Language-Independent Concept Spaces
Found in Large Language Models

A Thesis
Presented to
The Division of Mathematical
&
Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

A. Mokalla

May 2025

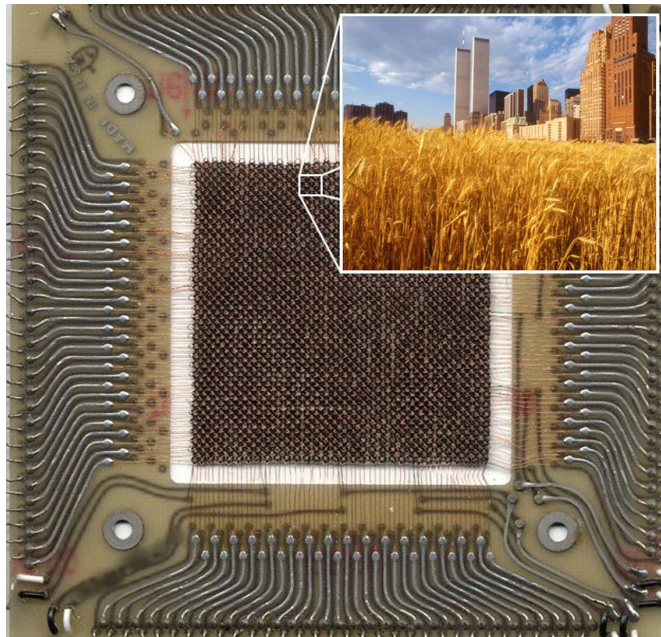Approved for the Division
(Computer Science)

_____
Greg Anderson

Acknowledgements

I want to thank Paul Bunyon, Greg Anderson, Rube Goldberg, Sameer ud Dowla Khan, my mom, and my dad. The list goes on, but if I thanked a single person more I probably wouldn't be able to stop.

Preface

τοῦτο μόνον δύναται ἡ ἐμὴ τέχνη.

[1–3]

Contents

Abstract

I investigate the existence of language-independent *concept spaces* within large language models (LLMs), wherein a model's internal representations collapse across linguistic boundaries to approximate shared semantic manifolds. By developing Universal Sparse Autoencoders, an extension of existing interpretability methods to the domain of language modeling, I demonstrate evidence that language-agnostic priors on interpretation techniques offer descriptive benefits. The sudden convergence of feature representations across translations of the same texts in different languages suggests an underlying structure of the model's reasoning. These concept spaces organize themselves according to meaning rather than to form, clustering semantically related concepts together independently of their surface realizations. This thesis extends theoretical insights into how artificial intelligences represent knowledge and offers practical interpretation methods for addressing biases embedded in multilingual models.

Dedication

This thesis is dedicated to those who create joy.

Introduction

ARGE language models (LLMs) are not programmed directly by humans. If they were, their behavior should be explainable in terms of a program that a human could write. Rather, they're programmed by, or "trained on," large amounts of data that are fed into them by humans. During this process of "learning to predict the next word," LLMs learn to solve the problems that allow them to make these predictions in the first place. These problem-solving strategies are embedded, somehow, into the billions of computations an LLM performs for each word it writes. However, these strategies have proven inscrutable to derive explicitly from examining these computations directly. This means that we don't actually have a satisfying explanation for the vast majority of the capabilities that LLMs blatantly possess.

We do know that LLMs are neural networks trained to predict the next token in a sequence of text.[1] Although this is a simple task to describe, the ability to accurately predict the next token in a text requires a complex representation of language structure, semantics, and the worlds they describe to necessarily exist within the model in some form or another. Speaking abstractly, a system ought never to be *less* complex than the output it produces.

Modern LLMs typically use a *transformer* architecture, which processes text through multiple layers of *self-attention* mechanisms [4]. In general terms, each layer of a transformer-based LLM updates a token's representation by using mappings that are directly determined by the tokens that preceded it. These representations are vectors in a high-dimensional *embedding space*. Each change to an embedding, or *activation*, lives in an *activation space* of the same size. Each layer in a transformer processes tokens through parallel *attention heads*, which compute

---

[1]A token is a bit like a morpheme, in that it's the smallest unit of meaning that the model works over, but is not necessarily an entire word.

weighted combinations of the representations from the previous layer to generate new activations. It then adds its new activation to the last layer's representation, and passes this sum to be processed further by the next layer (see Figure 1.1). The unique strength of this architecture is that it allows the model to selectively *attend* to different parts of the input sequence simultaneously, allowing it to regress over complex patterns and interdependencies in the data. As a result, the model learns a hierarchy of specialized charts of mappings that define a token's processing through each of these layers.

Particularly relevant to this thesis is how transformers handle multilingual tasks. Even before the current generation of LLMs, transformer-based models, like Google Translate, demonstrated an ability to learn language independent representations nearly a decade ago, and were found to be more effective and accurate translators than the state of the art machine translation methods of the time [5]. When trained on parallel texts in multiple languages, these models develop internal representations that capture semantic meaning in a way that is evidently independent of specific languages. This suggests that somewhere in the model's activation space, there exists a manifold of semantic concepts that is largely language-agnostic.

Before diving in, let's consider an example: the concept of a "tree" exists, and is fairly similar, across many languages, although the surface forms of this concept to which LLMs are actually exposed (e.g. the tokens for **"árbol"** in Spanish, "дерево" in Russian, "木" in Japanese) are highly variable. Despite the vastly different surface forms of these tokens, a multilingual model must learn to map them to similar regions in its activation space to perform translation tasks effectively. This implies that the model would benefit from learning to represent the underlying semantic concept independently of its linguistic realization, at least up to a certain level of precision.

However, the exact nature of these representations, and the question of their existence in the first place, are unclear. This is in large part due to a lack of interpretability methods that address the variance inherent to multilingual LLMs. Although at first glance learned token embeddings behave similarly to what we would expect of a truly language-agnostic or "semantic" embedding, these relationships rapidly degrade as tokens are processed into deeper layers of an LLM. Even in earlier layers, we fail to account for the complex effects that small changes can make to a model's eventual output. Further, interpreting token embeddings in
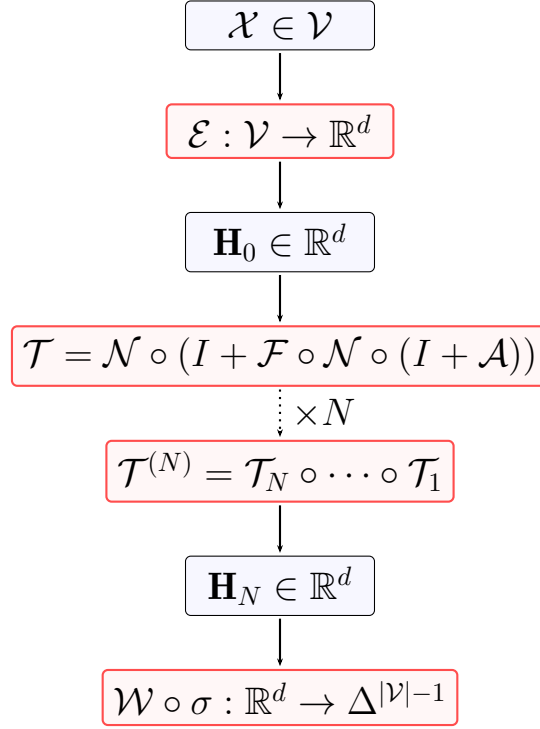
$$\mathcal{X} \in \mathcal{V}$$

$$\mathcal{E} : \mathcal{V} \to \mathbb{R}^d$$

$$\mathbf{H}_0 \in \mathbb{R}^d$$

$$\mathcal{T} = \mathcal{N} \circ (I + \mathcal{F} \circ \mathcal{N} \circ (I + \mathcal{A}))$$

$$\times N$$

$$\mathcal{T}^{(N)} = \mathcal{T}_N \circ \cdots \circ \mathcal{T}_1$$

$$\mathbf{H}_N \in \mathbb{R}^d$$

$$\mathcal{W} \circ \sigma : \mathbb{R}^d \to \Delta^{|\mathcal{V}|-1}$$

Figure 1.1: Schematic of the large $\mathcal{T}$ransformer-based language model architecture

isolation cannot tell us anything of the compositional processes that an LLM performs when using multiple embeddings to shape how future tokens are attended to. Before investigating further, we should review some basics of the object of our study.

1.1 Neural Networks

Neural networks consist of multiple "layers" of multiple "neurons." Often, the behavior of a neuron in one layer can influence most of the neurons in the following layers. Each neuron in a layer stores a number each time the network is used. These values are determined by the input to the network in its very first layer. The values stored in each following layer are then determined by layers processed previously. These networks' goal is to repeatedly *represent* their input at each layer as some increasingly useful list of numbers (i.e. as some increasingly useful

vector). But, what makes for a "useful" representation? Although there is no straightforward strategy, we can think of these representations as a learned series of composed *transformations* over the network's input space. Progressing from a network's input to its final layer, the composition of these transformations is often meant to model some complex phenomenon; for an arbitrary choice of input from the set of things that we're interested in examining, we want the network to represent it accordingly.

Although the precise internal abstract mechanisms used to accomplish this seemingly anisotropic and underspecified task are not entirely understood, it stands apparent that neural networks are capable of such behavior: on the one hand, we can use the language of optimization, function approximation, and loss gradients to give a satisfying general explanation for *why* neural networks are so expressively capable, which I review in the next chapter; on the other, we should seek to extend our formal account to explain *how* any specific, given neural network composes meaning from one layer to the next in a predictable, interpretable, and theoretically justifiable way.

Again, a given layer's representation of a given input to the network is stored as a single point in a high-dimensional vector space called an *activation space*, the size (dimension) of which is proportional to the number of neurons in that layer.[2] When a network represents something, one might expect that each meaningful *feature* of that input be represented within the network as some specific *direction* in the activation space.[3] For example, we would be delighted to discover that LLMs' reasoning over formal systems involves manipulating human-interpretable logical symbols hidden in some high-dimensional vector space, such as the symbol represented in Table 1.1. In this case, all the directions corresponding to this vector would be associated with a "logical implication" feature, which would be visible to anyone inspecting the hidden activations within an LLM.

In Figure 1.2, the x and y directions have approximately the meanings "decrease in peer review score" and "increase in the number of citations," respectively. The meaning of each datapoint within this space is represented as its distance

---

[2]This size, d, is constant from one layer to the next in all neural networks mentioned in this thesis unless otherwise noted.

[3]One definition of a feature is a symbol within the optimal encoding/embedding of that object. Another definition of a feature is a circuit within a network that represents an operation that humans can understand.
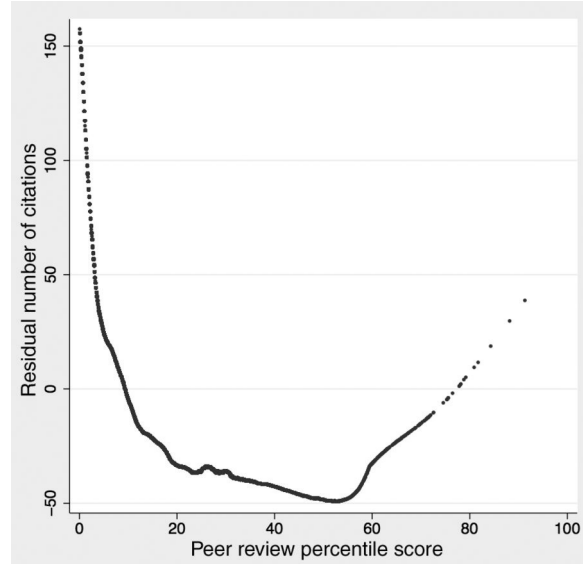
along these directions.

Figure 1.2: *Smoothed scatterplot of the nonparametric relationship between un-explained variation in grant application percentile score, after accounting for differences in field of research, year, and applicant qualifications. The panel plots the relationship between percentile scores and residual citations associated with a grant.* [6]

We would not, however, be able to use the same graph to also indicate a third dimension of information over each of these datapoints, for example, our degree of confidence in each point's accuracy. We would need to include some third direction z on the graph, e.g. color or height, which we can use to indicate our amount of confidence in each point. We need, then, to find a 3rd direction that's independent of, or orthogonal to, the first two directions x and y.

For the same reason that we are unable to simultaneously represent three features in $\mathbb{R}^2$, we should be unable to represent more than n unique features in $\mathbb{R}^n$, since any basis of $\mathbb{R}^n$ has only n directions as well [7]. But then, what if we simply added confidence intervals to Figure 1.2? Is there perhaps some clever way to make use of the underutilized areas of the two dimensions we do have in order to indicate information along a third?

Consistently and unexplainably, neural networks' representations are somehow capable of representing more unique features than the dimension of the space that they inhabit [8]. This ability is surprising, and contradicts the intuition that the atomic "unit" of meaning in the network is direction in the activation space; a Euclidean space $\mathbb{R}^n$ could not possibly contain an orthonormal set of more than n vectors.

An ongoing body of research suggests LLMs are able to exhibit this behavior because these n directions are actually approximations of some s^n orthogonal directions in an exponentially larger vector space wherein $s > 1$ and this "linear representation hypothesis" actually holds [9]. This is congruent with the Johnson–Lindenstrauss lemma, which states that for some error ε, there is a projection of points from a high dimensional space to an exponentially smaller one such that all pair-wise angles are preserved up to ε [10][4]. So, it's possible that some of the directions in a neural network's activation space with a privileged basis[5] represent multiple features *simultaneously* in the exponentially larger linear-representation space. The idea that a direction in the activation space can "mean" more than one thing has been dubbed the "Superposition Hypothesis" of neural network interpretability [12]. (See section 2.3.)

Sparse Autoencoders (SAEs) are a type of neural network that have been developed in order to learn the projections that other neural networks use to project between the higher-dimensional space where representative meaning is composed, and the lower-dimensional space to which they're constrained by their architectures. The "features" that SAES learn are the symbols that best help the network sparsely encode the input activation, meaning that they should correspond to such a projection. However, these symbols are not necessarily those that, given the same sparsity requirement, best explain the input to human interlocutors. My hypothesis is that the SAE features that tend to co-occur across many translations of a single given text are more indicative of the actual explainable semantic content of the activation, since these features are clearly independent of the language-specific formatting information that helps an SAE sparsely encode activation information. Unhelpful features like "whether the token starts with the letter c" or "words ending with -tion" should not be present across all translations

---

[4]Additionally, the maximum ratio between the two dimensions is also related to the constants ε and the number of points.

[5]Here, "privileged" simply means a basis to which embeddings are encouraged to align as a result of applying non-linearities, such as those that are used for activation functions [11].

