UNIVERSITÀ DEGLI STUDI DI TRIESTE

XXXVIII CICLO DEL DOTTORATO DI RICERCA IN

Applied Data Science and Artificial Intelligence

Protein Language Models: Interpretability and Applications

Settore scientifico-disciplinare: **INF-01**

DOTTORANDO / A
Edith Natalia Villegas García

COORDINATORE
PROF. Francesco Pauli

SUPERVISORE DI TESI
PROF. Alessio Ansuini

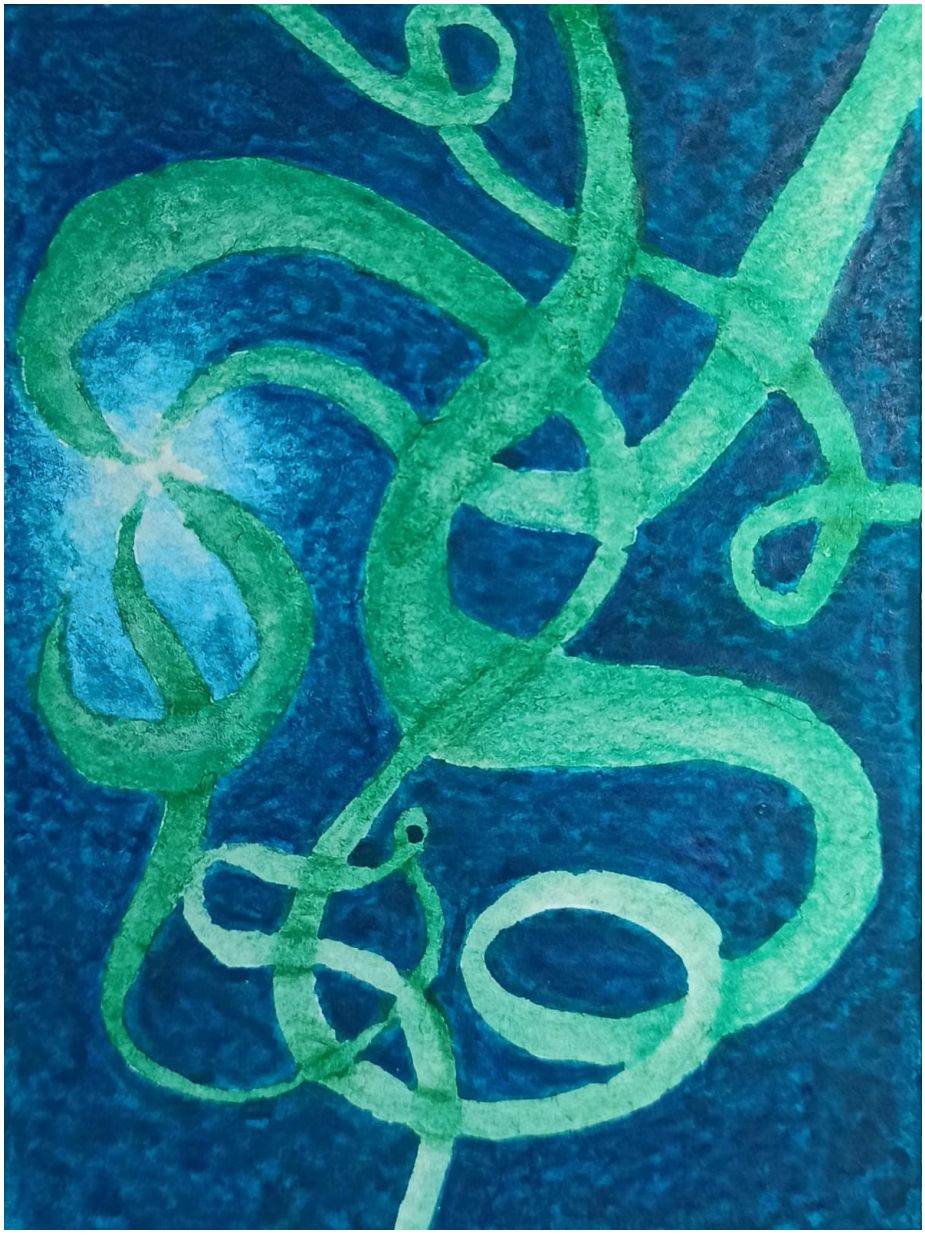
CO-SUPERVISORE DI TESI
PROF. Francesca Cuturello

ANNO ACCADEMICO 2024/2025

Protein Language Models: Interpretability and Applications

Edith Natalia Villegas Garcia

October 2025



Abstract

The advent of large-scale neural networks has revolutionized computational biology, enabling the development of powerful Protein Language Models (PLMs) that learn rich representations from amino acid sequences. This thesis explores the internal workings of these models through the lens of interpretability, with the aim of understanding the biological knowledge they capture and taking advantage of this understanding for practical applications.

We begin by investigating the intrinsic dimensionality (ID) of data representations across layers of deep neural networks trained on diverse modalities: DNA, proteins, natural language, and images (chapter 4). We find that while ID evolution varies across modalities, PLMs exhibit a remarkably robust and consistent pattern, suggesting they converge to a universal representation. This consistency makes PLMs a particularly fertile ground for interpretability research, as insights are likely to generalize across architectures.

Building on this, we apply Sparse Autoencoders (SAEs), a state-of-the-art mechanistic interpretability method, to dissect the representations of the ESM2-8M model (chapter 5). We successfully disentangle model representations and link them to biologically meaningful annotations from the UniProt database. Furthermore, we demonstrate that these features are actionable: by artificially activating SAE latents associated with zinc finger motifs during inference, we can steer the model to generate novel protein sequences containing these structural elements, validated by automated annotation and structure prediction.

We complement this with a classical analysis of model neurons (chapter 6), revealing that while individual neurons are polysemantic and each encoded concept is spread across the whole population of neurons; information about specific protein domains is concentrated in a small subset of these, with performance saturating rapidly as more neurons are added. We also identify the presence of "outlier dimensions" in PLMs, analogous to those previously described in natural language models, and show that one such dimension is strongly correlated with intrinsically disordered regions, potentially acting as a biological analogue of punctuation.

The practical utility of PLMs is further demonstrated through their application to predicting protein interaction interfaces (chapter 7). Finetuning PLMs for this task showcases their transfer learning capabilities on a problem of direct biological relevance.

Additionally, a bonus chapter (chapter 8) illustrates the power of unsupervised learning in a clinical context, where we cluster B-cell Chronic Lymphocytic Leukemia patient profiles to identify distinct risk groups based on treatment-free survival, resulting in a simple, clinically applicable algorithm for patient stratification.

In conclusion, this thesis provides a multifaceted examination of protein language models, from fundamental geometric properties and mechanistic interpretability to practical applications in protein generation and interaction interface prediction tasks. Our work underscores that PLMs are not merely black-box predictors but are learning structured, biologically-grounded representations that we can begin to understand, control, and exploit.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Alessio Ansuini, for his continuous support and encouragement throughout this work. I am also grateful to Alberto Cazzaniga and Stefano Cozzini for their help at AREA Science Park.

My sincere thanks go to Francesca Cuturello for her invaluable collaboration on both the leukemia and the protein interface interaction studies. I would also like to acknowledge my colleagues Chiara Moret, for her help with my work on intrinsic dimension estimation, and Yuri Gardinazzi and Antonio Feltrin for their work on the protein interface interaction projects. I would also like to thank to Federico Barone for our collaboration in the DPCfam project and for many insightful discussions throughout this period.

