happen is [39], in which the authors identify an important

difference between SAEs working on a low-dimensional subset of the data (in this case, data for the indirect object identification task) and SAEs trained on the full data. SAEs of the latter type are equally interpretable to those trained with decoder directions frozen throughout training, which indicates that it is the reduced complexity of the distribution that actually enables meaningful, interpretable directions. Thus, findings on the feasibility of using SAEs do not in this case carry over.

In this vein, our results give us reason to doubt *a priori* the causality of SAE features, but not conclusive evidence of the lack thereof. We therefore emphasize the need to verify causality before making general claims on the functionality of the paradigm.

6.3 Future Work

We make the case that the most pressing direction of future work in the SAE paradigm is towards a better understanding of the inductive biases of the model. We have mentioned previous work that made similar claims to ours – e.g. [36], who show that 37% of the variance across SAEs can be accounted for by the objective function alone – and we believe our results reinforce the need to pursue this direction in future work.

Concretely, a better understanding of inductive biases must satisfy several criteria. It must be comprehensive, not only in terms of architecture and objective function, but also across metrics like reconstruction, interpretability, disentanglement, and causality.
It must also be *controllable* – the ideal theory would also enable us to develop particular architectures and types of SAEs depending on our requirements, the granularity of features obtained, and so on. Steps have already been taken towards this in works like [60], who relate the dictionary size to the features obtained, the regularization coefficient in the objective, etc.

Another direction of future work that our work validates is the use of formal languages as a synthetic testbed in the natural language setting. This can be used not just, as we have shown, as an evaluative setup for interpretability paradigms, but also as an approach to interpretability in its own right, creating avenues for surgical control of the data and exploring the effects on the model.

Research Papers Based on the Thesis Work

Workshop Papers

1. **Abhinav S Menon**, Manish Shrivastava, David S Krueger, Ekdeep Lubana. Analyzing (In)Abilities of SAEs via Formal Languages. *MINT@NeurIPS 2025* https://openreview.net/forum?id=FDn8YNn6U6

Conference Papers

1. **Abhinav S Menon**, Manish Shrivastava, David S Krueger, Ekdeep Lubana. Analyzing (In)Abilities of SAEs via Formal Languages. *Proceedings of the 2025 Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics* https://openreview.net/attachment?id=eCo6y0c84f&name=anonymous_PDF

Bibliography

[1] Ahmed, Ossama, Träuble, Frederik, Goyal, Anirudh, Neitz, Alexander, Bengio, Yoshua, Schölkopf, Bernhard, Wüthrich, Manuel, & Bauer, Stefan. 2020. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*.

[2] Alain, Guillaume, & Bengio, Yoshua. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

[3] Allen-Zhu, Zeyuan, & Li, Yuanzhi. 2024. *Physics of Language Models: Part 1, Learning Hierarchical Language Structures*.

[4] Arps, David, Samih, Younes, Kallmeyer, Laura, & Sajjad, Hassan. 2022. Probing for constituency structure in neural language models. *arXiv preprint arXiv:2204.06201*.

[5] Belinkov, Yonatan. 2022. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics*, **48**(1), 207–219.

[6] Bereska, Leonard, & Gavves, Efstratios. 2024. Mechanistic Interpretability for AI Safety–A Review. *arXiv preprint arXiv:2404.14082*.

[7] Bhalla, Usha, Srinivas, Suraj, Ghandeharioun, Asma, & Lakkaraju, Himabindu. 2024. Towards Unifying Interpretability and Control: Evaluation via Intervention. *arXiv preprint arXiv:2411.04430*.

[8] Bhattamishra, Satwik, Ahuja, Kabir, & Goyal, Navin. 2020. On the ability and limitations of transformers to recognize formal languages. *arXiv preprint arXiv:2009.11264*.

[9] Bills, Steven, Cammarata, Nick, Mossing, Dan, Tillman, Henk, Gao, Leo, Goh, Gabriel, Sutskever, Ilya, Leike, Jan, Wu, Jeff, & Saunders, William. 2023. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, **2**.