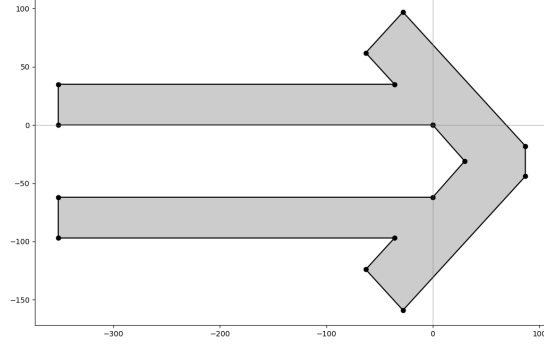| index	component $\in \mathbb{R}^2$
1	(30, -31)
2	(-30, -31)
3	(-352, 0)
4	(0, -35)
5	(316, 0)
6	(-27, -27)
7	(35, -35)
8	(115, 115)
9	(0, 26)
10	(-115, 115)
11	(-35, -35)
12	(27, -27)
13	(-316, 0)
14	(0, -35)
15	(352, 0)

Table 1.1: One discrete approximation of the "$\Rightarrow$" symbol as a vector in $(\mathbb{R}^2)^{15}$

of a text, but these exact "features" have in fact already been discovered in previous work on LLM interpretability using SAEs [13]. These types of "formatting" features are helpful in encoding specific activations, but not so much for human interpretability of the mechanisms that undergird and determine LLM conduct in general.

Background

ANY set of observations about a system (a *dataset*) can be uniquely represented as a set of points in a Euclidean space $\mathbb{R}^n$. This representation is defined by some element of an uncountably infinite group of transformations, encodings, or otherwise injective functions over all observations, ensuring that each possible unique observation have a unique representation [14].

For example, a phonetician might observe how someone produces a vowel, and then represent this observation as a three-dimensional point in $\mathbb{R}^3$, where the first axis corresponds to the speaker's tongue height, the second corresponds to the speaker's tongue backness, and the third corresponds to the vowel's length. A chemist might observe the outcome of a neutralization reaction, and represent their observation of the result as a point in $\mathbb{R}^5$, where the first axis corresponds to ΔpH, the second corresponds to the reaction time, the third to the amount of energy released, the fourth to the volume of CO_2 released, and the fifth to Δtemperature.

Crucially, our representation must uniquely represent *any* possible observation, not only those points already observed in our dataset. However, we can assume that we only make observations within a finite-dimensional domain; if there are aspects that, for whichever reason, we are uninterested in observing (e.g. how nasalized a vowel is, or the amount of air pressure present when the chemist performed their neutralization), then we do not need to represent these observed and discarded aspects in the first place.

We'll call the Euclidean representation of a possible observation the *embedding* of that observation, and say that our dataset is *embedded* in some n-dimensional Euclidean space $\mathbb{R}^n$. Thus, any single observation $\mathbf{x}$ can be viewed as a single discrete point in its *embedding space*. It's important to understand how these

representations work.

$$\langle \mathbf{x} \rangle \in \mathbb{R}^n$$

If aliens from another universe were to visit Earth, they would surely have trouble understanding many things. One immediate question they might have is why humans, who live on a fairly spherical Earth, insist on describing it as if it were flat. The aliens, you see, truly are from a flat planet, one which, seen from afar, resembles a sheet of paper.[6] We can understand, then, why they felt it was our fault when they crash-landed in the middle of the Pacific Ocean: "The maps you faxed us beforehand were flat!" they reproached, "You told us to land straight north from the Tropic of Capricorn, on something called an 'equator.' How were we to know that between the two lies an ocean? You said 'fly north!' Were we to know that you actually meant 'do *not* fly straight towards the north from the Tropic of Capricorn, instead fly following a great arc centered at the Earth's core'? Our planet doesn't even *have* a core. How are we to interpret anything you say, if we're not even sure what your coordinates refer to?"

Maybe our new friends are right. At the very least, we owe them an explanation. If the locations on planet Earth exist in three dimensions, why *do* we use only 2 dimensions in our coordinate system? Why is the origin of our coordinate system seemingly-arbitrarily located in the middle of the ocean, thousands of kilometers southwest of the West African Bulge? Why do our 2-dimensional coordinates become less meaningful when we're closer to the planet's poles? Why does the word "up" seem have opposite meanings when spoken on one side of the planet versus on the other? In general, why do the aliens have poor accuracy when they attempt to induct the meaning of novel data points based on previously observed representations? Would it be better to just describe things how the aliens expected us to, using Euclidean coordinates in $\mathbb{R}^3$ with the Earth's core at the origin?

2.1   Manifolds

Understanding certain properties of *manifolds* will be helpful in pinpointing where our communication with the aliens went wrong, helpful in understanding the potential benefits a 2-dimensional terrestrial coordinate system, and helpful in understanding the goals of this thesis.

---

[6]They're also from a universe with physical laws allowing such planets.

Broadly, a manifold is a mathematical object that must have certain simple properties when examined on small enough scales, but may have much more complicated properties when considered as a whole. Alternatively, a manifold is a topological space that when examined locally, resembles a Euclidean space. For example, consider the circle in Figure 2.3: when we look at it on a small-enough scale, we may be convinced that we're actually examining the straight line $\mathbb{R}$. Since a line is 1-dimensional, this makes a circle a 1-dimensional manifold, or a *1-manifold*. However, as we zoom back out to view more and more of our circle, we must eventually admit that it's not really a straight line.
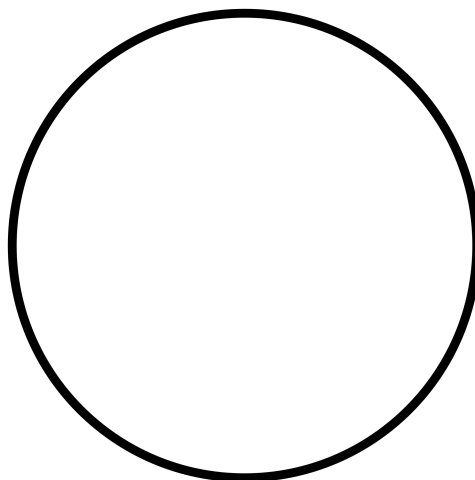

Figure 2.3: A plot of the manifold $S^1 = \mathbb{R}/\mathbb{Z}$. I use the quotient symbol to define S^1 as the classes under the equivalence relation $\forall x, y \in \mathbb{R} : (x - y) \in \mathbb{Z} \Leftrightarrow x \sim y$. *The circle is all places where continuing to the next place leads you to the one where you began.*

Formally, a topological space of a certain dimension is only a manifold if each point in that space has a neighborhood that's homeomorphic to an open subset of the Euclidean space with that dimension [15]. Notice that although the manifold S^1 in figure 2.3 is shown embedded in the 2-dimensional space $\mathbb{R}^2$, we do not strictly *need* two dimensions to describe a given point on this embedding. If we start by using the manifold's embedding space, $\mathbb{R}^2$, as its coordinate system, then

we would indeed need two coordinates to describe a point on the circle. For example, consider the observation "it's 11:00 o'clock." Let's call this proposition "**x**." Akin to the hour hand of an analogue clock, we could to describe this observation **x** on our circle in the typical way by its Cartesian coordinates as

$$\langle \mathbf{x} \rangle = \left(\frac{-1}{2}, \frac{\sqrt{3}}{2} \right) \in \mathbb{R}^2.$$

However, if we instead used a coordinate system designed specifically for our manifold, then we would no longer need two coordinates to represent points on it. For example, we could instead describe any point on the circle by the clockwise angle formed at the origin between that point and some fixed reference. So, by defining a new fixed-length polar coordinate system and basing our angle measurements from $(0, 1) \in \mathbb{R}^2$, we can represent the same point from before with a single dimension:

$$\langle \mathbf{x} \rangle' = 330° \in \mathbb{R}/360\mathbb{Z}.$$

Although both $\langle \mathbf{x} \rangle$ and $\langle \mathbf{x} \rangle'$ are descriptions of the same observation **x**, this new description method allows us to cut what we need to communicate about our observation by at least half while maintaining unique representations of each point on the manifold. Notice that this, too, tells us that a circle is a 1-manifold: "a manifold resembles an n-dimensional space upon close inspection" is equivalent to "there exists an atlas of n-dimensional charts[7] that effectively represent that manifold" is equivalent to "that is an n-manifold."

The aliens' notion of communication as predicated on a shared coordinate system generalizes well. Instead of the Cartesian coordinates of a point on a circle, we can instead define a basis once and give its angle. Instead of giving an explicit formula for each monic polynomial $p : \mathbb{C} \to \mathbb{C}$, we can instead define its degree once, and give its roots. Instead of the full description of a data point and its image under some function, we can instead define the function once and give only the data point. Instead of picking out something from a very large space, we can define a smaller manifold embedded within that space once, and pick out things from there.

---

[7]A chart is a homeomorphism from an open subset of a manifold to an open subset of $\mathbb{R}^n$. A chart can be intuited as one of the pieces of an invertible piecewise function that takes us from the old coordinate system to a new one. An atlas is a list of charts, and can be viewed as the complete function.

However, there's also a cost to this trick. As in the example with the aliens, using a more optimized coordinate system reduced the *generality* of our representations. Before telling the aliens where to land, we first would have needed to fax them an explanation of how our terrestrial coordinate system works. Before communicating a clock time on our circle, we first needed to establish which coordinate system we were using, and communicate in the old coordinate system that we chose $(0, 1) \in \mathbb{R}^2$ to be the fixed basis of our new system. By tailoring our coordinate system like this, we also lost the ability to communicate any points *not* on the manifold S^1, e.g. $(1, 1) \in \mathbb{R}^2$. This demonstrates that although using a manifold's ambient space is rarely the most efficient method of communicating points on that manifold, we maintain general *interpretability* by using ambient coordinates to embed our data, as we already understand what these coordinates mean.

Since we want to strike a balance between how much effort must go into communicating a given message (i.e. the dimension of that data point's Euclidean embedding space) and how much effort must go into parsing that message (i.e. how much effort must go into converting a data point back into an accurate estimate of the original observation), it would be helpful to have an upper bound for the dimension of the Euclidean space needed to embed a manifold. Our space must be large enough to enable meaningful *and interpretable* descriptions of the points on our manifold, yet not so large that our coordinate representations become cumbersome for us[8] to communicate and evaluate. If we continue to take "interpretable" to mean "Euclidean," we can interrogate this bound with our earlier example of the circle.

Although we can devise a one-dimensional coordinate system to describe a circle's points, this doesn't show us how we can embed a circle in a one-dimensional Euclidean space. In fact, I'm confident you can convince yourself that it is not possible to deform a circle into a line[9] whatsoever, and that any circle represented in the Euclidean space $\mathbb{R}$ has necessarily lost a crucial part of its structure. If we allow an additional dimension, we can of course embed a circle in $\mathbb{R}^2$ instead, as shown in figure 2.3. So, $\mathbb{R}^2$ is the smallest Euclidean space in which we can embed the manifold S^1. The useful observation here is that at times we may need an embedding space with a higher dimension than the manifold itself.

---

[8]Rather, cumbersome for *computers*.

[9]or into a line segment or into a ray

2.1.1   Approximation

The size of this difference was concretely bound and generalized to all smooth manifolds by Hassler Whitney in 1936:

Theorem 1 (**Whitney's Embedding Theorem**). *[16, Theorem 2]*

.

Any smooth m–manifold M can be embedded as $M \hookrightarrow \mathbb{R}^{2m}$.[10]

Theorem 2 (**Regular Value Theorem**). *[18, Theorem 5.14], [19, Theorem 9.9]*

If M is orientable,[11] *then M is also a level set of some differentiable function $\theta^* : \mathbb{R}^{m+1} \to \mathbb{R}$ such that 0 is a regular value of θ^*, i.e. that $\theta^{*-1}(0) = M$*[12] *and $\forall x \in \mathbb{R}^{m+1} : \theta^*(x) = 0 \implies \nabla\theta^*(x) \neq 0$.*

Definition 2.1.1.

A subset $X \subseteq \mathcal{X}$ is a level set *in $n - k$ variables of a function $g : \mathcal{X} \to \mathcal{Y}^n$ if and only if there exists some constant $c \in \mathcal{Y}^k$ in a subspace of the codomain $\mathcal{Y}^n$ such that X is the set of all points in $\mathcal{X}$ whose image under g has fixed coordinates equal to c. Then,*

$$X = \{x \in \mathcal{X} : \pi_k \circ g(x) = c\},$$

---

[10]The upper bound on the necessary ambient dimension of a smooth manifold's embedding, regardless of orientability, was reduced in [17] from $2m$ to $2m - \sum_{i \geq 0} \left(\left\lfloor \frac{m}{2^i} \right\rfloor \bmod 2 \right)$, i.e. to $2m$ minus m's *Hamming weight*. This suggests that if a manifold's dimension has an especially *high* Hamming distance to a string of zeroes, it can be embedded in an especially *low*-dimensional Euclidean space.

[11]Loosely speaking, a manifold is *orientable* if it admits a notion of "handedness." The Möbius strip, for counterexample, is a smooth manifold, but one that does not permit any consistent notion of a "left-" or a "right-hand side," and as such is *not* orientable. The manifolds concerned in this thesis are assumedly orientable.

[12]The functional inverse we're applying here, $\cdot^{-1} : \mathcal{Y}^{\mathcal{X}} \to 2^{\mathcal{X}^{\mathcal{Y}}}$, is defined not only over injective functions but over non-injective functions as well: the inverse of any non-injective function $g : \mathcal{X} \to \mathcal{Y}$ is the set given by $g^{-1} : y \overset{\text{def}}{\mapsto} \{x \in \mathcal{X} : g(x) = y\}$. For example, for inversions of set-valued functions, the inverse of a set $Y \in \mathcal{Y}$ will be defined as $g^{-1} : Y \overset{\text{def}}{\mapsto} \{x \in \mathcal{X} : g(x) \in Y\}$. In a similar vein I may at times write a function g over some subset of its domain $X \subseteq \mathcal{X}$, by which I mean $g : X \overset{\text{def}}{\mapsto} \bigcup_{x \in X} g(x)$.

where $\pi_k : \mathcal{Y}^n \to \mathcal{Y}^k$ is a fixed projection onto a k-dimensional subspace. Put more simply, X can be described as the set of points x whose image $g(x)$ has certain k coordinates fixed to c.

Intuitively, a level set is the collection of all points that, when mapped by our function, share the same constant values in certain dimensions while potentially varying in others. For instance, the surface of a sphere can be viewed as the level set of the function $f(x, y, z) = x^2 + y^2 + z^2$ with the constant value $c = 1$. Here, all points (x, y, z) on the sphere satisfy a single constraint while their individual coordinates remain free to vary, so long as they collectively maintain that constraint. Level sets allow us to identify structures within higher-dimensional spaces by fixing only the dimensions that define the structure's essential properties.

Although we lack an explicit construction, we know that for an m-manifold M that satisfies these conditions, such a function θ^* exists. Let

$$\Theta = \{\theta : \mathbb{R}^{m+1} \to \mathbb{R} \,|\, \forall x \in \mathbb{R}^{m+1} : \theta(x) = 0 \implies \nabla\theta(x) \neq 0\}$$

be the *parameterized family* of all functions that satisfy our known constraints on θ^*. By the Regular Value Theorem, we also know that θ^* is a member of Θ such that $\theta^{*-1}(0) = M$.