Finally, I wish to thank my family and my friends in Trieste and elsewhere for their constant support during these years.

Contents

Chapter 1

Introduction

Proteins are the fundamental workhorses of biology, orchestrating nearly every cellular process. Understanding their structure, function, and interactions is a central goal in life sciences. The exponential growth of biological sequence data has created an unprecedented opportunity to apply deep learning, leading to the development of Protein Language Models (PLMs) [81, 49]. Inspired by breakthroughs in natural language processing [141, 41, 109], these models treat amino acid sequences as sentences in a ”protein language” and, through self-supervised pre-training on millions of sequences, learn rich, contextual representations that encapsulate structural, functional, and evolutionary information.

Despite their remarkable performance on tasks like structure prediction [76, 81] and function annotation, PLMs are often regarded as black boxes. The complexity of architectures like the transformer [141], with its self-attention mechanism and millions of parameters, obscures *how* and *what* these models learn. The emerging field of *mechanistic interpretability* seeks to reverse-engineer neural networks into human-understandable components [91]. Applying these techniques to PLMs is not merely an academic exercise; it is crucial for validating their predictions, ensuring their reliability, and ultimately harnessing them for scientific discovery and protein design [5].

This thesis presents a comprehensive exploration of the interpretability of protein language models. Our work is structured to progress from analyzing fundamental geometric properties of model representations, to applying cutting-edge interpretability methods, and finally to demonstrating practical applications in both protein generation [60] and automated annotation (prediction of protein interaction interfaces).

Alongside the work presented here, we also collaborated with other colleagues on two different projects: Bayesian inference to deconvoluting mutational signatures [24] and unsupervised clustering to identify protein families in large datasets [14], further underscoring the power of computational approaches in biology.

We begin, in **chapter 2**, by establishing the foundational concepts of language models and the transformer architecture, as well as emerging alternatives like Mamba [63] and Hyena [105]. **Chapter 3** then provides a detailed overview of proteins as biological macromolecules and the computational tools and datasets used to study them.

The core of our investigation starts in **chapter 4.1**, where we probe the internal representations of neural networks by studying their *intrinsic dimensionality* (ID). We analyze how the ID of data representations evolves across layers of models trained on DNA, proteins, natural language, and images. A key finding is that PLMs exhibit a uniquely robust and consistent ID trajectory, even across different architectures and training regimes. This suggests that these models converge to a universal representation of protein space, making them exceptionally suitable for interpretability studies, as conclusions drawn from one model are more likely to generalize to others.

Motivated by this finding, **chapter 5** delves into mechanistic interpretability using *Sparse Autoencoders* (SAEs) [22, 33]. We train SAEs on the ESM2 model to disentangle its polysemantic representations into more interpretable features. We then develop an automated pipeline to interpret these features by matching them with curated annotations from the UniProt database, identifying latents associated with transmembrane regions, binding sites, and structural motifs like zinc fingers. Critically, we demonstrate that this interpretability is not merely descriptive but can guide model behavior: by manipulating these latent activations during inference, we can *steer* the model to generate novel protein sequences that contain zinc finger motifs, paving the way for potentially more complicated protein design applications.

In **chapter 6**, we supplement the SAE approach with a more classical analysis of model neurons in PLMs. We show that while individual neurons in PLMs are not easily interpretable, information about specific protein domains is highly concentrated in a small subset of neurons, with classifier performance saturating rapidly. This indicates a sparse and structured representation. Furthermore, we identify the phenomenon of *outlier dimensions* in PLMs, previously observed in natural language models [79]. We demonstrate that one such outlier dimension shows a strong correlation with intrinsically disordered regions, suggesting these regions may play a role analogous to punctuation in natural language.

The practical significance of PLMs becomes evident in **chapter 7**, where we finetune them to predict protein interaction interfaces. Accurately identifying which amino acids form the interface between interacting proteins is critical for understanding cellular signaling pathways, drug design, and synthetic biology. This chapter demonstrates the power of transfer learning, showing how knowledge embedded in PLMs during pre-training can be efficiently adapted to solve specific, high-impact biological problems.

Finally, **chapter 8** showcases an application of unsupervised learning in a clinical context, distinct from the PLM-focused work. We apply clustering algorithms to clinical and biomarker data from patients with B-cell Chronic Lymphocytic Leukemia. The resulting clusters correspond to distinct risk groups with significantly different treatment-free survival outcomes. We derive a simple decision tree that allows clinicians to stratify patient risk with high accuracy, demonstrating a direct path toward translating computational analysis into clinical practice [34].

In summary, this thesis bridges the gap between the abstract representations learned by deep learning models and their concrete biological interpretations and applications. We move from observing PLM performance to trying to understand their internal mechanisms, and ultimately, to controlling their output for potential protein design applications.

Chapter 2

Language Models

Abstract

This chapter introduces the foundations of language modeling and its extension to different data modalities. We begin by defining language models and discussing the impact of the transformer architecture [141] and its self-attention mechanism, which enabled unprecedented contextual understanding and generative capabilities (section 2.2). Through self supervised pretraining on large unlabeled corpora, models such as BERT [41] and GPT [109, 23] learn transferable representations that can be finetuned for a variety of downstream tasks. We describe the masked language modeling objective that underlies many such models (section 2.1) and outline the architecture of transformer networks (section 2.2). Finally, we present some emerging alternative architectures, including state space models, Mamba [63], and Hyena [105]—that aim to capture long-range dependencies more efficiently (section 2.3).

A language model is a model that can assign probabilities to given sequences of words. Recent advances in language modeling with neural networks have achieved remarkable capabilities for machine translation, generation of natural language text, question answering, and reasoning. These breakthroughs are largely driven by the *transformer* architecture proposed by [141], which makes use of the *self-attention* mechanism to incorporate contextual information. Later, models such as BERT [41] and GPT [109, 23] set new benchmarks on several different natural language tasks.

These models learn in a semi supervised manner, by first taking advantage of large amounts of unlabeled text now accessible on the internet for pretraining on a *pretext* task like language modeling, and then performing smaller scale training (*fine-tuning*) on a labeled dataset. The *pretext* task is a learning objective devised so that the model can learn semantically rich representations of the data that can then be re-used for other downstream tasks. A model trained this way to produce these representations is called a *foundation model*. This same approach has also been successfully applied to images [42], video [9], biological sequences [69, 74], and other multimodal types of data [108].

In the following sections, we describe in more detail the masked language modeling objective (Section 2.1), the transformer architecture and attention mechanism (Section 2.2), and some other architectures that have appeared

as alternatives to transformers (Section 2.3).

2.1 Masked Language Modeling

For natural language processing tasks, text first has to be split into words or word pieces, commonly referred to as *tokens*. This operation is called *tokenization*, and several different algorithms have been proposed to perform it in a more efficient manner ([123, 149]). For protein sequences, each amino acid can be considered a token; while for DNA, each nucleic acid can be taken individually or in groups of size k called k-mers.