[10] Blodgett, Su Lin, Barocas, Solon, Daumé III, Hal, & Wallach, Hanna. 2020. Language (technology) is power: A critical survey of" bias" in nlp. *arXiv preprint arXiv:2005.14050*.

[11] Bricken, Trenton, Templeton, Adly, Batson, Joshua, Chen, Brian, Jermyn, Adam, Conerly, Tom, Turner, Nicholas L., Anil, Cem, Denison, Carson, Askell, Amanda, Lasenby, Robert, Wu, Yifan,

Kravec, Shauna, Schiefer, Nicholas, Maxwell, Tim, Joseph, Nicholas, Tamkin, Alex, Nguyen, Karina, McLean, Brayden, Burke, Josiah E., Hume, Tristan, Carter, Shan, Henighan, Tom, & Olah, Chris. 2023. *Towards Monosemanticity: Decomposing Language Models With Dictionary Learning*. https://transformer-circuits.pub/2023/monosemantic-features.

[12] Burns, Collin, Ye, Haotian, Klein, Dan, & Steinhardt, Jacob. 2022. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*.

[13] Chaudhary, Maheep, & Geiger, Atticus. 2024. *Evaluating Open-Source Sparse Autoencoders on Disentangling Factual Knowledge in GPT-2 Small*.

[14] Cunningham, Hoagy, Ewart, Aidan, Riggs, Logan, Huben, Robert, & Sharkey, Lee. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

[15] Demircan, Can, Saanum, Tankred, Jagadish, Akshay K., Binz, Marcel, & Schulz, Eric. 2024. *Sparse Autoencoders Reveal Temporal Difference Learning in Large Language Models*.

[16] Dev, Sunipa, Li, Tao, Phillips, Jeff M, & Srikumar, Vivek. 2020. On measuring and mitigating biased inferences of word embeddings. *Pages 7659–7666 of: Proceedings of the AAAI conference on artificial intelligence*, vol. 34.

[17] Elazar, Yanai, Ravfogel, Shauli, Jacovi, Alon, & Goldberg, Yoav. 2021. Amnesic Probing: Behavioral Explanation with Amnesic Counterfactuals. *Transactions of the Association for Computational Linguistics*, **9**, 160–175.

[18] Elhage, Nelson, Hume, Tristan, Olsson, Catherine, Schiefer, Nicholas, Henighan, Tom, Kravec, Shauna, Hatfield-Dodds, Zac, Lasenby, Robert, Drain, Dawn, Chen, Carol, *et al.* 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

[19] Ethayarajh, Kawin, Duvenaud, David, & Hirst, Graeme. 2018. Towards understanding linear word analogies. *arXiv preprint arXiv:1810.04882*.

[20] Foote, Alex, Nanda, Neel, Kran, Esben, Konstas, Ioannis, Cohen, Shay, & Barez, Fazl. 2023. Neuron to graph: Interpreting language model neurons at scale. *arXiv preprint arXiv:2305.19911*.

[21] Gao, Leo, la Tour, Tom Dupré, Tillman, Henk, Goh, Gabriel, Troll, Rajan, Radford, Alec, Sutskever, Ilya, Leike, Jan, & Wu, Jeffrey. 2024. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

[22] Geva, Mor, Schuster, Roei, Berant, Jonathan, & Levy, Omer. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

[23] Gurnee, Wes, Nanda, Neel, Pauly, Matthew, Harvey, Katherine, Troitskii, Dmitrii, & Bertsimas, Dimitris. 2023. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*.

[24] Handa, Kunal, Tamkin, Alex, McCain, Miles, Huang, Saffron, Durmus, Esin, Heck, Sarah, Mueller, Jared, Hong, Jerry, Ritchie, Stuart, Belonax, Tim, *et al*. 2025. Which Economic Tasks are Performed with AI? Evidence from Millions of Claude Conversations. *arXiv preprint arXiv:2503.04761*.

[25] Higgins, Irina, Matthey, Loic, Pal, Arka, Burgess, Christopher, Glorot, Xavier, Botvinick, Matthew, Mohamed, Shakir, & Lerchner, Alexander. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*.