Let f be a function that maps from elements of Θ to their corresponding level sets at 0 in $\mathbb{R}^{m+1}$.

$$f : \Theta \to 2^{\mathbb{R}^{m+1}} : \forall\theta \in \Theta : f(\theta) \mapsto \theta^{-1}(0) = \{x \in \mathbb{R}^{m+1} : \theta(x) = 0\}$$

Then, by definition, $f(\theta^*) = M$.[13]

Postulate 2.1.1.

The goal of machine learning is to use a dataset $\mathcal{D}$ of n points x sampled from M and an algorithm Φ equipped with architecture $\hat{f}$ to find a single function $\hat{\theta} \in \Theta$ whose corresponding level set $f(\hat{\theta})$ best approximates M.

$$\Phi\left(\hat{f}, \mathcal{D} = \{x_i\}_{i=0}^n \,\middle|\, \left\langle \hat{f} \approx f, \mathcal{D} \subsetneq M \right\rangle\right) \mapsto \hat{\theta} : \hat{\theta} \in \Theta : f(\hat{\theta}) \approx f(\theta^*) = M$$

---

[13]There's additional topology with which we equip this construction which isn't relevant here.

However, immediate roadblocks arise:

1. The explicit form of f is unknown, and in general, not computationally tractable.

2. Even if f were known, the probability that $f^{-1}(\mathcal{D})$[14] intersects meaningfully with $f^{-1}(M)$ is extremely small since $\mathcal{D} \subsetneq M$.

There's seldom a simple search, sift, or sort over Θ that surely yields θ^* precisely. In fact, we almost always must instead make do with finding a function that approximates θ^* as closely as possible. My claim that observed datasets are sampled from a manifold is discussed more in the next section. However, assuming for now that it is true, one way to find some $\hat{\theta} \in \Theta$ that approximates θ^*, given only $\mathcal{D}$, might be to assume that our dataset $\mathcal{D}$ (which we know) is fairly representative of our manifold M itself (which we do not know),[15] and to calculate its inverse

$$\hat{\theta} \in f^{-1}(\hat{M}) \overset{?}{\approx} f^{-1}(M) = \theta^*$$

explicitly, where $\mathcal{D} \approx \hat{M}$ is an approximate embedding of our dataset into M's ambient dimension. However, calculating

$$f^{-1}(\mathcal{D}) = \{\theta \in \Theta : f(\theta) = \hat{M}\} = \{\theta \in \Theta, x \in \mathbb{R}^{m+1} : \theta(x) = 0 \Leftrightarrow x \in \mathcal{D}\}$$

is an incredibly difficult and ineffective way of approximating θ^*, for a number of reasons:

1. We don't actually know what f is; so far the only things we know about f is that f is a differentiable function from Θ to subsets of $\mathbb{R}^{m+1}$ and that

---

[14]I'll refer to *any* arbitrary approximate embedding $\mathcal{D} \cong \hat{M} \hookrightarrow \mathbb{R}^{m+1}$ as simply $\mathcal{D}$ going forward.

[15]To account for the possibility that our observations $x \in \{x_i\}_{i=1}^n = \mathcal{D}$ are imperfect approximations of actual points on M, we can either write instead that our dataset is sampled from a related noisy distribution, e.g. $\mathcal{D} = \{x_i\}_{i=0}^n, x_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(M) + \mathcal{N}(\mu)(\sigma)$. Going forward we may just assume that M is a smooth manifold induced by some hidden lower-dimensional-still information manifold M' when noise is applied to M', e.g. $x_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(M) = \text{Unif}(M') + \mathcal{N}(\mu)(\sigma)$. Even if this were a questionable assumption in general, this thesis examines text-based datasets, which ensures a high resolution and correspondence between our recorded observations $x \in \mathcal{D}$ (digital text) and the real-world information about the objects that they are meant to represent (words). This might not be the case with noisier observations, such as if we were attempting to use photos of a certain resolution to sample from the manifold of all photos, or, say, using digital song lyrics to sample from the space of all speech. In these cases we should more rigorously evaluate how such noise is to be modeled.

$\mathcal{D} \subsetneq f(\theta^*)$. Therefore, we must approximate f using a suitable *architecture* $\hat{f} : \Theta \to 2^{\mathbb{R}^{m+1}}$. Unfortunately there are uncountably infinite such functions fitting the bill. So, all else being equal, the probability that f is even a *computable* function is 0. This roadblock is an example of a *choice problem*, to which our only solution is approximation with $\hat{f}$. Further, since f is defined as an inverse image, there is no reason for calculating it to be straightforward. Without assuming further algebraic structure or access to pre-computation, a naïve approach might take infinitely longer to evaluate the true f than it would to evaluate some θ. This issue, which is related to the size of θ's domain $\mathbb{R}^{m+1}$, has been labelled the *Curse of Dimensionality.*

2. In our setup, $M \subsetneq \mathbb{R}^{m+1}$ is the unknown target manifold, and $\mathcal{D}$ is a finite sample drawn from M. So, $\mathcal{D}$ provides only a partial approximation of M, and by itself, does not uniquely determine M. Requiring exactly $\mathcal{D} \in f(\hat{\theta})$ enforces the data points to lie on the zero-level set. However, infinitely many level sets can *interpolate* a finite collection of points. As the size of $\mathcal{D}$ grows, we expect elements of $f(\hat{\theta})$ to approximate M more accurately. So, we design a loss function that encourages the learned level set to pass close to the samples without over-constraining $f(\hat{\theta})$ based only on the finite dataset $\mathcal{D}$.

3. Even if we were somehow able to efficiently calculate $f : \Theta \to 2^{\mathbb{R}^{m+1}}$, as we have currently defined things, there is still a probability of 0 that we would be able to sample an element of $f^{-1}(\mathcal{D})$ to approximate θ^*. It's highly unlikely that these preimages share an intersection because wherever $\theta(x)$ is zero, which would be the case for any ideal θ when $x = M$ or when $x = \mathcal{D}$, $\nabla\theta(x)$ must be nonzero in both cases. This poses an issue with our strategy of using $f^{-1}(\mathcal{D})$ to find a $\hat{\theta}$ that approximates θ^*, since our strict requirement that our $\hat{\theta}(\mathcal{D})$ be exactly equal to 0 means that $\hat{\theta}(M)$ can never be equal to 0 as well, since any method of *gradient descent* (discussed below) brings us in the direction of $\mathcal{D}$, where the gradient is necessarily non-zero. So, since we are assuming that the only information we have about M is $\mathcal{D}$, in order to find a satisfying $\hat{\theta}$ (i.e. one that is as close as possible to θ^*), we must relax our requirement that $f(\hat{\theta})$ exactly equal $\mathcal{D}$, and / or we must relax our requirement that $f(\hat{\theta})$ exactly equal M. Implementing these relaxations is not easy, and involves striking a balance between $\hat{\theta}$'s levels of *bias* and *variance* relative to θ^*. When we relax the former, we choose a $\hat{\theta}$ that has a higher bias, meaning that $f(\hat{\theta})$ is not particularly close to $\mathcal{D}$, and by extension, not particularly close to M either. This issue

is called *underfitting*. When we relax the latter, choosing a $\hat{\theta}$ that has a higher variance, $f(\hat{\theta})$ is instead extremely close to $\mathcal{D}$, and in fact so close to $\mathcal{D}$ that $f(\hat{\theta})$ has moved past M, the manifold we are actually interested in capturing in the first place, instead approaching just the subset $\mathcal{D} \subsetneq M$. This issue is called *overfitting*.

In sum, we want to use our dataset $\mathcal{D}$ to find a $\hat{\theta} \in \Theta$ that is *close* to θ^* by minimizing $\hat{\theta}$'s bias (e.g. $f(\hat{\theta})$'s distance to M), by moving $\hat{\theta}$ in the direction of $f^{-1}(\mathcal{D})$. At the same time, we want to ensure that $f(\hat{\theta})$ not get *too* close to to $\mathcal{D}$ such that it begin to move away from M. We achieve this by restricting, in one way or another, how complex or irregular $\hat{\theta}$ may be. This restriction is meant to ensure that $\hat{\theta}$'s "complexity budget" be spent only along those dimensions that are most important for approximating $f^{-1}(\mathcal{D})$, which we hope are also the dimensions most important for approximating $f^{-1}(M)$, without allowing an excess of wanton regard for the exact datapoints on M that happen to be included in the dataset $\mathcal{D}$. We can do this by choosing some random starting element $\theta_0 \sim k(\Theta)$, and then moving θ_0 in the direction of θ^* t times. To accomplish this strategy, called *gradient descent*, we'll first need to build a few things:

1. The first thing we need is an explicit construction of $\hat{f} : \Theta \to 2^{\mathbb{R}^{m+1}}$, which we hope is similar to the true hidden function $f : \Theta \to 2^{\mathbb{R}^{m+1}}$, which we don't know. This function $\hat{f}$ is called Φ's *architecture*. Since $\hat{f}$ reflects our prior beliefs about the relationship between Θ and M, $\hat{f}$ is also called a *prior* on our model. We have no reason to believe that f is a linear function, and we want $\hat{f}$ to be a highly expressive function so that it may capture as much about f and Θ as we might like it to capture. As we see soon, at times we may also want to calculate partial derivatives of functions composed with $\hat{f}$. So, we often choose a prior architecture $\hat{f}$ that is a non-linear and differentially parameterized function. However, since our choice of $\hat{f}$ is related to the specific manifold that we are trying to learn, $M \hookrightarrow \mathbb{R}^{m+1} \subsetneq \mathbb{R}^{2m-1}$, there is not much else we can say *in general* about what qualities a good machine learning architecture should have. Certain architectures work better than others for learning representations of certain types of manifolds.

2. We approximate θ^* by choosing a point $\theta_i \in \Theta$ that's increasingly closer and closer to θ^*. To ensure that we're actually bringing θ_i in the direction of θ^*, we need a way of measuring the distance between $\hat{f}(\theta_i)$ and $f(\theta^*) = M$. We can define the distance between a single function θ_i and the unknown

$\theta^* \in \Theta$ as

$$\mathcal{L}_{\mathcal{D}} : \Theta \to \mathbb{R}_{\geq 0} : \forall \theta_i \in \Theta : \mathcal{L}_{\mathcal{D}}(\theta_i) \mapsto \mathbb{E}_{x \sim \mathrm{Unif}(\mathcal{D})}[\mathrm{dist}(x, f(\theta))],$$

where $\mathrm{dist}(x, f(\theta))$ is the mean distance from x to the elements of $f(\theta)$. (We later modify this sampling procedure to measure the distance to a subset of the elements of $\mathcal{D}$ rather than to the entire dataset.)

3. Finally, we need to somehow guarantee that our algorithm Φ continually bring our point θ_i closer to θ^*. Iteratively, at each time step i, we want to guarantee that

$$\Phi(\mathcal{D}, \theta_i) = \theta_{i+1}$$

$$\mathcal{L}_{\mathcal{D}}(\theta_i) > \mathcal{L}_{\mathcal{D}}(\theta_{i+1}).$$

If we can achieve all of the the above, then we know that the distance between $\hat{f}(\theta_i)$ and $\mathcal{D}$ must converge to a minimum

$$\lim_{i \to \infty} \mathcal{L}_{\mathcal{D}}(\theta_i) = \epsilon \in \mathbb{R}_{\geq 0}.$$

When calculated like this, however, we actually have no guarantee that this minimum ϵ be a global minimum of $\mathrm{Im}(\mathcal{L}_{\mathcal{D}})$. But, for reasons discussed above, we don't actually want to converge to the global minimum $\arg\min_{\theta \in \Theta} \mathcal{L}_{\mathcal{D}}(\theta)$, since this would mean that we're fitting our model's parameters perfectly to our dataset $\mathcal{D}$, and $\hat{f}(\arg\min_{\theta \in \Theta} \mathcal{L}_{\mathcal{D}}(\theta)) \approx \mathcal{D}$ is likely quite a poor approximation of M (and perhaps poorer still under f than under $\hat{f}$). There is even no guarantee that the global minimum of the true, unobserved function $\mathcal{L}_{\theta^*}$ be θ^* (unless for some reason M is known to have a convex embedding). It's even less plausible that the global minimum of our approximation of this function, $\arg\min_{\theta \in \Theta} \mathcal{L}_{\mathcal{D}}(\theta)$, be θ^*, if we are even able to find it in the first place. In fact, even in this case where we actually have somehow found the global minimum of our unknown function θ^*, it is possible that for certain f,

$$\theta_i = \arg\min_{\theta \in \Theta} \mathcal{L}_{\theta^*}(\theta) \notin \theta^*$$

and so

$$f(\theta_i) \neq M.$$

2.1.2 Backpropogation

To implement this Φ as a concrete algorithm, we rely on a technique called *backpropagation*. Backpropagation is a mechanism of computing the gradients of our loss function with respect to the parameters of our model, $\nabla \mathcal{L}_{\mathcal{D}}(\theta_t)$. We use backpropogation to update our parameters, for as many iterations as we like, in a way that consistently decreases the loss of the current parameters,

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta_t),$$

here $\alpha \in \mathbb{R}_{>0}$ is a scalar value called the *learning rate* that determines the size of each update's step, i.e. how "far" we are moving in Θ'. This gradient-based update rule is the physical implementation of Φ that we use in practice. With an appropriately chosen learning rate, this implementation locally guarantees that $\mathcal{L}_{\mathcal{D}}(\theta_t) > \mathcal{L}_{\mathcal{D}}(\theta_{t+1})$.

To compute $\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta_t)$ efficiently, backpropagation uses the *chain rule*. This means it actually works *backward* through our model, in order for the model to behave as desired when we pass values through it *forward*. Concretely, for our parameterized function $\hat{f}$, the partial derivative of the loss with respect to some specific parameter $\theta_j \in \theta$ is

$$\frac{\partial \mathcal{L}_{\mathcal{D}}(\theta)}{\partial \theta_j} = \mathbb{E}_{x \sim \text{Unif}(\mathcal{D})} \left[\frac{\partial \hat{\ell}(\hat{f}^{-1}(x), \theta)}{\partial \hat{f}(\theta)} \cdot \frac{\partial \hat{f}(\theta)}{\partial \theta_j} \right]$$

For a neural network, where $\hat{f}$ is a composition of multiple functions (i.e. of multiple layers), the chain rule extends to

$$\frac{\partial \mathcal{L}_{\mathcal{D}}(\theta)}{\partial \theta_j} = \mathbb{E}_{x \sim \text{Unif}(\mathcal{D})} \left[\frac{\partial \hat{\ell}}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial a^{(L-1)}} \cdot \cdots \cdot \frac{\partial a^{(l+1)}}{\partial a^{(l)}} \cdot \frac{\partial a^{(l)}}{\partial \theta_j} \right],$$

where $a^{(l)} \in \mathbb{R}^d$ represents the activations at layer l of the network. The analytic beauty of backpropagation lies in its ability to reuse these intermediate gradient computations, and calculate gradients only once per layer and propagating them backward through the network, instead of having to recompute the full path of each parameter's gradient.