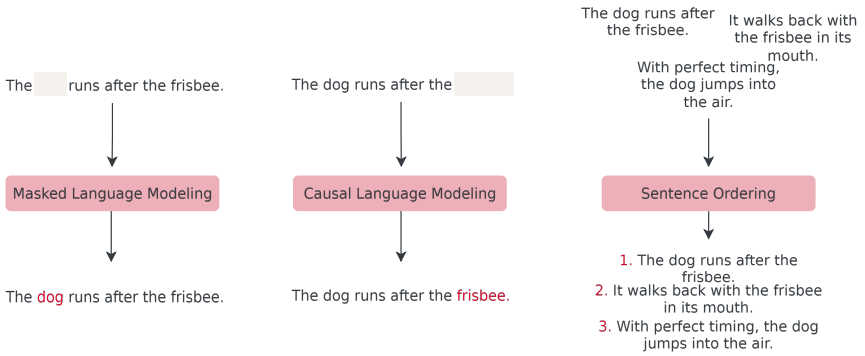


Figure 2.1: An illustration of the Masked Language Modeling, Causal Language Modeling (Next Token Prediction), and Sentence Ordering tasks. In Masked Language Modeling, random tokens are masked from the input, and the model has to predict the missing tokens based on the context (added comma). In Causal Language Modeling, the model predicts the next token based on previous tokens. In Sentence Ordering, the model has to organize sentences in the correct sequence.

Language models are usually trained on the tasks illustrated in figure 2.1. For example, BERT was trained for masked language modeling and next sentence prediction tasks, while GPT is an auto-regressive model that predicts the next token in a sequence. Protein language models are trained on the masked language modeling task, so we will focus our discussion on this task.

For masked language modeling, random tokens from the input are corrupted or replaced with a special $\langle \text{MASK} \rangle$ token, and the model is trained to directly predict the missing tokens given the rest of the context. The loss function for this task is an extension of the standard cross-entropy loss that is applied only to the masked tokens. It can be expressed as:

$$\mathcal{L}_{\text{MLM}}(\theta) = -\frac{1}{|M|} \sum_{i \in M} \log p_\theta(x_i \mid x_{\setminus M}) \tag{2.1}$$

where $p_\theta(x_i \mid x_{\setminus M})$ is the probability assigned by the model to the ground truth token x_i given its masked sequence context $x_{\setminus M}$, and M is the set of all

masked tokens.

2.2 Transformers

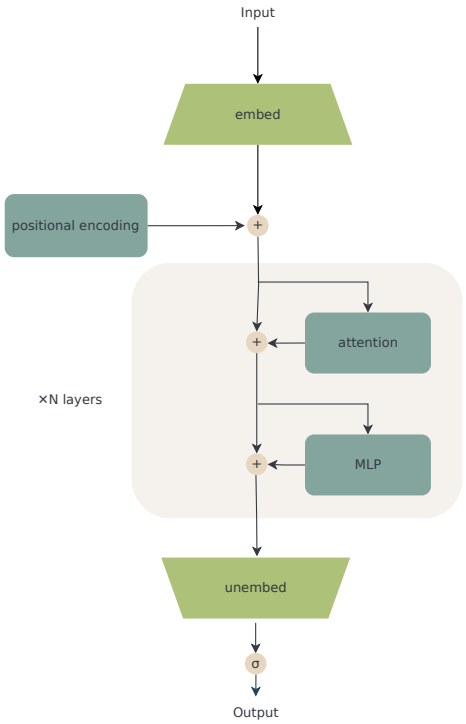


Figure 2.2: The Transformer architecture: The tokenized input is embedded through a linear projection, and an additional positional encoding is added so that the model can recognize positional information. The input is then passed through several transformer blocks before finally going through the unembedding layer, which projects the results back to the appropriate dimensions. These outputs are normalized through a softmax at the end of the model to produce the final output probabilities.

The main block of the transformer architecture consists of normal fully connected layers (MLP: multi-layer perceptrons) interleaved with *attention* layers, as shown in figure 2.2. The input is first tokenized, as described in the previous section, and a linear projection first *embeds* each token into a vector of dimension d (also called the *hidden dimension* of the model). We call the result of this operation the matrix X, which has dimensions (d, L), where L is the length of the original input sequence (in number of tokens). A *positional encoding* is added to the embedding so that the model can recognize positional information, which would otherwise be lost.

The *embedding* or representation X passes through several transformer blocks before being *unembedded* back to a matrix of size (L, D), where D

is the size of the dictionary (number of possible tokens). This final matrix is normalized so that, in the end, we have a list of probabilities for each token at each position in the sequence.

The core component of the transformer architecture is the attention mechanism. This mechanism allows the model to incorporate contextual information into the vector representation of each token. For example, in English, the same word can have completely unrelated meanings, and we are only able to determine the correct meaning based on context. In the sentences:

The dogs **bark**.

The **bark** of this tree is medicinal.

We understand the meaning of the word **bark** through other keywords in the rest of the sentence.

In the attention mechanism, each embedded token is re-weighted to include information from all other tokens. Intuitively, we assign a higher weight to tokens that are closer in meaning. We call the matrix that gives these weights the *attention map*. Mathematically, this is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{2.2}$$

Q, K, and V are called the queries, keys, and values vectors, which are linear projections of the input X obtained by multiplying with the corresponding weight matrices W^Q, W^K, and W^V. These matrices act as the learnable parameters in the attention mechanism and give more expressivity to the network. The $QK^\top$ multiplication, scaled and normalized, acts as the attention map, which decides which tokens will "communicate" with each other.

Self-attention in a transformer model means that both the queries Q and the keys K are projections of the same input sequence, instead of two different sequences (i.e. a sentence in English and its translation in French). *Self-attention* is used in a network with a Masked Language Modeling objective, as opposed to a network used for translation tasks. In a standard transformer network, the self-attention mechanism is extended to *multi-head attention*. This is an extension where several attention maps are calculated in parallel, allowing the model to create different "similarity measures" in each head. This is given by:

$$\text{head}_i = \text{Attention}_i(Q, K, V) \tag{2.3}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.4}$$

where W_i^Q, W_i^K, W_i^V are the projection weights used for each attention head i in a model with h heads, and the final output is projected with the output weights W^O.

2.3 Alternative Architectures

Although transformer based models have enjoyed a great deal of popularity due to their effectiveness at solving language related tasks, they are not the most computationally efficient. In particular, the computational cost of the attention mechanism that they rely on increases quadratically with sequence length. This

limits the maximum length of the sequences that a transformer can process, and thus the maximum context window that is available to them. This problem has led to several groups trying to propose a model that uses a subquadratic operation that is as effective as transformers and also allows the context length to be increased.

To propose alternative architectures, some groups have built on insights learned from studying the mechanisms through which the transformer architecture works [56, 105, 63]. In particular, it seems that one key ingredient for transformers to be able to effectively learn from context is the formation of *induction heads*: attention heads that are able to complete patterns of the form $AB..A \rightarrow B$ [95]. Architectures that are capable of solving simple versions of this task seem to perform much better at larger scales on natural language tasks that rely on in-context learning (learning from the current context).

State space models (SSMs) have been proposed as an alternative to the attention mechanism, given that they are linear with sequence length and have proven successful for tasks such as time series modeling [64]. These models can be thought of as lying in between a recurrent neural network (RNN) and a convolutional neural network (CNN). SSMs map an input variable u_i to an output variable y_i by passing through an intermediate state x_i with the following equations:

$$x_i = A, x_{i-1} + B, u_i, \tag{2.5}$$

$$y_i = C, x_i + D, u_i. \tag{2.6}$$

where A, B, C, D matrices are learned from the data. This operation can also be written as a convolution, as the matrices are invariant with i.