[26] Huang, Jing, Geiger, Atticus, D'Oosterlinck, Karel, Wu, Zhengxuan, & Potts, Christopher. 2023. Rigorously assessing natural language explanations of neurons. *arXiv preprint arXiv:2309.10312*.

[27] Hyvarinen, Aapo, & Morioka, Hiroshi. 2017. Nonlinear ICA of temporally dependent stationary sources. *Pages 460–469 of: Artificial Intelligence and Statistics*. PMLR.

[28] Hyvärinen, Aapo, & Pajunen, Petteri. 1999. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, **12**(3), 429–439.

[29] Hyvarinen, Aapo, Sasaki, Hiroaki, & Turner, Richard. 2019. Nonlinear ICA using auxiliary variables and generalized contrastive learning. *Pages 859–868 of: The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR.

[30] Karpathy, Andrej. 2022. *nanoGPT*. https://github.com/karpathy/nanoGPT.

[31] Klindt, David, Schott, Lukas, Sharma, Yash, Ustyuzhaninov, Ivan, Brendel, Wieland, Bethge, Matthias, & Paiton, Dylan. 2020. Towards nonlinear disentanglement in natural data with temporal sparse coding. *arXiv preprint arXiv:2007.10930*.

[32] Lachapelle, Sébastien, Deleu, Tristan, Mahajan, Divyat, Mitliagkas, Ioannis, Bengio, Yoshua, Lacoste-Julien, Simon, & Bertrand, Quentin. 2023. Synergies between disentanglement and sparsity: Generalization and identifiability in multi-task learning. *Pages 18171–18206 of: International Conference on Machine Learning*. PMLR.

[33] Lan, Michael, Torr, Philip, Meek, Austin, Khakzar, Ashkan, Krueger, David, & Barez, Fazl. 2024. *Sparse Autoencoders Reveal Universal Feature Spaces Across Large Language Models*.

[34] Li, Kenneth, Burns, Alexander, Morcos, Ari, LeCun, Yann, & Perez, Ethan. 2023a. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task. *arXiv preprint arXiv:2305.10408*.

[35] Li, Kenneth, Patel, Oam, Viégas, Fernanda, Pfister, Hanspeter, & Wattenberg, Martin. 2023b. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, **36**, 41451–41530.

[36] Locatello, Francesco, Bauer, Stefan, Lucic, Mario, Raetsch, Gunnar, Gelly, Sylvain, Schölkopf, Bernhard, & Bachem, Olivier. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. *Pages 4114–4124 of: international conference on machine learning*. PMLR.

[37] Locatello, Francesco, Poole, Ben, Rätsch, Gunnar, Schölkopf, Bernhard, Bachem, Olivier, & Tschannen, Michael. 2020. Weakly-supervised disentanglement without compromises. *Pages 6348–6359 of: International Conference on Machine Learning*. PMLR.

[38] Lubana, Ekdeep Singh, Kawaguchi, Kyogo, Dick, Robert P, & Tanaka, Hidenori. 2024. A percolation model of emergence: Analyzing transformers trained on a formal language. *arXiv preprint arXiv:2408.12578*.

[39] Makelov, Aleksandar, Lange, George, & Nanda, Neel. 2024. Towards principled evaluations of sparse autoencoders for interpretability and control. *arXiv preprint arXiv:2405.08366*.

[40] Marks, Samuel, Rager, Can, Michaud, Eric J, Belinkov, Yonatan, Bau, David, & Mueller, Aaron. 2024. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*.

[41] Meng, Kevin, Bau, David, Andonian, Alex, & Belinkov, Yonatan. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, **35**, 17359–17372.

[42] Meng, Kevin, Solar-Lezama, Armando, Belinkov, Yonatan, & Bau, David. 2023. Mass-Editing Memory in a Transformer. *arXiv preprint arXiv:2304.14767*.

[43] Mészáros, Anna, Ujváry, Szilvia, Brendel, Wieland, Reizinger, Patrik, & Huszár, Ferenc. 2024. Rule extrapolation in language modeling: A study of compositional generalization on OOD prompts. *Advances in Neural Information Processing Systems*, **37**, 34870–34899.