However, for a dataset of non-trivial size, computing this expectation over all data points does become computationally difficult. In practice, we often ap-

proximate this expectation using *mini-batch* stochastic gradient descent by only calculating gradients on small, randomly selected subsets of the dataset

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta_t) \approx \frac{1}{|B|} \sum_{x \in B} \nabla_\theta \hat{\ell}(\hat{f}^{-1}(x), \theta_t),$$

where $B \subsetneq \mathcal{D}$ is a randomly sampled mini-batch. This scheme gives a implementable construction of Φ that ensures that as $t \to \infty$, θ_t approaches a local minimum of $\mathcal{L}_\mathcal{D}$ in almost any case we would want to use it. How close this minimum actually is to θ^* depends on several factors, such as the quality of our dataset, the quality of our priors, how we initialize θ_0, the choice of α, and how well $\mathcal{D}$ represents M. Empirically, by using regularization techniques, this works quite well in high-dimensional settings at finding a θ_t such that $\hat{f}(\theta_t)$ is a approximation of M, even if it can't guarantee convergence to the true manifold M. This thesis can be partially viewed as an examination of how more sophisticated batch-selection and gradient-accumulation techniques than random and independent selections can improve this convergence.

2.1.3 The Manifold Hypothesis

The weaker interpretation of Whitney's Embedding Theorem is that *any compact manifold M*, not just those that are smooth, can be embedded into a Euclidean space $\mathbb{R}^N$ for a sufficiently large N [16, Theorem 1]. Although a dataset embedded in a high-dimensional Euclidean space *may* contain embeddings of many lower-dimensional manifolds, under what conditions should we expect it to? According to Whiteley, Gray, and Rubin-Delancy, the answer is "always:"

> *The Manifold Hypothesis is a widely accepted tenet of Machine Learning which asserts that nominally* **high-dimensional data are in fact concentrated near a low-dimensional manifold, embedded in high-dimensional space.** *This phenomenon is observed empirically in many real world situations, has led to development of a wide range of statistical methods in the last few decades, and has been suggested as a key factor in the success of modern AI technologies.* [21]

The Manifold Hypothesis posits that a large enough collection of observations of high-dimensional data points, i.e. vectors that reflect real observations, does not span the full space of possible embeddings, and further, approximates a manifold with a much lower dimension than the space in which it's embed-
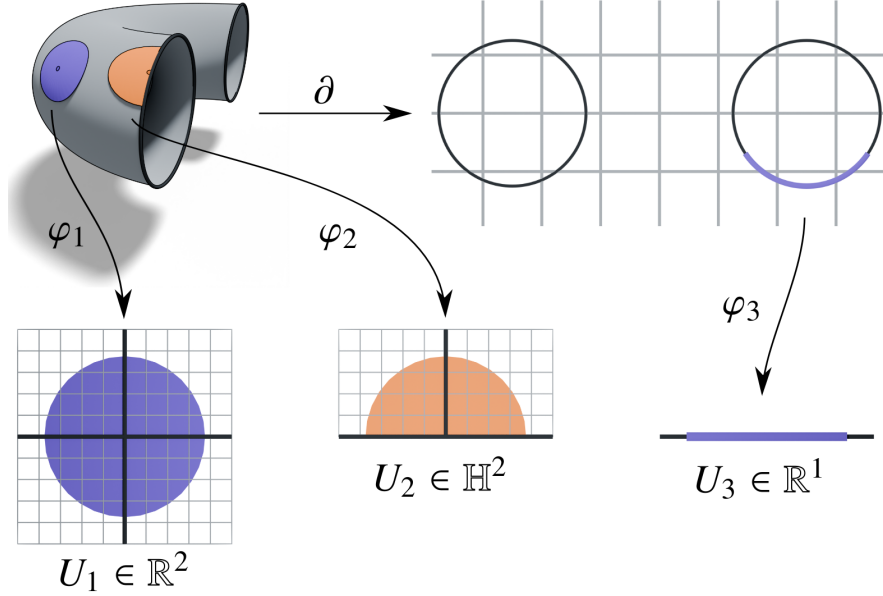
Figure 2.4: Cross-section of the manifold $\mathbb{T}^2 = S^1 \times S^1$. Discovering only certain embedded submanifolds, e.g. $\partial \mathbb{T}^2 = S^1 \sqcup S^1$, can be especially informative about the overall structure of the larger manifold. [20]

ded. Thus we expect to be able to efficiently describe high-dimensional data with low-dimensional coordinates when those data reflect observations over some high-dimensional domain. For language models specifically, this hypothesis suggests that the embeddings from different languages may converge toward shared semantic spaces that are far more compact than their surface realizations would suggest, despite the vast differences linguistic surface forms.

> *The human brain confronts the same problem in everyday perception, ex-*
> *tracting from its high-dimensional sensory inputs—30,000 auditory nerve*
> *fibers or 10^6 optic nerve fibers—a manageably small number of perceptu-*
> *ally relevant features.* [14]

The caveat, however, is that we may only discover such a coordinate system once we have observed and processed (or *learned* [22–24]) enough high-dimensional data points to be confident that our low-dimensional manifold will continue to align with future observations; before we can fit a coordinate system to a low dimensional manifold, we must have a good guess about what the manifold *is*.

Concretely, before the phonetician can claim that a speaker's vowel space is concentrated in the back of the mouth, they must first collect enough data points to be confident in this claim. Once they are, they may make predictions about the distribution of future allophonic variation in vowels. Before the chemist can claim a relationship between a reaction's change in pH and the amount of CO_2 it releases, they too must collect enough data points to be confident of this relationship. Once they have, they may make predictions about how a novel reaction that *absorbs* CO_2 will change in pH. In both cases, our observer must also establish whether the other factors that they're measuring (e.g. vowel length or change in temperature) reliably interact with this relationship. Once we are able to establish in which regions of the embedding space we expect new observations to lie, and in which regions we would be surprised to find new observations, we have begun to *model* our high-dimensional system using lower-dimensional representations.
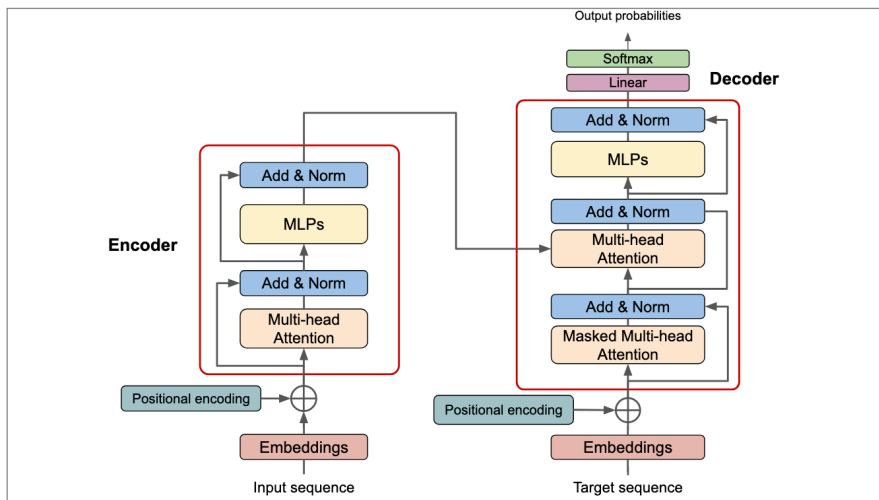

Figure 2.5: Computational graph of the transformer architecture showing how token embeddings flow through attention layers to produce activations [4]

Recent work has shown that language-independent representations emerge gradually through the layers of LLMs. Early layers tend to focus more on language-specific features like syntax and morphology, while deeper layers develop more abstract, semantic representations [25]. This hierarchical organization suggests that the manifold of semantic concepts is constructed through successive transformations, each layer initially bringing the representations closer to a language-agnostic form.

2.2   Interpretability

Understanding how LLMs arrive at their outputs, a field known as LLM *interpretability*, has become increasingly important as these models are deployed. A central question to this field is how the convoluted and dense activation-vector representations of language within an LLM can be decomposed into a small set of human-interpretable *features*. Traditional approaches to neural network interpretability, such as analyzing individual neurons, PCA, sparse probing, model diffing, influence functions, or attention patterns, have provided fairly limited insights when applied to modern language models. This limitation likely stems from the complex and unexpected ways in which information is distributed across the network's activation space, since these methods rely on the assumption that the activation space is congruent with the model's concept space, without regard for superposition or other factors.

In order to understand the behavior of these models, we must be able to describe the features that determine that model's behavior. The trajectory of work in this area has taken "feature" to mean any aspect of a network that closely represents a predicate that can be understood by a human researcher. Early interpretability efforts focused on identifying specific neurons or attention heads that appeared to encode interpretable features within a model. For instance, researchers found attention heads that seemed specialized for tasks like token repetition or resolving syntactic coreference [26]. However, these initial *circuit-based* approaches faced several challenges:

1. Important features are often encoded across many neurons rather than in individual units [27].

2. The same neurons often represent different features depending on the context [28].

3. Multiple features might share the same neural resources through interference patterns [12].

More recent approaches have attempted to understand model behavior through dimensionality reduction techniques like UMAP [29] or t-SNE [30]. While these methods can reveal clusters and patterns in the activation space, they often struggle to capture the full complexity of the representations, and they impose a uniform threshold for a feature's detection. This is particularly true when dealing with multilingual models, where the same semantic concept might be encoded

with varying strength and uniformity across languages.

A key challenge in interpretability research is the apparent contradiction between the dimensionality of the activation space and the richness of the representations it contains. Modern LLMs can distinguish between and manipulate an enormous number of concepts, yet they somehow do so using activation vectors of relatively modest dimensionality. This observation has led to a fundamental shift in how we think about neural representations.

2.3 The Superposition Hypothesis

According to the Superposition Hypothesis, the activation vectors we observe inside LLMs and other neural networks are projections from an exponentially larger space where semantic concepts might be represented more cleanly into the activation space. This is where the hypothesis becomes relevant to interpretability: rather than each concept having its own dedicated direction in the activation space, multiple highly unrelated concepts might share the same dimensions through complex patterns of interference and superposition.

The traditional view assumed that each feature or concept would require its own dimension in the activation space, similar to how a phonetician or chemist might represent a specific variable through one-hot encoding. However, this view fails to explain how models can represent more scalar concepts than they have dimensions. This limitation of the traditional view has led to the development of the "Superposition Hypothesis," [12] which suggests that neural networks can efficiently encode far more features than their dimensional capacity would suggest by allowing features to share dimensions through carefully orchestrated feature overlaps. This mechanism is posited to exist within and motivate the expressive capabilities of all neural networks, including the LLMs I investigate here.

If the latent variables with which a model "reasons" / problem-solves only form a manifold embedded in a much larger ambient space than the activation space, then the counterintuitive abilities of LLMs discussed earlier would make much more sense. If the restrictions on the space were relaxed during training, and it only had to maintain the manifold's orientable normal bundles up to some very small deviation in the norms, then that manifold's embedding may need to grow significantly (even exponentially) before it becomes interpretable. This perspective has important implications for how we approach model interpretabil-
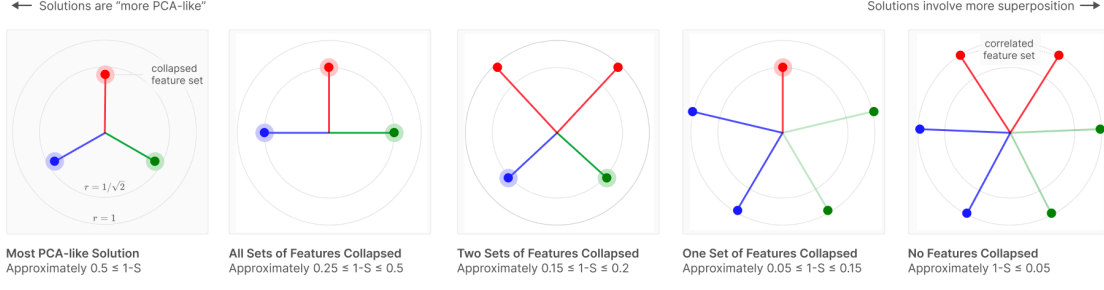
Figure 2.6: How > 2 features could be encoded in the 2-dimensional activation space $\mathbb{R}^2$ through superposition [12]

ity. If features are indeed stored in superposition, then traditional methods like PCA that assume orthogonal feature directions may be fundamentally limited. This realization has motivated the development of new interpretability techniques, particularly sparse autoencoders, which attempt to disentangle these superposed representations into more interpretable features.

2.4 Sparse Autoencoders

Our method for feature analysis, instead of approximating the representation of features orthogonal to the activation space, should approximate the principle components of the transformation itself from the large superimposed space to the smaller activation space [31]. This method allows one to discover features in superposition, without first assuming that features exist within a uniform structure in the encoding of the LLM. Recently, much work in mechanistic language model interpretability has concerned training SAEs and their variants to make this approximation. If we assume the superposition hypothesis, then these features are only encoded by linear composition within the exponentially larger linear-representation space, but not strictly by individual activation directions.

A three-layer textbook autoencoder (Figure 2.7) consists of an input "encoding" layer of n_{enc} neurons, an output "decoding" layer of the same size $n_{\mathrm{dec}} = n_{\mathrm{enc}}$, and a hidden "latent" layer in-between and smaller than both, with size $n_{\mathrm{hid}} < n_{\mathrm{enc}}$. The goal is to train the network to reconstruct its inputs efficiently, given the intermediate constraint. The network must learn to represent the input activation as precisely as possible after it is projected onto the much smaller subspace of the hidden layer by optimizing these non-linear projections for accuracy.

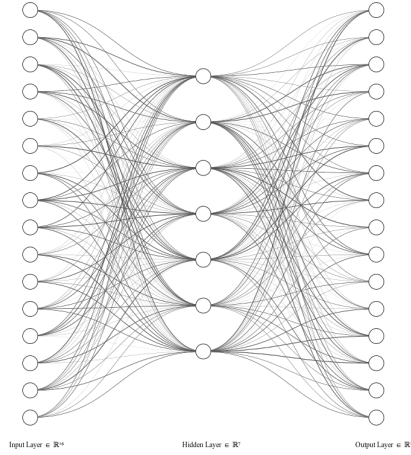Input Layer ∈ ℝ⁴⁸   Hidden Layer ∈ ℝ⁷   Output Layer ∈ ℝ⁴⁸

Figure 2.7: Autoencoders are meant to compress their input to a lower-dimensional encoding, representing it with less neurons, and then use no information besides this representation to reconstruct the original input on the other side. In doing so, one hopes that each of the fewer neurons / dimensions in the hidden middle layer represent some useful, salient, or informative feature of the inputs.