As classical SSMs by themselves are not as effective for natural language tasks as attention, several changes have been proposed that also improve the computational efficiency of their implementations, such as H3 [56] and Mamba [63]. In the following section we will describe in a bit more detail the architecture of Mamba and Hyena [105], a closely related model.

2.3.1 Mamba

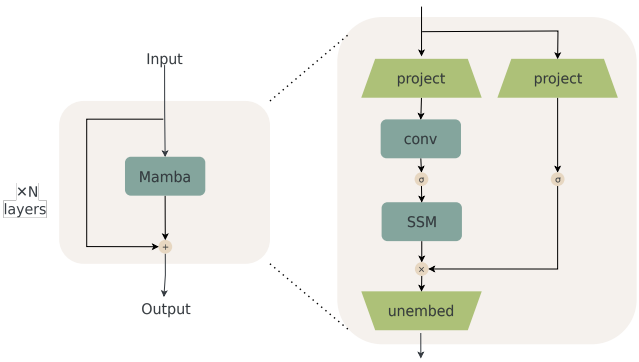


Figure 2.3: The architecture of the Mamba model and composition of Mamba blocks. Each Mamba block combines a short convolution that incorporates local context information into each token with an SSM block that incorporates global context information from all tokens. These blocks are followed by a gating operation (matrix multiplication between embeddings) that allows information to flow between different channels (hidden dimensions) in each token.

The Mamba architecture is a recently proposed model based on SSMs that incorporates several different tricks to make its implementation faster. The architecture is sketched in figure 2.3. The main innovation that makes Mamba different from other state space models is a *selection mechanism*. Basically, we make some of the matrices of the SSM dependent on the input instead of invariant. This gives the model the ability to select data in an input dependent manner and perform in-context learning [63].

The model architecture consists of stacked Mamba blocks. Each of these blocks includes a short convolution that enriches the representations with local context before the SSM block: this part of the model combines information from different tokens. Then, a gating mechanism allows the model to combine information from different hidden dimensions in the representations.

2.3.2 Hyena

The Hyena hierarchy model [105] is a recently proposed architecture where traditional attention blocks are replaced by a *Hyena operator*. In this operator, we apply stacked element-wise gating operations followed by long convolutions to different versions of the input that have undergone a linear projection and a small convolution. A diagram of this architecture is shown in figure 2.4.

The gating and long convolutions can be thought of as successive multiplications in the original and Fourier space. The parameters in the kernel for the long convolution are not directly learned; instead, they are output by a short MLP that takes as input the size of the desired kernel. In this way, the size of the input sequence can be kept flexible.

The large context length that is achievable with the Hyena architecture

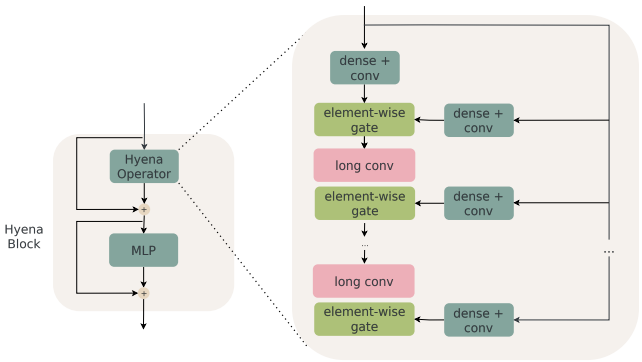


Figure 2.4: Hyena architecture. The architecture is very similar to the transformer architecture but with attention blocks replaced by Hyena operators. Each Hyena operator combines element-wise gating with successive long convolutions (convolutions with the same size as the input, parameterized by an MLP) before the final output.

makes it well suited for DNA modeling, where interactions can be several hundred thousand nucleotides apart [89].

Other groups have tried creating hybrid models that take advantage of the strengths of the different types of layers. An example is the striped Hyena architecture, where transformer layers are interleaved with Hyena layers to create an efficient and performant model that has been applied to both natural language [106] and DNA data [88].

Chapter 3

Proteins

Abstract

This chapter provides an overview of proteins as fundamental biological macromolecules and introduces the computational tools and datasets used to study them. We begin by describing the structure and function of proteins, from their amino acid composition and folding into secondary and tertiary structures to their diverse biological roles (Section) 3.1). Building on this foundation, we introduce protein language models (PLMs)—deep learning architectures inspired by natural language processing that learn representations of protein sequences useful for structural and functional prediction (Section) 3.2). We summarize major PLMs such as ESM, ProtTrans, Ankh, and MSA-Transformer, along with models for structure prediction. The chapter also surveys key biological databases (Section) 3.3) such as UniProt, InterPro, and the Protein Data Bank (PDB), which provide curated protein sequences, annotations, and structures, and introduces essential computational tools (Section) 3.4) like MMSeqs2 and PyMOL for protein analysis and visualization.

3.1 What are proteins?

Proteins are among the fundamental building blocks of life [148]. They can play many different roles inside an organism: as enzymes for various biochemical reactions, as structural support such as collagen and keratin, as receptors for cellular communication, or as molecules that transport other molecules and combat pathogens in more advanced organisms.

Proteins are chains of amino acids (polypeptides) that fold into a complex 3D structure that usually determines their biological function. Interactions between amino acids dictate the formation of secondary structure elements such as α-helices and β-sheets (figure) 3.1), which in turn form the overall tertiary structure of the protein. The protein sequence is encoded by a DNA sequence that is translated into an amino acid sequence.

There are 20 standard amino acids found in nature that have a fixed correspondence with codons in DNA. If we assign each amino acid a letter in the alphabet, we can represent a protein as a sequence of letters. These se-

quences can be studied through language models, an approach that we detail in section 3.2.

The series of amino acids that form a protein can be determined through direct sequencing, through sequencing of intermediate gene products, or predicted from the corresponding gene. A protein's 3D structure can be resolved through different experimental methods, mentioned in Section 3.3.5. Different databases that host protein sequences and biological annotations, as well as other tools for working with proteins, are detailed in Sections 3.3 and 3.4.

3.2 Protein Language Models

Given the success of the transformer and masked language modeling for natural language, several groups have tried to apply the same methodology to protein sequences. Proteins are represented as sequences of letters or tokens, and a certain percentage of the data is randomly masked or corrupted before being passed as input for model training. The model then learns how to complete the missing parts of the sequence, and in doing so, forms protein representations that contain a lot of useful information for other tasks. Some of the first examples of these types of models include the ProtTrans family of models [49] and ESM (Evolutionary Scale Modeling) [115], which were trained on billions of protein sequences. Later on, a second version of ESM scaled this approach to much larger models (ESM-2) that were then used with additional layers for structural prediction (ESM-Fold) [81]. Other approaches, like Ankh, explore the influence of hyperparameter optimization on performance improvement while keeping parameter count low[48]. An extension of this approach to multiple sequence alignments is given by [112], which builds a model that they call the MSA-Transformer.

The representations that these models learn for proteins have been shown to contain a lot of useful structural, functional, and evolutionary information, although because of their black-box nature it is not clear how they process this information. In transformers, it has been observed that the attention mechanism focuses mainly on binding sites, with attention maps seemingly encoding residue-to-residue contacts [142]. There also seems to be a progression in attention from lower-level concepts (secondary structure features) in earlier layers to higher-level concepts like binding sites and residue contacts in deeper layers. Unsupervised contact prediction from attention maps has also been explored by [113].