[44] Michaud, Eric, Liu, Ziming, Girit, Uzay, & Tegmark, Max. 2023. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, **36**, 28699–28722.

[45] Mikolov, Tomas, Chen, Kai, Corrado, Greg, & Dean, Jeffrey. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[46] Mu, Jesse, & Andreas, Jacob. 2020. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, **33**, 17153–17163.

[47] Nanda, Neel, Lee, Andrew, & Wattenberg, Martin. 2023a. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*.

[48] Nanda, Neel, Chan, Lawrence, Lieberum, Tom, Smith, Jess, & Steinhardt, Jacob. 2023b. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

[49] Nguyen, Thanh Tam, Huynh, Thanh Trung, Ren, Zhao, Nguyen, Phi Le, Liew, Alan Wee-Chung, Yin, Hongzhi, & Nguyen, Quoc Viet Hung. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

[50] Olah, Chris, Cammarata, Nick, Schubert, Ludwig, Goh, Gabriel, Petrov, Michael, & Carter, Shan. 2020. Zoom in: An introduction to circuits. *Distill*, **5**(3), e00024–001.

[51] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

[52] Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. 2014. GloVe: Global Vectors for Word Representation. *Pages 1532–1543 of: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

[53] Peters, Matthew E., Neumann, Mark, Iyyer, Mohit, Gardner, Matt, Clark, Christopher, Lee, Kenton, & Zettlemoyer, Luke. 2018. *Deep contextualized word representations*.

[54] Pimentel, Tiago, Valvoda, Josef, Maudslay, Rowan Hall, Zmigrod, Ran, Williams, Adina, & Cotterell, Ryan. 2020. Information-theoretic probing for linguistic structure. *arXiv preprint arXiv:2004.03061*.

[55] Qu, Youyang, Yuan, Xin, Ding, Ming, Ni, Wei, Rakotoarivelo, Thierry, & Smith, David. 2023. Learn to unlearn: A survey on machine unlearning. *arXiv preprint arXiv:2305.07512*.

[56] Ravfogel, Shauli, Elazar, Yanai, Gonen, Hila, Twiton, Michael, & Goldberg, Yoav. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. *arXiv preprint arXiv:2004.07667*.

[57] Rogers, Anna, Kovaleva, Olga, & Rumshisky, Anna. 2021. A primer in BERTology: What we know about how BERT works. *Transactions of the association for computational linguistics*, **8**, 842–866.

[58] Şahin, Gözde Gül, Vania, Clara, Kuznetsov, Ilia, & Gurevych, Iryna. 2020. LINSPECTOR: Multilingual probing tasks for word representations. *Computational Linguistics*, **46**(2), 335–385.

[59] Sepliarskaia, Anna, Kiseleva, Julia, & de Rijke, Maarten. 2019. How to not measure disentanglement. *arXiv preprint arXiv:1910.05587*.

[60] Sharkey, Lee, Braun, Dan, & Beren. 2024. *[Interim research report] Taking features out of superposition with sparse autoencoders*. https://www.lesswrong.com/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition.

[61] Shu, Rui, Chen, Yining, Kumar, Abhishek, Ermon, Stefano, & Poole, Ben. 2019. Weakly supervised disentanglement with guarantees. *arXiv preprint arXiv:1910.09772*.

[62] Sipser, Michael. 1996. Introduction to the Theory of Computation. *ACM Sigact News*, **27**(1), 27–29.

[63] Spies, Alex F, Edwards, William, Ivanitskiy, Michael L, Skapars, Adrians, Räuker, Tilman, Inoue, Katsumi, Russo, Alessandra, & Shanahan, Murray. 2024. Transformers Use Causal World Models in Maze-Solving Tasks. *arXiv preprint arXiv:2412.11867*.

[64] Squires, Chandler, Seigal, Anna, Bhate, Salil S, & Uhler, Caroline. 2023. Linear causal disentanglement via interventions. *Pages 32540–32560 of: International Conference on Machine Learning*. PMLR.