Sparse autoencoders (Figure 2.8) have two architectural differences from standard autoencoders: the hidden latent dimension, and the loss function used for training. These changes allow SAEs to be used to reconstruct the features of an activation vector by *sparsely* encoding it as a vector in a much *larger* latent space. The hidden layer of a SAE has a dimension of $\lceil x^{n_{\text{enc}}} \rceil$ for some $x \in \mathbb{R}_+$, which is much larger than the dimension of the other two layers, unlike in a standard autoencoder where it is strictly smaller. This accounts for our expectation that an activation vector of length n_{enc} somehow represents a subset of $x^{n_{\text{enc}}}$ feature directions in a larger space.

The second key difference is in the loss function. While standard autoencoders typically use only a reconstruction loss term (such as mean squared error between the input and output), sparse autoencoders need additional regularization terms that explicitly enforce sparsity in the hidden representations. Common approaches include L1 regularization, which penalizes the absolute values of activations. KL-divergence terms can also constrain the average activation of hidden units to a specific target value. These sparsity penalties ensure that for any given

input, only a small subset of the *dictionary elements* in the hidden layer activate significantly at a time, encouraging the model to learn more interpretable and disentangled features. The sparsity constraint is critical because without it, the network would simply learn to use all available dimensions to copy its input to its output directly, rather than approximating the data's underlyingly sparse structure.
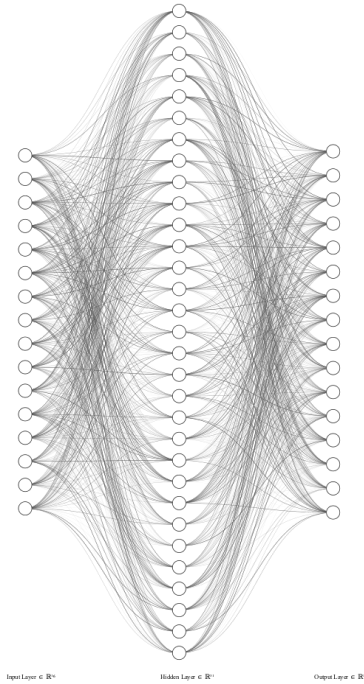

Figure 2.8: Sparse autoencoders must learn to represent their inputs as a sparse combination of features from a set of available features exponentially larger than their input dimension.

2.4.1 Tensions

Although sparse autoencoders have recently achieved remarkable results [25, 31, 33–40], allowing glimpses into what many previously assumed should perhaps be evaluated as only something of an inscrutable black box [41], they still leave much to be desired [42]:
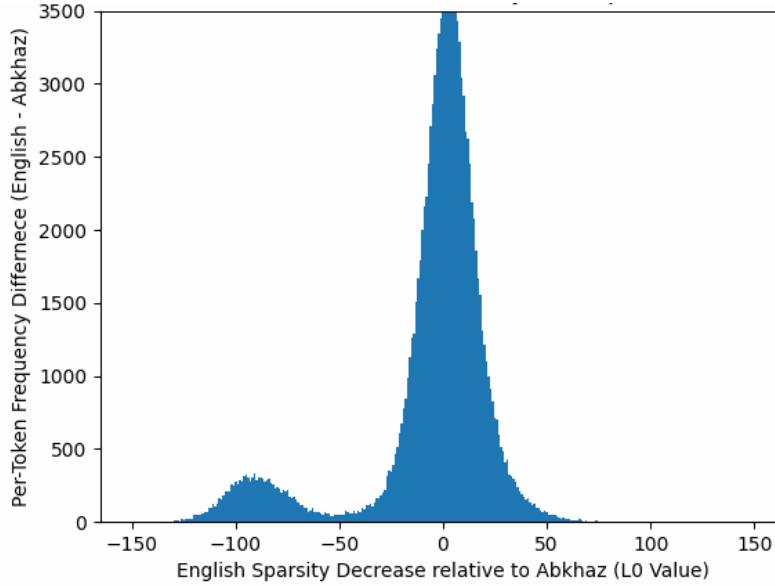
Figure 2.9: Decrease in the number of concept directions (L_0) in SAE encodings of activations from the English-language versions versus the number of concept directions in the same SAE's encodings of activations from the Abkhaz-language versions of the same Wikipedia entries. Each blue pixel represents the difference in the number of times one concept was activated by a single token in either language on a pre-trained SAE [32]. While most concepts tend to fire a similar number of times across both versions of a Wikipedia page, and the principal component of the distribution perhaps has a bias indicating higher concept for firing on Abkhaz versions, there is a highly significant number of concepts in the secondary component that often don't fire at all when processing LLM representations of Abkhaz text that do fire when processing LLM representations of the same text in English. It appears that in GPT-2, the LLM from which these activations were extracted, these concepts only fire once per each of the ≈ 90 times that they fire on the equivalent English text.

1. **Uniform sparsity**: Standard SAEs are encouraged to learn the same number of features for every single activation, regardless of the complexity of its true features [43]. This is an issue whenever we expect certain parts of LLM processing to be more complex than others. For example, SAEs often perform poorly on activations from languages that are poorly represented in the LLM's training dataset. As seen in Figure 2.9, when LLMs are trained more

on text from one language than from another, their activations demonstrate richer and more fine-grained internal representations when operating over the first language than when examining how they respond to the second. This "poverty of stimulus" in the training data is a serious issue, creating bias for worse performance in interpreting behavior in underrepresented languages. This bias is also an issue when we become interested in using SAEs to decouple meaning from these LLM-internal activations, since the SAE is likewise more incentivized to learn to represent the abundance of features that are used in relation to one language while sacrificing descriptive power of the other.

2. **$L_0 \approx L_1$ approximation**: In an ideal world, we would directly optimize sparsity by the L_0 norm (the count of non-zero concepts), but since this operation is not differentiable, we typically use the L_1 norm as a differentiable proxy. This approximation can be imperfect, leading to suboptimal sparsity patterns where many features are nearly but not exactly zero.

3. **Dead latents**: These are directions in the concept space that never activate significantly for any input, wasting representational capacity in an SAE's latent space. They emerge when the optimization process fails to assign meaningful roles to certain features, sometimes as the result of over-sparsification. Variants like BatchTopK and the auxiliary loss terms explained later address this by ensuring more uniform utilization of the concept space.

4. **Feature splitting**: A single coherent concept may be artificially split across multiple concept space dimensions, making interpretation difficult [44, 45]. Somehow allowing features to partially activate and collaborate more flexibly, such as by providing many distorted examples from which a mean can emerge, allows the model to more consistently learn concepts aligning with our expectations of them.

5. **Feature absorption**: The converse problem exists, where a single dictionary element absorbs multiple distinct concepts [44, 45]. This creates entangled representations where one feature might simultaneously encode, for instance, both "redness" and "roundness" after being exposed to too many apples, rather than separating these features. Models specifically addressing this issue enforce cleaner feature boundaries through additional regularization terms.

6. **Different modalities**: SAEs trained on activations from different models,

languages, or positions in the network often develop incompatible feature dictionaries, making feature quality poor and cross-comparison difficult. Features corresponding to punctuation or to head- or layer-specific tasks are not the types of concepts that we're interested in detangling [46]. My approache address this by encouraging convergent representations across these boundaries, just as how translation dictionaries allow people to find corresponding concepts across different languages.

2.4.2 Resolutions

Many improvements have already been made to address these issues [47–49], most of which involve altering the SAE's sparsity constraints. Let's revisit some past variants of autoencoders in general, and then discuss a few developments aimed specifically at improving the efficacy of sparse autoencoders at LLM dictionary learning:

- **TopK** [38]: This was the one of the original sparse activation functions for SAEs that enforces sparsity by retaining only the k highest activations in each feature vector while zeroing out all others. It imposes a hard constraint on the number of active features that any one activation can map to, creating a clean separation between activated and dormant features. However, this approach can be overly-rigid, as it fails to account for variations in the natural sparsity level across different inputs and instead imposes uniform sparsity across all activations regardless of context.

- **BatchTopK** [50]: This is an improvement over standard TopK that enforces sparsity at the batch level rather than individually for each input. By considering the top $k \cdot b$ activations across an entire batch of b inputs simultaneously, this approach allocates more features to inputs that genuinely require greater representational capacity while maintaining the same overall sparsity budget. The method has been shown to significantly reduce feature splitting (i.e. reduce the occurrences of single concepts being represented along multiple dimensions) and improve overall reconstruction fidelity.

- **Gated Sparse Autoencoders** [51]: These incorporate a gating mechanism that provides a more nuanced approach to feature selection than the binary on-off decisions of TopK methods. Each feature is acted on by a learned gate that can partially activate features, allowing the model to express degrees of confidence in a feature. This addition addresses the problem of feature

splitting as well, where a single coherent feature is artificially represented across multiple dictionary elements due to the hard constraints of TopK. For example, if the an activation contains the concept "zebra," but the SAE has already learned the concepts "has stripes" and "horse-like animal," then it would be unlikely to generalize to a third category. A gated approach allows for more integrated concept representations.

- **JumpReLU** [52]: This variant modifies the standard ReLU activation function by introducing a "jump" threshold. Rather than activating linearly from zero, features only begin activating after exceeding a learned threshold, creating a deliberate gap that discourages weak, spurious activations. This architectural change, actually inspired by research into conditioned synaptic behaviors in brains, helps distinguish genuinely meaningful signals from background noise.

- **Universal** [53]: This approach, first seen in pre-print this February, focuses on learning representations that transfer across different models or data modalities. Rather than training separate dictionaries for each domain, Universal SAEs identify common underlying features that remain invariant across shifts in the input distribution. While the cited work applied this technique to images across different vision models, I show that this principle extends to language representations across different languages under the *same* language model. This approach resembles how linguists at times attempt to identify universal principles that exist across all human languages despite surface differences. This prior work, in which a Universal SAE converged the concepts encoding identical images as they are represented within different image models, provides support for the methodological approach of training SAEs towards convergence. Since this approach did not actually learn convergent features between different material representations of a shared underlying human-interpretable concept (i.e. translations in natural language), but rather encouraged convergent interpretations of different machine-learned representations of identical photos, it is an open question whether this strategy can be applied to the present task. While it is highly interesting that unrelated image model architectures appear to share structural similarities in how they parse identical photos, it is unclear whether single language models share structural similarities in how it parses unique translations of the same underlying text. If they did, this could be exploited to derive more meaningful explanations of LLM behavior.

Building on these architectures, my hypothesis concerns the behavior of feature

activations across languages. For an activation x in the LLM at a specified layer, the strength of a feature f_i from that activation, $f_i(x)$, can be quantified as the activation of hidden neuron $a_i^{(\text{hid})}$ in the SAE:

$$f_i(x) = a_i^{(\text{hid})}(x) = \text{ReLU}(W_i^{\text{enc}} \cdot x + b_i^{\text{enc}}),$$

where W^{enc} and b^{enc} are the trained parameters of the SAE's encoder network. The central claim of this thesis is that SAE interpretability improves as the representations of translations of the same text become more similar to one another. Specifically, interpretability improves when:

$$\mathbb{E}_{\sim \mathcal{D}} \left[\sum_{i \in [x^{n_{\text{enc}}}]} \sum_{a,b \in X} \frac{f_i(t_a)}{f_i(t_b)|X|x^{l_{\text{enc}}}} \right] \to 1.$$

The above quantifies the convergence of feature activations across translations, with values approaching 1 indicating greater consistency in how semantic content is represented across languages.

To empirically investigate this, I examine token activations from translations in the EurLex dataset [54, 55], which is a comprehensive repository of EU legislation. This dataset is particularly valuable to me because it contains legal texts paired with high-quality human translations across many European languages, which provides a controlled environment to study how the same semantic content is represented across different linguistic forms, since these translations are required to be highly consistent with one another, as they are all equally legally binding. By analyzing how LLM activations respond to these parallel texts, we can begin to understand the emergence of language-independent concept spaces within Universal Sparse Autoencoders.
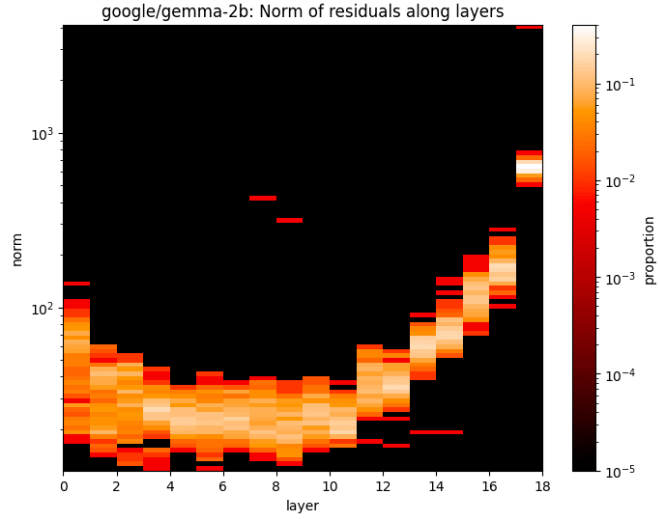
Methods

Figure 3.10: The norm distribution of residual activation states in different layers of the residual stream of Gemma-2B, the specific LLM investigated in this thesis, during inference. Recent work suggests that the most semantically abstracted representations are found in the central layers of LLMs, with earlier and later layers containing more language-specific features. This thesis considers activations found at layer 9 of this model [49].

3.1 Constructs

AY you are given a text $t \in T \in \mathcal{D}_t = \{T_i\}_{i=1}$ from a dataset of texts translated into various languages $\mathcal{D}_t$, with each T_i the concatenated translation of that text into each of l languages $T = \{t_1, t_2, \ldots, t_l\}$.