With the masked language modeling approach on its own, networks learn protein representations that contain a lot of structural information, but the representations can be improved by directly incorporating structural information, such as with ESM-3, a multimodal model that also uses functional labels for its training [69].

The problem of predicting a protein structure from its sequence was also famously addressed by AlphaFold, with a model that contains a submodule similar to the MSA-Transformer [76], and more recently with a diffusion model [4]. This problem can also be addressed as a sequence to sequence translation task (such as in the ProtT5 model [71]), given that the protein structure is tokenized into a structural sequence. This tokenization into a structural alphabet has been done by [140] with an autoencoder model which was originally

developed for fast structural alignment (Foldseek). Predicting an amino acid sequence from a given protein structure, also known as inverse folding, has been addressed by models such as ESM-IF [73].

Recent architectural advances in natural language modeling have been incorporated into the ProtMamba [124] and PTM-Mamba [102] models.

3.3 Datasets

3.3.1 UniProt

The UniProt (Universal Protein Resource) database is a large collection of protein sequences and corresponding annotations [2, 20]. Most sequences come from directly translating coding sequence data, but there are different levels of evidence for existence assigned to each protein, ranging from experimental measurements (proteins with a resolved structure in the PDB or from direct protein sequencing) to proteins whose existence is inferred from homology to other proteins.

UniProt is subdivided into different resources:

- **UniProtKB**: The UniProt Knowledgebase is the main resource, containing several million protein sequences and numerous biological annotations where available. It comprises the SwissProt set, which is manually curated and reviewed by experts, and the TrEMBL set, which contains sequences that have been annotated computationally.

  The latest release of UniProtKB (version 2025_03, as of August 2025) contains more than 573,000 protein sequences in SwissProt and more than 253 million protein sequences in TrEMBL [31].

- **UniParc**: A non redundant archive of all protein sequences that have been published in major databases, including sequences that have been deemed erroneous. UniParc assigns a stable identifier to each unique sequence, making it possible to track it across different databases.

- **UniRef**: UniProt Reference Clusters provide sequences clustered at 100%, 90%, and 50% sequence identity (termed UniRef100, UniRef90, and UniRef50).

- **Proteomes**: Proteomes are the sets of proteins that are expressed by a biological organism. UniProt contains around 25,000 *reference proteomes* that have been selected to be representative of different organisms across the tree of life.

3.3.2 UniProtKB Annotations

Each entry in the UniProtKB database contains several biological annotations that are associated with its protein sequence. This usually includes the name of the associated gene and the taxonomic identifier and lineage of the original organism (with its NCBI, National Center for Biotechnology Information, taxonomy identifier) [122]; experimentally determined structures from the Protein Data Bank, and several other annotations of structural and functional features.

In the following section, we describe some of these selected features, also shown in figure 3.1.

Functional Annotations

- *Gene Ontology Annotations*: Annotations of the function and location of the protein, described in more detail in section 3.3.3.
- *Enzyme Commission Number*: A hierarchical classification of enzymes based on the chemical reactions they catalyze [84].

Functional Sites

- *Binding Site*: Specific residues that interact with another protein or a small molecule.
- *Active Site*: The residues in enzymes at which biochemical reactions occur.
- *Zinc Finger Region*: A short structural motif stabilized by a zinc ion, which typically binds DNA or RNA molecules.
- *Region of Interest*: In UniProt, a region of interest is a sequence segment with a particular biological significance that is not described by other labels.

Structural and Topological Annotations

- *Transmembrane Region*: A region of a protein that spans a lipid bilayer, and thus can fix it in place in a membrane.
- *Intramembrane Region*: A region of the protein embedded within the hydrophobic core of a lipid bilayer, inside a membrane.
- *Topological Domain*: In UniProt, this is a complementary label used for membrane-spanning proteins. It describes the subcellular location of a protein region with respect to a membrane, e.g., cytoplasmic, nuclear, intravirion, or extracellular.
- *Secondary Structure*: Basic secondary structure elements: α-helices, β-sheets, and turns.

Post-Translational Modifications (PTMs)

Post-translational modifications are chemical changes that amino acids in a protein can undergo after the protein has been synthesized [111].

- *Disulfide Bond*: The formation of a covalent bond between two cysteine residues that contributes to the structural stability of a protein.
- *Glycosylation Site*: The addition of sugar chains to specific residues through a covalent bond.

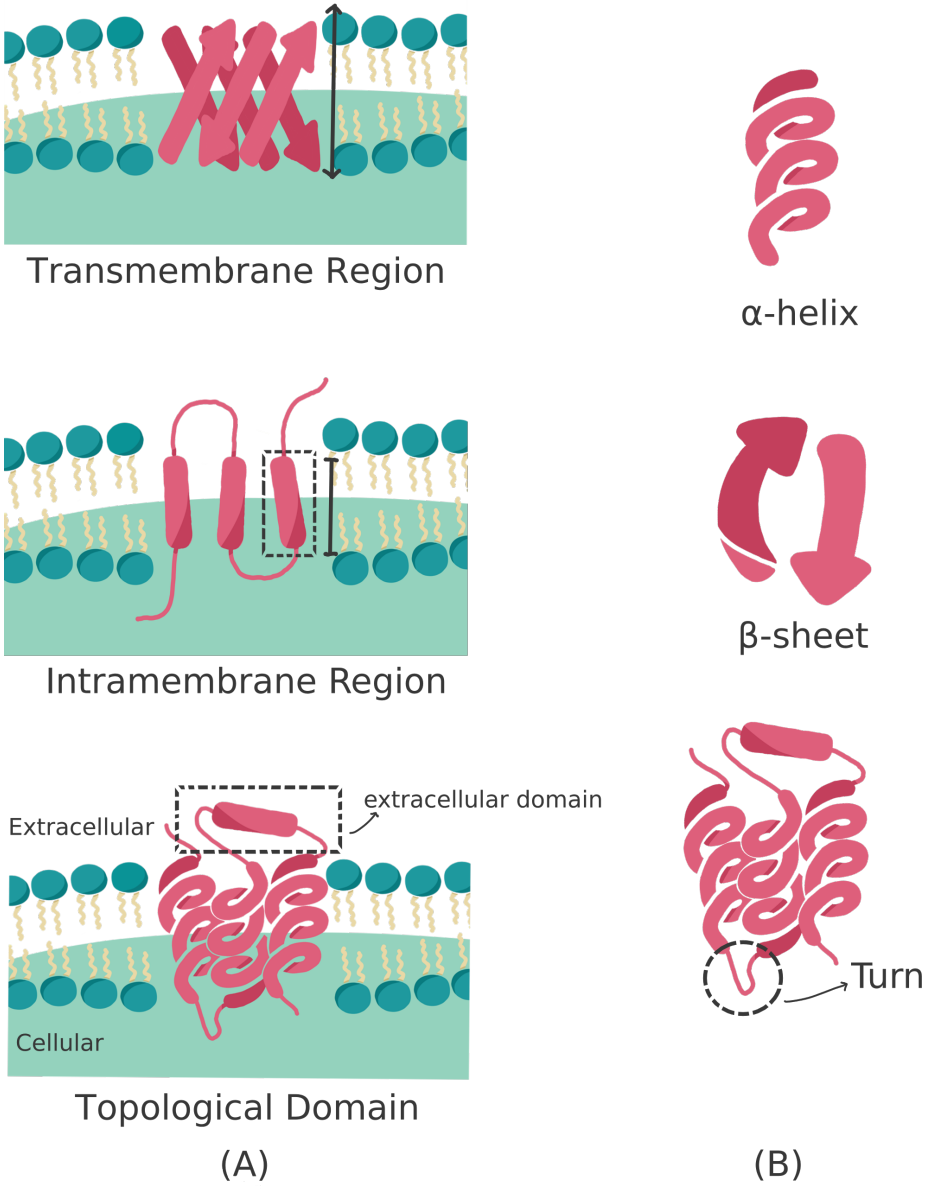


Figure 3.1: Diagram showing different types of annotations present in the UniProtKB database. (A): Domain labels with respect to a lipid bilayer. The transmembrane region traverses the membrane from one side to the other, like the beta-porin in the example. The intramembrane region is a domain contained within the membrane. A topological domain is a region of the protein not in the membrane, in this example an extracellular domain. (B): Basic structural elements of proteins: α-helix, β-sheet, and turns.