[65] Strobl, Lena, Merrill, William, Weiss, Gail, Chiang, David, & Angluin, Dana. 2023. Transformers as recognizers of formal languages: A survey on expressivity. *arXiv preprint arXiv:2311.00208*.

[66] Sundararajan, Mukund, Taly, Ankur, & Yan, Qiqi. 2017. Axiomatic Attribution for Deep Networks. *Pages 3319–3328 of: International Conference on Machine Learning*.

[67] Sutton, Richard S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, **3**, 9–44.

[68] Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, & Rabinovich, Andrew. 2015. Going deeper with convolutions. *Pages 1–9 of: Proceedings of the IEEE conference on computer vision and pattern recognition*.

[69] Templeton, Adly, Conerly, Tom, Marcus, Jonathan, Lindsey, Jack, Bricken, Trenton, Chen, Brian, Pearce, Adam, Citro, Craig, Ameisen, Emmanuel, Jones, Andy, Cunningham, Hoagy, Turner, Nicholas L, McDougall, Callum, MacDiarmid, Monte, Freeman, C. Daniel, Sumers, Theodore R., Rees, Edward, Batson, Joshua, Jermyn, Adam, Carter, Shan, Olah, Chris, & Henighan, Tom. 2024. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread*.

[70] Tenney, Ian, Xia, Patrick, Chen, Berlin, Wang, Alex, Poliak, Adam, McCoy, R Thomas, Kim, Najoung, Van Durme, Benjamin, Bowman, Samuel R, Das, Dipanjan, *et al.* 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

[71] Touvron, Hugo, Lavril, Thibaut, Izacard, Gautier, Martinet, Xavier, Lachaux, Marie-Anne, Lacroix, Timothée, Rozière, Baptiste, Goyal, Naman, Hambro, Eric, Azhar, Faisal, *et al.* 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[72] Träuble, Frederik, Creager, Elliot, Kilbertus, Niki, Locatello, Francesco, Dittadi, Andrea, Goyal, Anirudh, Schölkopf, Bernhard, & Bauer, Stefan. 2021. On disentangled representations learned from correlated data. *Pages 10401–10412 of: International conference on machine learning*. PMLR.

[73] Turner, Alexander Matt, Thiergart, Lisa, Leech, Gavin, Udell, David, Vazquez, Juan J, Mini, Ulisse, & MacDiarmid, Monte. 2023. Activation addition: Steering language models without optimization. *arXiv e-prints*, arXiv–2308.

[74] Verma, Vikas, Lamb, Alex, Beckham, Christopher, Najafi, Amir, Mitliagkas, Ioannis, Lopez-Paz, David, & Bengio, Yoshua. 2019. Manifold mixup: Better representations by interpolating hidden states. *Pages 6438–6447 of: International conference on machine learning*. PMLR.

[75] Wang, Kevin, Variengien, Alexandre, Conmy, Arthur, Shlegeris, Buck, & Steinhardt, Jacob. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

[76] Weiss, Gail, Goldberg, Yoav, & Yahav, Eran. 2018. Extracting automata from recurrent neural networks using queries and counterexamples. *Pages 5247–5256 of: International Conference on Machine Learning*. PMLR.

[77] Wu, Zhengxuan, Arora, Aryaman, Geiger, Atticus, Wang, Zheng, Huang, Jing, Jurafsky, Dan, Manning, Christopher D, & Potts, Christopher. 2025. AxBench: Steering LLMs? Even Simple Baselines Outperform Sparse Autoencoders. *arXiv preprint arXiv:2501.17148*.

[78] Xu, Jie, Wu, Zihan, Wang, Cong, & Jia, Xiaohua. 2024. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

[79] Yellin, Daniel M, & Weiss, Gail. 2021. Synthesizing context-free grammars from recurrent neural networks (extended version). *arXiv preprint arXiv:2101.08200*.

[80] Yu, Zeping, & Ananiadou, Sophia. 2023. Neuron-level knowledge attribution in large language models. *arXiv preprint arXiv:2312.12141*.

Formal Languages for Mechanistic Interpretability

ORIGINALITY REPORT

11%	10%	6%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

8%

★ arxiv.org

Internet Source

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		