Let $\mathcal{D} = \{X_i\}_{i=1}^{|\mathcal{D}_t|}$ be the collection of latent signals sent at a specific layer of a sufficiently well-trained LLM upon processing the final token of each text, with X_i the signal sent upon processing T_i. In my case, this LLM was Gemma-2B, and activations were collected at the output residual of the 9th layer upon processing texts from the EurLex Dataset. I used this setup to train the first universal sparse autoencoder for language model activations:

3.2   Universal Sparse Autoencoder

UniversalSAE (USAE) is an approximation of some function $\theta^* \in \Theta$. This model approximates specifically over the subset of Θ that defines everywhere-differentiable functions of the form

$$\text{dec} \circ \text{enc} : \mathbb{R}^{d \times l} \to \mathbb{R}^{s^d} \to \mathbb{R}^{d \times l},$$

where $s \in \mathbb{Z}_{>1}$ is our superposition factor, $d \in \mathbb{Z}_+$ is the dimension of our LLM's activation space, $l \in \mathbb{Z}_+$ is the number of languages from which we have activations in $\mathcal{D}$, and $\mathbb{R}^{s^d}$ is an approximation of the LLM's *concept space.*

Recalling from earlier that θ^* is a parameterized function with codomain $\mathbb{R}$, we need to define a loss function $\mathcal{L}_{\text{USAE}}$ on the parameters of $\text{enc} \circ \text{dec}$ that scores these parameters' performance, and completes the mapping of $\text{dec} \circ \text{enc}$ onto $\mathbb{R}$. So, our ideal loss function, which scores different functions $\text{enc} \circ \text{dec}$, is some function of type

$$\mathcal{L}_{\theta^*} : \mathbb{R}^{d \times l \times \mathbb{R}^{s^d} \times \mathbb{R}^{d \times l}} \to \mathbb{R}.$$

We will define this function in more detail once we've had the chance to think a bit more about what we want it to accomplish. The first thing we need to do is constrain its infinite domain. To this end, we define the subset $\Theta' \subsetneq \Theta$ over which we're approximating θ^* as the parameterization of the **w**eights and **b**iases that define the functions dec and enc. If we let

$$\mathbf{W}_{\text{enc}} \in \mathbb{R}^{d \times s^d}, \mathbf{W}_{\text{dec}} \in \mathbb{R}^{d \times s^d \times l} \text{ and } \mathbf{b}_{\text{enc}} \in \mathbb{R}^{s^d}, \mathbf{b}_{\text{dec}} \in \mathbb{R}^{d \times l}$$

be the weights and biases that parameterize each respectively, then we may let each tuple of the form

$$\theta = (\mathbf{W}_{\text{enc}}, \mathbf{W}_{\text{dec}}, \mathbf{b}_{\text{enc}}, \mathbf{b}_{\text{dec}})$$

uniquely define each member of the parameterized family of functions over which

we're searching:

$$\Theta' = \{\theta : \theta = \mathcal{L}_{\theta^*}(\text{dec} \circ \text{enc}) : \text{dec} \circ \text{enc}(X) \mapsto \text{USAE}(X, \theta)\},$$

where

$$\text{USAE}(X, \theta) \overset{\text{def}}{\mapsto} \text{BTK}(\text{RELU}(\mathbf{W}_{\text{enc}}^{\langle l \rangle} X + \mathbf{b}_{\text{enc}}^{\langle l \rangle})) \mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}}.$$

Here, RELU is the REctified Linear Unit function, which simply sets all negative values in a vector to 0. The function BTK, i.e. BATCHTOPK, is also a non-linear activation function, specifically the BATCH-wise version of the TOPK function. These are discussed in the background section.

$$\forall n \in \mathbb{Z}_+ : \forall x \in \mathbb{R}^n : \forall i \in [\![1, n]\!] : \text{RELU} : \mathbb{R}^n \to \mathbb{R}^n : \text{RELU}(x)_i \mapsto \max(x_i, 0).$$

Returning to our mystery function, we've decided it must be composed of the form

$$\text{enc}(X) = \text{BTK}(\text{RELU}(\mathbf{W}_{\text{enc}}^{\langle l \rangle} X + \mathbf{b}_{\text{enc}}^{\langle l \rangle})),$$

which we will call X's *encoding*. Likewise,

$$\text{dec} \circ \text{enc}(X) = \text{enc}(X) \mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}},$$

will be called X's *reconstruction*. The superscript $\langle l \rangle$ means that we are duplicating and concatenating the exact same matrix / vector l times with itself, so that it may be of the correct length, with one *identical* copy of the same encoder existing for each of l languages. This can been seen equivalently as concatenating each language's activation's encoding under one and the same the same *universal* encoder, and then decoding each of these concatenated encodings using language-specific decoders. To be explicit, if we allow $||$ to be the concatenating operator and $X, \mathbf{W}_{\text{dec}}, \mathbf{b}_{\text{dec}}$ tensors, then

$$\text{USAE}(X, \theta)$$

$$= \text{BTK}(\text{RELU}(\mathbf{W}_{\text{enc}}^{\langle l \rangle} X + \mathbf{b}_{\text{enc}}^{\langle l \rangle})) \mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}}$$

$$= \bigg|\bigg|_{i=1}^{l} \left[\bigg|\bigg|_{j=1}^{l} [\text{BTK}(\text{RELU}(\mathbf{W}_{\text{enc}} X_{[id:(i+1)d,]} + \mathbf{b}_{\text{enc}}))] \mathbf{W}_{\text{dec}[jd:(j+1)d,]} + \mathbf{b}_{\text{dec}[jd:(j+1)d,]} \right].$$

With this in mind, we can also refer to

$$\text{USAE}_{i,j}(X, \theta)$$

$$\overset{\text{def}}{=} \text{BTK}(\text{RELU}(\mathbf{W}_{\text{enc}}X_{[id:(i+1)d,]} + \mathbf{b}_{\text{enc}}))\mathbf{W}_{\text{dec}[jd:(j+1)d,]} + \mathbf{b}_{\text{dec}[jd:(j+1)d,]}$$

as the "reconstruction of language i decoded into language j," or a bit more specifically, as "the encoding of the activation from the translation of text X into language i reconstructed into the activation from the translation of text X into language j." We can also call the following "the encoding of language i" and "the decoding of the encoding c under language j's decoder:"

$$\text{enc}_i(X) \overset{\text{def}}{=} \text{BTK}(\text{RELU}(\mathbf{W}_{\text{enc}}X_{[id:(i+1)d,]}))$$

$$\text{dec}_j(c) \overset{\text{def}}{=} c\mathbf{W}_{\text{dec}[jd:(j+1)d,]} + \mathbf{b}_{\text{dec}[jd:(j+1)d,]}.$$

Finally, I denote the following as the specific activation given from text $X \in \mathcal{D}$ in language $i \in [\![1, l]\!]$:

$$x_i \overset{\text{def}}{=} X_{[id:(i+1)d,]},$$

and so we can write either of the following, which refer to the same encoding:

$$\text{enc}_i(X) = \text{enc}(x_i).$$

We now have a concrete definition of the parameters of our model that we are optimizing, namely $\mathbf{W}_{\text{enc}}, \mathbf{b}_{\text{enc}}, \mathbf{W}_{\text{dec}},$ and $\mathbf{b}_{\text{dec}}$. The crucial missing piece is a construction of our loss function, which takes our model to $\mathbb{R}$. We want that for any LLM activation $\text{enc}_i^{-1}(c) = x_i \in \mathbb{R}^d$, that $\text{enc}_i(X) = c$ is a sparse vector corresponding to X's meaning *for all languages in our dataset $i \in [\![1, l]\!]$ simultaneously.* This means that we want $\text{enc}_i(X)$ to be similar to $\text{enc}_j(X)$ for all languages i and j. We also want each direction along which $\text{enc}_i(X)$ points to correspond to a specific *concept*, and that each concept's "intensity" (e.g. its strength, salience, relevance, or importance in a given context) be proportional to $\text{enc}_i(X) = c$'s magnitude along that concept's dimension. For example, if X_1 is a law regarding the necessary procedure that must be carried out when reporting alleged petty tax fraud in Belgium in 1993, and X_2 is one of the universal declarations of human rights, then we should expect that $\text{enc}_i(X_1) = c_1$ point in more distinct directions than $\text{enc}_i(X_2) = c_2$. Here, "sparse" simply means that $\text{enc}_i(X)$ is 0 along most dimensions, under the assumption that *most* concepts are not associated with a given LLM activation. For example, we would expect c_1 to be sparser than c_2, but would still expect that c_2 be zero along *most* dimensions. We would also like that the intersection of these sparse non-zero dimensions be in negligible proportion to the number of dimensions that each inhabits individually, since these two laws do not share strong semantic relationships with one another.

3.3 $\mathcal{L}_{\textbf{USAE}}$

We're now ready to specify one of the defining characteristics of the Universal Sparse Autoencoder, which is its loss function.

3.3.1 Definition

Finding functions that satisfy everything we've formally constrained so far is trivial, so long as these functions are well-typed. We haven't yet placed any real constraints regarding what we want this function's behavior to actually look like. Recall that we've chosen to approximate a function from this specific class of functions Θ' so that we may customize our construction of $\mathcal{L}_{USAE}$ to reflect the degree to which our specific goals of convergent and accurate representations have been accomplished. In order to ensure that minimizing $\mathcal{L}_{USAE}$ correspond to finding a θ that maximizes our objectives, I define our loss function as

$$\mathcal{L}_{USAE}(X,\theta) = \sum_{i=1}^{l} \frac{\lambda_i}{l} \|x_i - \text{dec}_i \circ \ \text{enc}(x_i)\|_2^2 \tag{3.1}$$

$$+ \sum_{i=1}^{l} \frac{\hat{\lambda}_i}{l} \|\text{enc}(x_i)\| \tag{3.2}$$

$$+ \sum_{i=1}^{l} \frac{\lambda_{\text{aux}}}{l} \|x_i - \text{dec}_i(\textsc{AuxTopK}(\textsc{ReLU}(\mathbf{W}_{\text{enc}}x_i)))\|_2^2 \tag{3.3}$$

$$+ \sum_{i,j \in [\![1,l]\!]} \lambda_c \cdot \left(1 - \frac{\text{enc}\,(x_i) \cdot \text{enc}\,(x_j)}{\|\text{enc}(x_i)\| \times \|\text{enc}(x_j)\|} \right) \tag{3.4}$$

$$+ \sum_{i,j \in [\![1,l]\!]} \lambda' \|x_i - \text{dec}_i \circ \text{enc}(x_j)\|_2^2 \tag{3.5}$$

3.3.2 Justification

These summands correspond in a literal sense to these goals, with each λ representing a hyperparameter that can be adjusted to weigh each of these goals induvidually. The first three terms are adapted directly to the multilingual case (where $l > 1$) from previous work [38, 50], while the final two are my thesis' contributions. Let's go over each:

(3.1) This is the mean squared error between the network's input, x_i, with the

network's output, $\text{dec}_i \circ \text{enc}(x_i)$. This measures how well the input is reconstructed after it passes through the network. We take a weighted mean λ_i of this value for all translations of $x_i \in X$.

(3.2) This, in addition to our BATCHTOPK activation function, is our second sparsity term. The L_1 norm of x_i's encoding corresponds to the absolute magnitude of the concepts that x_i activates. We also take a weighted mean $\hat{\lambda}_i$ of this value for each $x_i \in X$. We would ideally like to minimize for $\|\text{enc}(x_i)\|_0$, the total number of non-zero concepts in $\text{enc}(x_i)$, rather than $\|\text{enc}(x_i)\|$, the total magnitude of these concepts. Unfortunately this discrete count is not differentiable, and so in order to integrate it into our deep learning framework, it must be approximated by the L_1 term. Under nice conditions, however, we know that minimizing either minimizes the other, and this method of enforcing sparsity works empirically. The crucial change I've made to these functions from their original implementations is that each hyperparameter, indexed by i, can be made *language-specific* to account for biases in the LLM's training to be able to more accurately "think" in one language than in another.

(3.3) In order for the training process to encourage the USAE to utilize as much of its latent space as possible, and so that concepts are more uniformly distributed throughout this space, we want to discourage *dead features*. A dead feature is simply a dimension of the latent space that is being underutilized. In my implementation I define this as a dimension in which an activation has not registered for more than five batches. This AUXiliary loss term encourages these dimensions to be used by adding the reconstruction loss between the input x_i, and what its reconstruction would have been if, instead of decoding the top k activations in the batch, we instead decoded the $(k - k_{\text{aux}} - 1)$th through $(k - 1)$th highest activating neurons to try to reconstruct each x_i in the batch. In essence, instead of motivating the network to concentrate its representation in a dense part of the activation space, we want that the model encode all features so that only those that are most strong may be passed to the decoder network. So to do this, instead of evaluating the reconstruction loss based on the k strongest features, we evaluate the k_{aux} *next* strongest features. Here, k_{aux} is another hyperparameter that can be tuned. The authors of the original implementation of this strategy in the monolingual case note that the best values seem to be whichever power of 2 is nearest $\frac{d}{2}$, but that training is not especially sensitive to this choice [50]. Most implementations of this loss term I've seen

arbitrarily set k_{aux} to 512, regardless of d. By rewarding the network for a larger section of the latent space's reconstruction accuracy, but discounting this loss by λ_{aux} (usually $\frac{1}{32}$, but the authors again note that training is not especially sensitive to this hyperparameter), we're able to encourage the model to make more optimal use of its latent space while maintaining sparsity. We again take the mean of this value over all languages, but I do not implement language-specific weighting since this would counteract the goal of developing similar activations across languages. The final two terms, for this reason, also lack language-specific hyperparameters, whereas the previous two were indexed by some $i \in [\![1, l]\!]$.

(3.4) This cosine similarity term encourages the network to point activations from different translations of the same text in similar directions. This is because the cosine similarity of two vectors monotonically increases as their norms align, and 1 minus this value converges to 0 *iff* the two vectors align completely. The hyperparameter λ_c must be kept extremely small but non-zero for successful training, as discussed in the next section.

(3.5) In order to encourage the model to develop language-independent representations, we optimize the network's ability use *any* language i's decoder to decode the encodings of activations from *any* source language j. We take the mean squared error of the network's reconstruction of the never-seen activation with the activation itself. My data demonstrate that successful training requires that λ' be *many* orders of magnitude larger than $l \cdot \sum_{i=1}^{l}[\lambda_i]$, meaning that successful concept convergence relies on a much higher emphasis on the network's ability to sparsely *translate* activations rather than on its ability to sparsely *reconstruct* activations.

Results

I was able to successfully implement and evaluate instances of this architecture, and in doing so, measured the quality of the concept spaces that they approximate. First, these are the hyperparameter values that I found to work well for training, as well as the terms they depend on:

Hyperparameter	Value	Notes
α	2^{-18}	This is our learning rate.
s	as large as possible	A larger concept dictionary (with pro-
k	$2^4 \cdot s^d$	portionally scaled sparsity constraints
		as well) has been shown to always yield
		more interpretable results with SAEs,
		but is limited in practice by restraints
		on computational memory. My
		implementation uses $s = 2^{\frac{1}{128}}$ with
		$s^d = 2^5 \cdot d$.
λ_i	$2^0 \cdot l_i^{-1}$	The value l_i is equal to the proportion
$\hat{\lambda}_i$	$2^{-5} \cdot l_i^{-1}$	of training data that went into training
$\lambda_{\mathbf{aux}}$	2^{-4}	the LLM in language i relative to all
$k_{\mathbf{aux}}$	$2^{-1} \cdot s^d$	other languages. If the LLM was
λ_c	2^{-10}	trained on all languages equally, then
λ'	$2^7 \cdot \lambda_i$	$l_i^{-1} = l^{-1}$. If more training data was in
		language i than in language j, then
		$l_j^{-1} > l_i^{-1}$. Regardless, $\sum_{i=1}^{l} \lambda_i \cdot l^{-1} = 1$.