3.3.3 Gene Ontology (GO) Annotations

The Gene Ontology (GO) is a structured, controlled vocabulary used to describe the functions of genes and gene products consistently across species and databases [6, 11]. It provides a way to represent biological concepts in a standardized way. Words in the Gene Ontology are called GO terms; each GO term is given a unique identifier and a textual description, as well as hierarchical relations to other GO terms. GO annotations are relationships between a GO term and a specific gene product (e.g., a particular protein). They are divided into three main categories:

- **Molecular Function**: Defines the basic activity of a gene product at the molecular level (such as catalysis or transcription regulation). Examples include *insulin receptor activity* and *protein kinase activity*.
- **Cellular Location**: Describes the location where the molecular function takes place. This includes cellular structures like the cell membrane, cytoskeleton, and nucleus, but also viral components and protein-protein complexes.
- **Biological Process**: Describes the larger biological pathways or goals that a gene or gene product contributes to. Example terms include *DNA repair* and *apoptosis*.

3.3.4 InterPro

InterPro is a tool for predicting the presence of protein domains in amino acid sequences [19]. A protein domain is an evolutionarily conserved region that acts as an independent structural and functional unit (see figure 3.2). Several different databases exist, with manually curated or automatically discovered protein domains based on sequence [101, 126, 14, 118] or structural data [29, 145]. InterPro combines these databases into an integrated resource for studying protein sequences.

3.3.5 Protein Data Bank (PDB)

The Protein Data Bank (PDB) is an open-access archive for experimentally determined 3D structures of biological macromolecules, including proteins, nucleic acids, and complex assemblies [1]. It is managed by the Worldwide Protein Data Bank consortium, which oversees the collection of experimental data globally, and currently hosts over 241,000 structures. Most of these were determined using X-ray crystallography, while others were obtained through nuclear magnetic resonance (NMR) spectroscopy and, more recently, an increasing number through electron microscopy (EM) [107].

The standard file format for the description of structures in the PDB is PDBx/mmCIF (macromolecular crystallographic information file), which describes atomic coordinates as well as experimental metadata [1].

3.3.6 Structural Classification of Proteins (SCOP)

The Structural Classification of Proteins (SCOP) database is a manually curated hierarchy of protein structural domains that classifies proteins according to structural and evolutionary relationships [86]. Proteins are grouped into families, which are then grouped into superfamilies, folds, and classes.

- *Family*: proteins with at least 30% sequence similarity, but also some proteins with lower similarity that share very similar structure and function.
- *Superfamily*: families that share structural and functional features that suggest a common evolutionary origin.
- *Fold*: superfamilies that have the same major secondary structures in the same arrangement.
- *Class*: groups of folds classified into five main classes based mainly on their α and β content (α: mostly α-helices, β: mostly β-sheets, α/β: a combination of α-helices and β-sheets, $\alpha + \beta$: mostly segregated α and β, multi-domain: containing domains of different folds).

The Structural Classification of Proteins — Extended (SCOPe) is an extension of version 1.75 of the SCOP database, where new experimental structures are semi-automatically added to the existing classification [27].

The ASTRAL compendium additionally provides sequences for the SCOPe domains, filtered to different levels of sequence identity to avoid redundancy [55].

Figure 3.2: Illustration of two different structural domains within a protein, highlighted in orange and blue.

3.4 Tools

3.4.1 MMSeqs2

MMSeqs2 (Many-against-Many sequence searching) [129] is a toolkit for the alignment and clustering of biological sequences (protein and nucleotide sequences). MMSeqs2 is widely used by the bioinformatics community to fil-

ter redundant protein sequences by clustering or selecting representative sequences based on a given similarity threshold.

MMSeqs2 accelerates sequence similarity searches by performing a prefiltering step that involves breaking up protein sequences into k-mers and identifying high scoring k-mer matches. The MMSeqs2 implementation incorporates several key computational optimizations that make it orders of magnitude faster than older popular tools such as BLAST [7].

3.4.2 PyMOL

PyMOL [37, 116] is a widely adopted, (mostly) open-source molecular visualization platform used in structural biology to analyze and visualize macromolecular structures and trajectories. Written in Python, PyMOL generates high quality 3D renderings of molecules such as proteins, nucleic acids, and small ligands, and supports several file formats, including the PDB standard.

Chapter 4

Intrinsic Dimensionality through a Neural Network

Abstract

The intrinsic dimension of a dataset is the minimum number of parameters needed to describe it. In this chapter, we investigate how the intrinsic dimensionality of data representations evolves across layers of deep neural networks trained on diverse modalities, including DNA, proteins, natural language, and images. The intrinsic dimension serves as a geometric probe of how neural networks process information. Vision models display a "hunchback" pattern: an early rise followed by a decline, mirroring results from convolutional networks. Protein language models exhibit a reproducible two phased pattern: an initial ascent, followed by a dip in the intrinsic dimension estimated through local neighborhoods and a second peak at larger scales. DNA and natural language models show more variable behavior, with transformer-based architectures producing modality-specific curves that depend strongly on architecture and dataset diversity.

Comparing across modalities, we find that protein models display the most robust and consistent intrinsic dimension trajectories, even across multimodal architectures, supporting the view that these models have converged to some sort of universal representation. These findings position protein models as particularly interesting for interpretability studies, where insights into their internal organization are likely to generalize across architectures and training regimes.

Natural data like images and text occupy high-dimensional spaces (e.g., millions of pixels for photos or thousands of different words/tokens for language), but valid examples tend to lie on or close to a much lower-dimensional structure within this space, known as a manifold. This idea is called the manifold hypothesis, and it is the justification behind dimensionality reduction techniques such as Isomap [135], t-SNE [83] and UMAP [85] (see figure 4.1). The minimum number of parameters needed to describe this manifold is called the *intrinsic dimension* [18]. In other words, the intrinsic dimension is the answer to the question: what is the smallest number of variables needed to describe a dataset?

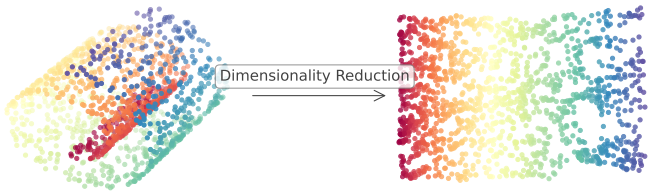


Figure 4.1: The Swiss roll dataset: a classic 2D manifold that curves through 3D space. Besides some noise, points in the dataset can be localized with only two parameters. Through dimensionality reduction algorithms like Isomap [135], this dataset can be flattened to a 2D representation without significant information loss.