The cosine similarity hyperparameter, λ_c, must be kept extremely small for successful training, as anything larger causes the network to point nearly all inputs in the same direction, even those that are not translations of the same text, resulting in an extremely high percentage of dead features. There seems to be
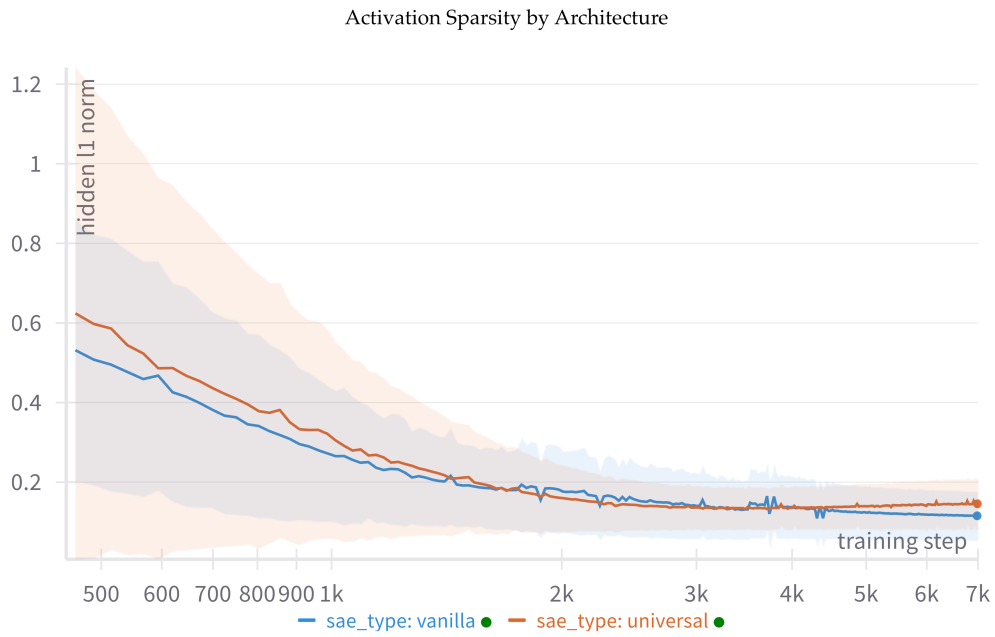
Activation Sparsity by Architecture

Figure 4.11: Sparsity remains consistent across vanilla (blue) and Universal (orange) sparse autoencoders, although the Universal variant transitions from a phase of high to low sparsity variance.
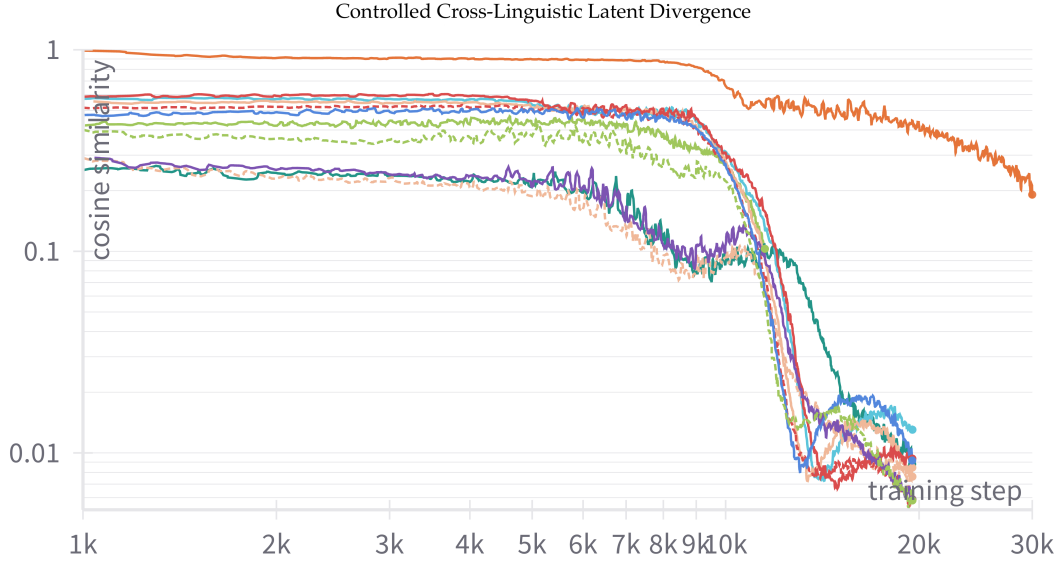
Figure 4.12: Plot of batch-wise test semantic divergence (i.e. negative convergence) across various training runs. Test semantic convergence was measured by dividing the mean cosine similarity of pairs of encodings of related texts' (translations) activations and dividing this value by the mean cosine similarity of randomly-sampled, likely unrelated pairs of activations from the same languages. Semantic convergence in a USAE's concept space approximation appears to occur suddenly. This supports the hypothesis that a training signal that encourages language-agnostic behavior allows sparse autoencoders to rapidly collapse encodings without harming reconstruction fidelity by encoding semantic content rather than form. Cosine similarity increases in a similar fashion when $\lambda_c = 0$ and the network is not encouraged explicitly to converge (orange), but achieves convergence at a significantly slower rate. This is because the network is still encouraged to converge indirectly by the translation loss term (3.5), but not directly by the cosine similarity loss term. In as many training steps, BatchTopK SAEs (not shown) do not have significant drops in cross-linguistic representation cosine similarity, and remain divergent.

an inversely proportional relationship between the optimal λ_c and the consistency of the optimal λ_i for different values of $i \in [\![1, l]\!]$. That is, the more our dataset $\mathcal{D}$ is biased towards certain languages over others, the less we should encourage the network to point translations in the same direction. This is likely because in the biased setting, the network learns to point translations paired with an over-represented language in the direction of that language's translation, rather than in a direction truly representing the shared semantic content of the two translations. Conversely, when $\lambda_c = 0$ and we remove this term from our loss function entirely, I observe that measured cosine similarity (i.e. the term to the right of the fraction in (2.6)) still decreases in the same way, but scaled by an exponential factor. For example, in Figure 4.12, where the x-axis is log-transformed, the loss graph when $\lambda_c = 0$ (in orange) appears asymptotically equivalent to when $\lambda_c > 0$ (the rest). This suggests convergence is being approached in both cases, but that a weak signal to encourage otherwise random noise in the loss landscape to tend towards a shared space vastly aids the rapid convergence of cross-linguistic concepts. We can evaluate this empirically by examining the maximally-activating elements in the neighborhood of a direction in the encoding spaces of both traditional and Universal SAEs. To compose the following, I trained both types of SAEs to have virtually equivalent sparsities and reconstruction accuracies, chose a random law from $\mathcal{D}$, and sampled the direction in the encoding space activated by this law's final token. I then collected the top-3 nearest neighbors out of the entire dataset for both, presented in Table 4.2. While the vanilla SAE grouped law 0 with other English translations of laws that in some sense relate to financial regulation, the Universal SAE successfully grouped it with both its Spanish counterpart, and with one other law (and its translation) concerning granting approval for the same specific program's enactment.

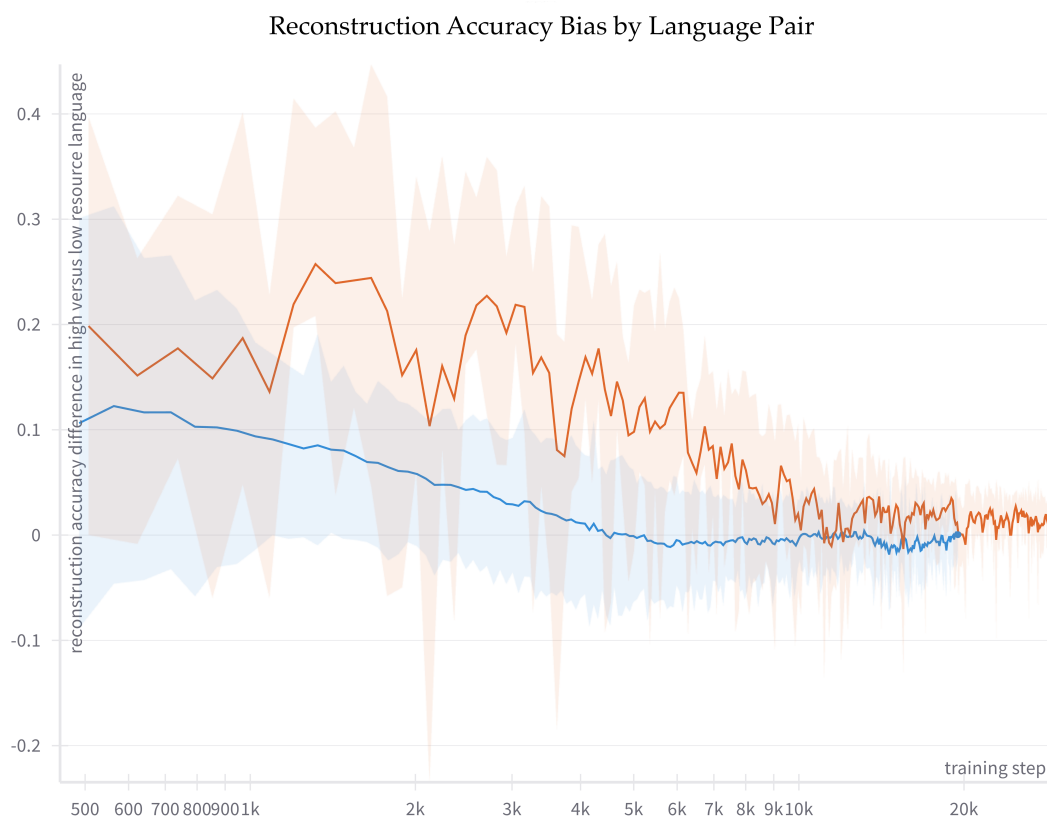Reconstruction Accuracy Bias by Language Pair

Figure 4.13: Differences in the reconstruction accuracies between two languages over time for 1) a Universal Sparse Autoencoder training on English and Spanish texts (blue) and 2) a Universal Sparse Autoencoder training on English and Latvian texts (orange). When training a typical sparse autoencoder, there is no way to counteract the biases embedded in the training data for one language over another, and these values always converge at a value significantly greater or less than 0 (not shown). However, with Universal Sparse Autoencoders, even in the case where one language is much more represented in the LLM's training data than the other (orange), the difference in reconstruction accuracy between the two languages eventually tends towards 0. Although this convergence happens much slower when there is high disparity between the two languages in the training data (in this case taking about twice as long), it's nonetheless a novel property of this species of architecture.

Rank	Vanilla
0	COMMISSION DECISION of 7 December 1993 concerning the grant of assistance from the cohesion financia...
1	COMMISSION REGULATION (EC) No 1754/1999 of 6 August 1999 fixing the minimum selling prices for beef...
2	COMMISSION REGULATION (EEC) No 1876/93 of 12 July 1993 re-establishing the levying of customs duties...
3	COUNCIL DECISION of 4 December 2007 concerning the conclusion of the Agreement in the form of an Exc...

	Universal
0	COMMISSION DECISION of 7 December 1993 concerning the grant of assistance from the cohesion financia...
1	DECISIÓN DE LA COMISIÓN de 7 de diciembre de 1993 sobre la concesión de una ayuda del instrumento fi...
2	COMMISSION DECISION of 21 December 1988 on a specific programme for the provision of facilitie...
3	DECISIÓN DE LA COMISIÓN de 21 de diciembre de 1988 sobre el programa específico relativo al eq...

Table 4.2: First lines of the of laws whose activations are most strongly conceptually associated with the first (law 0) under two types of sparse autoencoders.

Discussion

HESE results provide evidence for the existence of language-independent concept spaces within large language models, and for the benefits that can be drawn in approximating them. Through applying Universal Sparse Autoencoders to the domain of language, I've demonstrated that LLM activations from different languages, when processing semantically equivalent texts, are mapped to similar regions in a high-dimensional concept space that preserve the essential semantic content while discarding language-specific syntactic and morphological peculiarities.

Insights emerge. The rapid convergence of feature representations across languages suggests that the concept space is not an artifact of our training procedure but truly reflects structure within the LLM's internal representations. The fact that semantic convergence occurs suddenly suggests a phase transition in the optimization landscape at which point the network discovers that aligning concepts across languages is the most efficient solution to its constrained optimization problem.

It appears that the concept space is organized according to meaning rather than form, with semantically related concepts clustering together regardless of the surface realization of these concepts across languages. Looking ahead, I plan to continue this line of inquiry. One immediate extension I'm considering will be to scale the number of languages further to investigate whether the concept space becomes more refined or more abstract as the diversity of linguistic inputs increases. Preliminary experiments with up to at most four languages suggest that adding more languages enhances the quality of the concept space in certain contexts, but computational constraints prevented a more thorough investigation. It seemed empirically that a language's popularity among the LLM's training data played a large factor in this.

Another interesting direction would be to examine how concept representations evolve across different layers of the network. While this study focused on layer 9 of Gemma-2B, examining the trajectory of concept formation from early to late layers could reveal how abstract semantic representations emerge from more concrete linguistic features. Recent work has shown that composing SAE features into feature "circuits" within LLMs is a very useful way of interpreting and modifying LLM behavior [33]. Universal Sparse Autoencoders could help trace these circuits with better precision to accomplish more interpretable symbolic behaviors.

The techniques developed in this thesis could be used to analyze how concepts are composed and manipulated in cross-linguistic settings. By examining how combinations of concepts interact across different languages, we might also gain insight into whether the concept space bears any relation to formal linguistic theories of compositional semantics. These results might one day contribute to or inspire more effective methods for cross-lingual transfer machine learning, machine translation, or multilingual retrieval-augmented generation, have been identified as important directions for future interpretability research [56]. The Universal Sparse Autoencoder might map content from different languages to a shared semantic space where equivalence, accuracy, and similarity can be more accurately assessed.

Beyond just feature-circuits, virtually all recent advances in LLM interpretability have been founded on sparse encodings of LLM latents. Future interpretability work, such as research into feature circuits, must explain how discrete concepts interact with one another inside machine-learned models. This is why, before we can describe how they interact, fundamentally establishing highly accurate and concise baseline concepts in the first place is a crucial step at the onset of our quest for interpretability.

In conclusion, this thesis demonstrates that large language models are capable of developing rich, language-independent *concept spaces* that capture semantic content in a way that seems to transcend specific linguistic surface realizations. We can begin to understand how meaning is represented and manipulated within increasingly ubiquitous systems of generative AI, and may take steps to balance the biases they entail, by understanding these spaces. As these models continue to advance and their applications expand, understanding their internal representations is both an academic curiosity and a practical necessity. The methods and findings presented are a step in this direction.