Studying how the values of the intrinsic dimension evolve through the layers of a neural network can potentially help us understand some aspects of how the model processes information internally. Intuitively, we could expect the intrinsic dimension to increase in early layers, as the network extracts several different features from the data and combines them in different ways to create more complex ones. For example, it has been seen that early layers in vision models contain components that correspond to edge detectors in several different orientations, and blurred detectors for different colors and textures [90]. The intrinsic dimension then decreases in later layers as the network discards irrelevant information and compresses the representation into a more compact form [8, 139]. In the vision example, different textures and shapes are combined to create distinct object categories like cars, animals and heads. The intrinsic dimension trajectories can help us compare different models and architectures (e.g., convolutional neural networks vs. transformers vs. state space models) and understand how likely they are to be converging to similar internal representations. Additionally, the values of the intrinsic dimension are a potential tool for identifying efficiently compressed representations in models that are more likely to have good generalization [8].

In this chapter, we will analyze the intrinsic dimension along the layers of neural networks trained on biological sequences (proteins and DNA) and natural data (text and images).

4.1 Previous Works

The first work to study how the intrinsic dimension of representations evolves across the layers of a deep neural network was [8]. They observed a consistent *hunchback* pattern (where the intrinsic dimension initially rises, peaks at intermediate layers, and then declines) in multiple convolutional neural net-

works trained for image classification, as well as an inverse relationship between model performance and the final value of the intrinsic dimension, with better performing classifiers exhibiting lower dimensionality in their last hidden layer.

Later, [139] performed a similar study on transformer models for protein and image data, finding that intrinsic dimension evolution follows a consistent trend with an initial peak followed by a plateau region. Intermediate layers in the plateau, where the intrinsic dimension reaches a local minimum, seem to be more semantically rich (capturing remote homology in proteins and class labels in image data) and better suited for other downstream tasks. This plateau value also seems to agree with the one measured by [52] for proteins belonging to the same family (an indirect measure of the variability allowed in sequence evolution). The general shape of this intrinsic dimension curve also aligns with the intuition that transformer models act as compression algorithms [38].

A similar analysis for natural language transformers [28] found a comparable intrinsic dimension progression curve, but unlike [139], it concluded that representations at the peak (rather than the plateau minimum) were most effective for transfer learning. Other works include [144], which studied the evolution of the intrinsic dimension in language models at the token level, and [138], which found that AI generated text has a significantly lower intrinsic dimension than human written texts, and proposed this metric as a way to detect it. More recent work has shown that intermediate layer representations are better suited for downstream tasks in state space models as well as in transformers [128].

4.2 Intrinsic Dimension Estimation

Several different techniques for estimating the value of the intrinsic dimension of a dataset have been proposed [53, 26, 117]. Here, we use GRIDE (Generalized Ratios Intrinsic Dimension Estimator) [40], an estimator based on the Two-NN (Two Nearest Neighbors) estimator [51]. We will first describe Two-NN to provide the necessary background, and will then go on to explain how GRIDE differs from it.

4.2.1 Background: Two-NN

The Two-NN estimator is based on the distance between data points and its two nearest neighbors, hence the name.

Given a data point x_i sampled from a d-dimensional manifold, we denote the distance to the j-th nearest neighbor as $r_{i,j}$. Under the assumption of constant local density, it can be proven that the ratio of the distances to its two nearest neighbors $\mu_i = r_{i,1}/r_{i,2}$, follows a Pareto distribution given by

$$f(\mu) = d\mu^{-(d+1)} \tag{4.1}$$

and a corresponding cumulative distribution:

$$F(\mu) = (1 - \mu^{-d}) \tag{4.2}$$

from which we can calculate the dimension d

$$d = -\frac{\log\left(1 - F(\mu)\right)}{\log\left(\mu\right)} \tag{4.3}$$

We can estimate the intrinsic dimension by calculating the ratio μ_i for each data point and its empirical cumulative distribution $F_{emp}(\mu_i)$. By plotting $-\log\left(1 - F_{emp}(\mu_i)\right)$ against $\log\left(\mu_i\right)$ and fitting a line through the origin, the slope of this line will give us the estimated value.

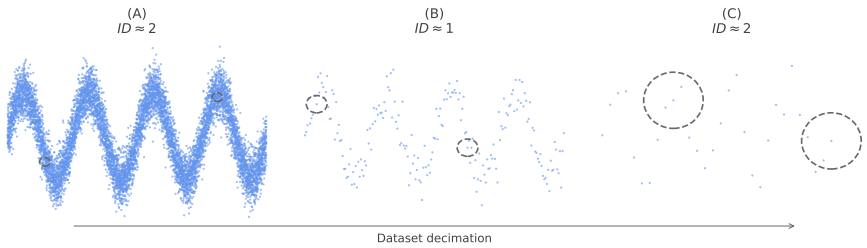


Figure 4.2: These plots illustrate the influence of scale on the estimated value of the intrinsic dimension (ID). The points are generated from a sinusoidal curve with gaussian noise along the y-axis. (A) In the first dataset points are packed with a high density, and the nearest neighbors to each point are very close together at a distance where noise in the dataset is relevant and in practice there are two degrees of freedom. (B) When some of the data points are decimated, the noise influences the estimation less and the 1D structure of the data becomes clearer for the algorithm. (C) When the density of points becomes too sparse, the curvature of the sine becomes more decisive and pushes the estimation towards a value closer to 2.

The value of the intrinsic dimension changes depending on the size of the neighborhood involved in the estimation (as illustrated in figure 4.2). For this reason, it is important to perform a *scale analysis* and estimate the intrinsic dimension on subsequently smaller subsets of the original dataset (by decimating the data). We choose as the final value of the estimation the one that remains stable across several different dataset sizes.

4.2.2 GRIDE

The Generalized Ratios Intrinsic Dimension Estimator (GRIDE) builds on the Two-NN method but extends it to work with nearest neighbors of different ranks. This allows the estimation of the ID as an explicit function of scale without performing any decimation [40].

Given a data point x_i and its n_1-th and n_2-th nearest neighbors, we define the ratio:

$$\mu_{i,n_1,n_2} = \frac{r_{i,n_1}}{r_{i,n_2}} \tag{4.4}$$

If we assume that the density of data points remains constant in the neighborhood size, the value of μ_{i,n_1,n_2} should follow a generalized Pareto distribution:

$$f_{\mu_{i,n_1,n_2}}(\mu) = \frac{d(\mu^d - 1)^{n_2 - n_1 - 1}}{\mu^{(n_2-1)d+1} B(n_2 - n_1, n_1)} \tag{4.5}$$

where $B(\cdot, \cdot)$ is the beta function. The details of this derivation can be found in [40]. An estimate of the intrinsic dimension can then be obtained by numerically maximizing the likelihood.

For our analyses, we use $n_2 = 2n_1$ for the scale of the neighborhood. We also refer to n_2 as scale or k.

4.3 Estimates on Neural Networks

In the following section, we describe in detail experimental estimates of the intrinsic dimension obtained on several different neural networks using GRIDE. The datasets used for the estimation are described in section 4.3.1, and the models in section 4.3.2. Each data point was embedded through one of the models, and representations were extracted through the layers at the end of each block (e.g., at the end of each transformer block in transformer models). We describe the resulting curves in section 4.3.3, followed by some conclusions.