Appendix

Code

All code written for this thesis can be found here:
github.com/gggggv15/UniversalSAE

Cross-Linguistic Training Analysis

Here I demonstrate that a simple language model trained on 2 languages at once outperforms an otherwise equivalent model trained on one language and then the other sequentially. This could be evidence that the first model learns to encode the semantic representation of tokens faster than the second because it was more easily able to distinguish between language-specific and underlying shared semantic content of the training data.
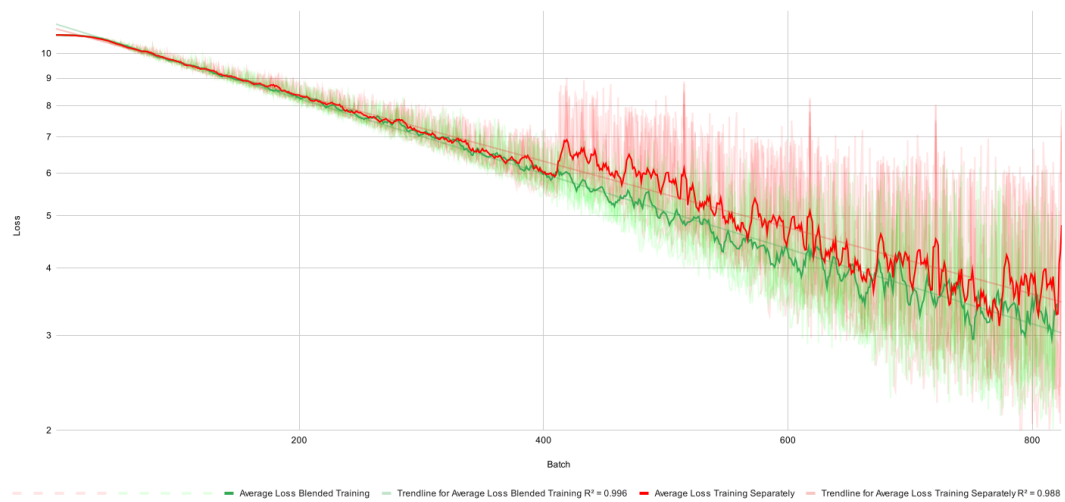
Cited Works

[1] Plato, "Theaetetus," in *Plato, Plato in Twelve Volumes, Vol. 12*, H. N. Fowler, Ed. Cambridge, MA: Harvard University Press, 1921, p. 210c, socrates: "τοςοῦτον γὰρ μόνον ἡ ἐμὴ τέχνη δύναται".

[2] A. Denes, "View with new york's financial center," https://rarehistoricalphotos.com/wheatfield-manhattan-1982/, 1982, image from Wheatfield – A Confrontation, photographed by same.

[3] T. Maillard, "Magnetic core memory card," https://en.wikipedia.org/wiki/File:Magnetic_core_memory_card.jpg, 2007, magnetic core memory extracted from a CDC 6600 computer (1961), 10.8 cm × 10.8 cm, capacity: 1024 bits. Licensed under CC BY-SA 3.0 and under GNU Free Documentation License.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016. [Online]. Available: https://arxiv.org/abs/1409.0473

[6] D. Li and L. Agha, "Big names or big ideas: Do peer-review panels select the best science proposals?" *Science*, vol. 348, no. 6233, pp. 434–438, 2015. [Online]. Available: 10.1126/science.aaa0185

[7] H. Graßmann, "Die wissenschaft der extensiven grösse oder die ausdehnungslehre, eine neue mathematische disciplin. bd. 1." Leipzig, 1844, german Text Archive. [Online]. Available: https://www.deutschestextarchiv.de/grassmann_ausdehnungslehre_1844

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. [Online]. Available: https://api.semanticscholar.org/CorpusID:205001834

[9] L. Riggs, "Really strong features found in residual stream," https://www.alignmentforum.org/posts/Q76CpqHeEMykKpFdB/really-strong-features-found-in-residual-stream, 2023, aI Alignment Forum post, published on 8 July 2023.

[10] W. Johnson and J. Lindenstrauss, "Extensions of lipschitz maps into a hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189–206, 01 1984.

[11] N. Nanda, "A comprehensive mechanistic interpretability explainer & glossary," 2023. [Online]. Available: https://dynalist.io/d/n2ZWtnoYHrU1s4vnFSAQ519J

[12] N. E. et al., "Toy models of superposition," 2022. [Online]. Available: https://arxiv.org/abs/2209.10652

[13] N. Nanda, "Neuroscope: A website for mechanistic interpretability of language models," 2023. [Online]. Available: https://neuroscope.io

[14] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.290.5500.2319

[15] B. Riemann, "Grundlagen für eine allgemeine theorie der functionen einer veränderlichen complexen grösse," Ph.D. dissertation, Universität Göttingen, 1851, in: Gesammelte mathematische Werke und wissenschaftlicher Nachlass.

[16] H. Whitney, "Differentiable manifolds," *Annals of Mathematics*, vol. 37, no. 3, pp. 645–680, 1936.

[17] E. H. Brown, "Some remarks on projective stiefel manifolds, immersions of projective spaces, and the whitney invariant," *Proceedings of the*

American Mathematical Society, vol. 80, no. 4, pp. 595–600, 1978. [Online]. Available: https://www.ams.org/journals/proc/1980-080-04/S0002-9939-1980-0587951-X/S0002-9939-1980-0587951-X.pdf

[18] J. M. Lee, *Introduction to Smooth Manifolds*, 2nd ed., ser. Graduate Texts in Mathematics. New York: Springer, 2013, vol. 218, regular Value / Level-Set (Pre-image) Theorem: see Theorem 5.14.

[19] L. W. Tu, "The regular level set theorem," in *An Introduction to Manifolds*, 2nd ed., ser. Universitext. New York: Springer, 2011, ch. 9.3, pp. 105–106.

[20] Pekkog, "Boundary of manifold with charts," https://commons.wikimedia.org/wiki/File:Boundary_of_Manifold_with_charts.png, 2024, depiction of a manifold with boundary with an interior chart and a boundary chart as well as its boundary again with an interior chart. Uploaded while editing "Manifold" on en.wikipedia.org. Licensed under CC A-SA 4.0 I.

[21] N. Whiteley, A. Gray, and P. Rubin-Delanchy, "Statistical exploration of the manifold hypothesis," 2024. [Online]. Available: https://arxiv.org/abs/2208.11665

[22] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0042698997001697

[23] D. E. Rumelhart, J. L. McClelland, and P. R. Group, *Parallel Distributed Processing, Volume 1: Explorations in the Microstructure of Cognition: Foundations.* The MIT Press, 07 1986. [Online]. Available: https://doi.org/10.7551/mitpress/5236.001.0001

[24] L. Evanson, C. Bulteau, M. Chipaux, G. Dorfmüller, S. Ferrand-Sorbets, E. Raffo, S. Rosenberg, P. Bourdillon, and J. King, "Emergence of language in the developing brain," https://ai.meta.com/research/publications/emergence-of-language-in-the-developing-brain/, May 2025, meta AI Research publication.

[25] J. Lindsey, W. Gurnee, E. Ameisen, B. Chen, A. Pearce, N. L. Turner, C. Citro, D. Abrahams, S. Carter, B. Hosmer, J. Marcus, M. Sklar, A. Templeton, T. Bricken, C. McDougall, H. Cunningham, T. Henighan, A. Jermyn, A. Jones, A. Persic, Z. Qi, T. B. Thompson, S. Zimmerman,

K. Rivoire, T. Conerly, C. Olah, and J. Batson, "On the biology of a large language model," https://transformer-circuits.pub/2025/attribution-graphs/biology.html, Mar. 2025, accessed 2025-05-05.

[26] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "In-context learning and induction heads," 2022. [Online]. Available: https://arxiv.org/abs/2209.11895

[27] E. Ameisen and J. Batson, "Circuits updates - september 2024," Sep 2024. [Online]. Available: https://transformer-circuits.pub/2024/september-update/index.html

[28] W. Gurnee, N. Nanda, M. Pauly, K. Harvey, D. Troitskii, and D. Bertsimas, "Finding neurons in a haystack: Case studies with sparse probing," 2023. [Online]. Available: https://arxiv.org/abs/2305.01610

[29] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426

[30] L. van der Maaten and G. Hinton, "Viualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 11 2008.

[31] T. Bricken and E. al., "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, Oct 2023. [Online]. Available: https://transformer-circuits.pub/2023/monosemantic-features/index.html

[32] J. Bloom, "Open source sparse autoencoders for all residual stream layers of gpt2-small," Feb 2024. [Online]. Available: https://www.lesswrong.com/posts/f9EgfLSurAiqRJySD/open-source-sparse-autoencoders-for-all-residual-stream

[33] S. Marks, C. Rager, E. J. Michaud, Y. Belinkov, D. Bau, and A. Mueller, "Sparse feature circuits: Discovering and editing interpretable causal graphs in language models," 2024. [Online]. Available: https://arxiv.org/abs/2403.19647

[34] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," 2023. [Online]. Available: https://arxiv.org/abs/2309.08600

[35] N. Nanda, "Open source replication & commentary on anthropic's dictionary learning paper," Oct 2023. [Online]. Available: https://www.lesswrong.com/posts/fKuugaxt2XLTkASkk/open-source-replication-and-commentary-on-anthropic-s

[36] L. M. I. Team, "Gemma scope: helping the safety community shed light on the inner workings of language models," Jul 2024. [Online]. Available: https://deepmind.google/discover/blog/gemma-scope-helping-the-safety-community-shed-light-on-the-inner-workings-of-language-models/

[37] J. Bloom and D. Chanin, "Evaluating saes with sae lens evals," https://github.com/jbloomAus/SAELens/blob/main/tutorials/evaluating_saes_with_sae_lens_evals.ipynb, 2024.

[38] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu, "Scaling and evaluating sparse autoencoders," 2024. [Online]. Available: https://arxiv.org/abs/2406.04093

[39] G. Paulo, A. Mallen, C. Juang, and N. Belrose, "Automatically interpreting millions of features in large language models," 2024. [Online]. Available: https://arxiv.org/abs/2410.13928

[40] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan, "Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet," *Transformer Circuits Thread*, 2024. [Online]. Available: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

[41] Y. Y. Elboher, J. Gottschlich, and G. Katz, "An abstraction-based framework for neural network verification," in *Computer Aided Verification*, S. K. Lahiri and C. Wang, Eds. Cham: Springer International Publishing, 2020, p. 43–65. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-53288-8_3

[42] A. Makelov, G. Lange, and N. Nanda, "Towards principled evaluations of sparse autoencoders for interpretability and control," 2024. [Online]. Available: https://arxiv.org/abs/2405.08366

[43] K. Ayonrinde, "Standard saes might be incoherent: A choosing problem & a "concise" solution," Oct 2024. [Online]. Available: https://www.lesswrong.com/posts/vNCAQLcJSzTgjPaWS/standard-saes-might-be-incoherent-a-choosing-problem-and-a

[44] K. Hänni, J. Mendel, and K. Kozaronek, "Decomposing activations into features: How many and how do we find them? — a survey," 2023, unpublished manuscript. Submitted May 23, 2023. Available at: https://kaarelh.github.io/ai/decomposition.pdf.

[45] D. Chanin, J. Wilken-Smith, T. Dulka, H. Bhatnagar, and J. Bloom, "A is for absorption: Studying feature splitting and absorption in sparse autoencoders," 2024. [Online]. Available: https://arxiv.org/abs/2409.14507

[46] D. Till, "Do sparse autoencoders find "true features"," 2024. [Online]. Available: https://www.lesswrong.com/posts/QoR8noAB3Mp2KBA4B/do-sparse-autoencoders-find-true-features

[47] J. Lindsey, A. Templeton, J. Marcus, T. Conerly, J. Batson, and C. Olah, "Sparse crosscoders for cross-layer features and model diffing," https://transformer-circuits.pub/2024/crosscoders/index.html, 2024, anthropic. Published on October 25, 2024.

[48] J. Olmo, J. Wilson, M. Forsey, B. Hepner, T. V. Howe, and D. Wingate, "Features that make a difference: Leveraging gradients for improved dictionary learning," 2024. [Online]. Available: https://arxiv.org/abs/2411.10397

[49] F. Hu. (2024, Apr.) Normalizing sparse autoencoders. LessWrong, posted 8 Apr 2024. [Online]. Available: https://www.lesswrong.com/posts/3ZCKSArYwgg9P4hqQ/normalizing-sparse-autoencoders

[50] B. Bussmann, P. Leask, and N. Nanda, "Batchtopk sparse autoencoders," 2024. [Online]. Available: https://arxiv.org/abs/2412.06410

[51] S. Rajamanoharan, A. Conmy, L. Smith, T. Lieberum, V. Varma, J. Kramár, R. Shah, and N. Nanda, "Improving dictionary learning with gated sparse autoencoders," 2024. [Online]. Available: https://arxiv.org/abs/2404.16014

[52] S. Rajamanoharan, T. Lieberum, N. Sonnerat, A. Conmy, V. Varma, J. Kramár, and N. Nanda, "Jumping ahead: Improving reconstruction

fidelity with jumprelu sparse autoencoders," 2024. [Online]. Available: https://arxiv.org/abs/2407.14435

[53] H. Thasarathan, J. Forsyth, T. Fel, M. Kowal, and K. Derpanis, "Universal sparse autoencoders: Interpretable cross-model concept alignment," 2025. [Online]. Available: https://arxiv.org/abs/2502.03714

[54] I. Chalkidis, M. Fergadiotis, and I. Androutsopoulos, "MultiEURLEX - a multi-lingual and multi-label legal document classification dataset for zero-shot cross-lingual transfer," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6974–6996. [Online]. Available: https://aclanthology.org/2021.emnlp-main.559/

[55] Q. L. et al., "Datasets: A community library for natural language processing," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 175–184. [Online]. Available: https://aclanthology.org/2021.emnlp-demo.21

[56] M. Wang, H. Adel, L. Lange, Y. Liu, E. Nie, J. Strötgen, and H. Schütze, "Lost in multilinguality: Dissecting cross-lingual factual inconsistency in transformer language models," 2025. [Online]. Available: https://arxiv.org/abs/2504.04264

[57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[58] M. W. Hirsch, *Differential Topology.* Springer-Verlag New York, 1976, 4.6.5. [Online]. Available: https://people.dm.unipi.it/benedett/HIRSCH.pdf

[59] A. F. Spies, "Sparse autoencoders," https://unsearch-ai.notion.site/Sparse-Autoencoders-30e04d55f6f64f66a7a1c97751bbacf5, 2023, notion document. Created on November 28, 2023, 6:20 PM; last edited on June 11, 2024, 4:34 PM.

[60] S. Marks and A. Mueller, "dictionary learning," https://github.com/saprmarks/dictionary_learning, 2024.

[61] L. Gao and J. Wu, "sparse autoencoder," https://github.com/openai/sparse_autoencoder, 2024.

[62] J. Bloom and D. Chanin, "pretrained_saes.yaml," https://github.com/jbloomAus/SAELens/blob/main/sae_lens/pretrained_saes.yaml, 2024.

[63] J. Bloom, C. Tigges, and D. Chanin, "Saelens," https://github.com/jbloomAus/SAELens, 2024.

[64] N. Nanda and J. Bloom, "Transformerlens," https://github.com/TransformerLensOrg/TransformerLens, 2022.