4.3.1 Datasets

Biological Sequences

For the biological sequence models, we use a matched dataset of genes and proteins coming from prokaryotic organisms that we built from proteins in the Uniprot Swissprot database [2] with genomic sequences from the NCBI (National Center for Biotechnology Information) RefSeq database [94].

We extract all proteins originating from prokaryotic organisms in Swissprot version 2024_01, and filter for proteins with the highest levels of experimental evidence ("evidence at protein level" and "evidence at transcript level"). We use CD-HIT [57] to extract only proteins with at most 50% sequence identity (using a word length of 3 following CD-HIT's user guide).

Afterwards, we match each protein sequence to a sequence in the NCBI RefSeq database (using the "gene name" field from Swissprot), and kept only sequences with names starting with "WP" (these are non-redundant gene sequences that correspond to unique proteins). The genomic sequences were downloaded in May 2024. In total, we had $\sim$ 5.7k data points.

Additionally, for some DNA language models, we use a dataset of DNA sequences of length 1k extracted from the human reference genome (GRCh38 [121]). We had around $\sim$ 7.8k points in this dataset.

Natural Language and Images

Here we use images and image captions from the Microsoft Common Objects in Context (COCO) [80] dataset. We selected $\sim$ 5k images from the 2017 version.

4.3.2 Models

We analyze the following models:

DNA Models

- *Nucleotide Transformer*: This family of transformers is composed of encoder-type models with the same architecture as ESM2. Nucleotides are tokenized as k-mers of 6 characters each, with a maximum context length of up to 6kbp (kilobase pairs). Different variants of the model are trained on the human reference genome (500M parameters); the 1000 Genomes Project, which contains variants of the whole human genome (500M and 2.5B parameters); and a multi-species genomic dataset [35] (2.5B parameters). The size of the embeddings is 1280 for the 500M paramater models and 2560 for the 2.5B parameter models.
- *Nucleotide Transformer v2*: This is an improvement over the first version of the Nucleotide Transformer with rotary positional encodings, longer training times, and a longer sequence context of 12kbp. All of the models are trained on a multi-species genomic dataset [35]. There are four versions, each with 50M, 100M, 250M, and 500M parameters. The embedding dimension is 1024 for the biggest model, 768 for the 250M parameters model and 512 for the 50M and 100M model.
- *HyenaDNA*: A family of models with a Hyena architecture trained on the human reference genome with single nucleotide resolution [89]. Each model has a different context length, from 16k for the smallest model to 1M for the largest one. The number of parameters for these models is much smaller than for the transformer models (parameter counts $\sim$ 451k, 635k, 1.66M, 4.07M, 14.2M, 28.2M, and 54.6M). The embedding sizes are 128 for the smallest model and 256 for the others.
- *Caduceus*: These are models with bidirectional Mamba blocks and RC (reverse complement DNA sequence) equivariance, with single nucleotide resolution [120]. Different variants of the model were trained with a context length of 1k or 131k tokens. The largest model has around 7.7M parameters. The embedding sizes used were 118 (for 1k context) and 256 (for both 1k and 131k context).

Protein Models

- *ProtTrans*: The ProtTrans models were trained on the UniRef [2] or BFD [76] datasets, each with a different architecture inspired by a natural language model (T5, BERT, Electra, and XLNet). The models we studied here have the following parameter counts: ProtT5-XL-UniRef50 (3B), ProtBERT (420M), ProtElectra (420M), and ProtXLNet (409M). All the models have an embedding dimension of 1024.
- *ESM2*: Encoder type transformer models trained on UniRef sequences [81]. We study the models with sizes 8M, 35M, 150M, and 3B. The models have hidden dimensions of 320, 480, 640, 1280 and 2560.

- *Ankh*: These models maintain a similar architecture to the ProtT5 models but were more intensively optimized to be performant without having to scale up the parameter count [48]. The base model has 736M parameters, and the large model has 1.9B parameters (both with 48 layers). The hidden dimensions of the models are 768 for the base model and 1536 for the large model.
- *ESM3*: Multimodal transformer models trained on sequences from the UniRef and MGnify databases, as well as structures from the PDB, AlphaFoldDB, and ESMAtlas. Due to model availability, we only study ESM3-open, which has 1.4B parameters [69]. The embedding dimension is 1536.
- *ProstT5*: This is a T5-type transformer trained on a translation task: it translates from sequence (amino acids) to structure (FoldSeek alphabet representation). It has 3B parameters [71]. The embedding dimension is 1024.

Vision Models

- *Vision Transformer (ViT)*: This is a BERT architecture applied to image data [42]. Images are tokenized as sequences of patches of 16×16 pixels (or other patch sizes). The models were trained on ImageNet [39], with parameter counts of 86M for the "base" model, 307M for the "large" model, and 632M for the "huge" model. The hidden sizes are 768, 1024 and 1280.
- *Contrastive Language–Image Pretraining (CLIP)*: These are multimodal image language models that join an image encoder to a text encoder within the same architecture [108]. The image encoder part is the same as for ViT; the difference here is that they are trained with a contrastive learning objective: the image and text encoders are trained simultaneously to minimize the distance between representations of paired images and text. The hidden sizes are 768 for the base models and 1024 for the large models.
- *Shifted Windows Transformer (Swin)*: These are hierarchical vision transformers where patches of increasing sizes are processed at each stage in the model, making the process somewhat more similar to how a convolutional neural network would process the data [82]. They are trained on images from ImageNet. The sizes of the models are 29M for the "tiny" version, 50M for the "small," 88M for the "base," and 197M for the "large." The hidden dimensions are of 192, 192, 256 and 384.

Natural Language Models

- *Pythia*: These are a set of transformer models trained on the Pile dataset [58] specifically for interpretability research. They use rotary positional encodings, flash attention, and have a context length of 2048 tokens. We use the models ranging from 14M to 6.9B parameters. The hidden sizes go from 128 for the smallest model to 2560 for the bigger one.

- *Meta Llama 3*: The Meta Llama 3 models are transformer models with rotary positional encodings trained on a bespoke dataset [62]. We use the 3B and 70B models, which have a context length of 8k tokens. The hidden sizes are 4096 and 8192.
- *Mamba*: Models with the Mamba architecture [63], described earlier. We use the 130M, 370M, 790M, and 1.4B parameter models. The models were trained on the Pile dataset, with a context length of 2048, although they can process much longer inputs. The embedding dimensions are 768, 1024 and 1536.
- *Striped Hyena*: A Striped Hyena model (a hybrid between a transformer and a Hyena model [106]) trained on the RedPajama dataset [147] with 7B parameters and a 32k context. The hidden size is 4096.

4.3.3 Results

DNA

We calculated our intrinsic dimension estimates using the human reference genome dataset for the HyenaDNA and Caduceus models, as they were trained only on human DNA sequences. The results are shown in Figure 4.3. For the Nucleotide Transformer v2 models, we estimated the intrinsic dimension on both the prokaryotic dataset and the human reference genome. In both cases, the curves were qualitatively similar (with a slightly more pronounced minimum around 60% of the depth and a final ascent for the human dataset), so we show only the results for the multi-species prokaryotic dataset. For the Nucleotide Transformer v1 models, we use the human reference genome for models trained on human DNA (the human reference genome and the 1000 Genomes Project), and we show the intrinsic dimension curves estimated on both the human and prokaryotic datasets for the multispecies model. All curves shown in the main text use a $k_{max} = 16$, but the full-scale analysis can be found in Section B.

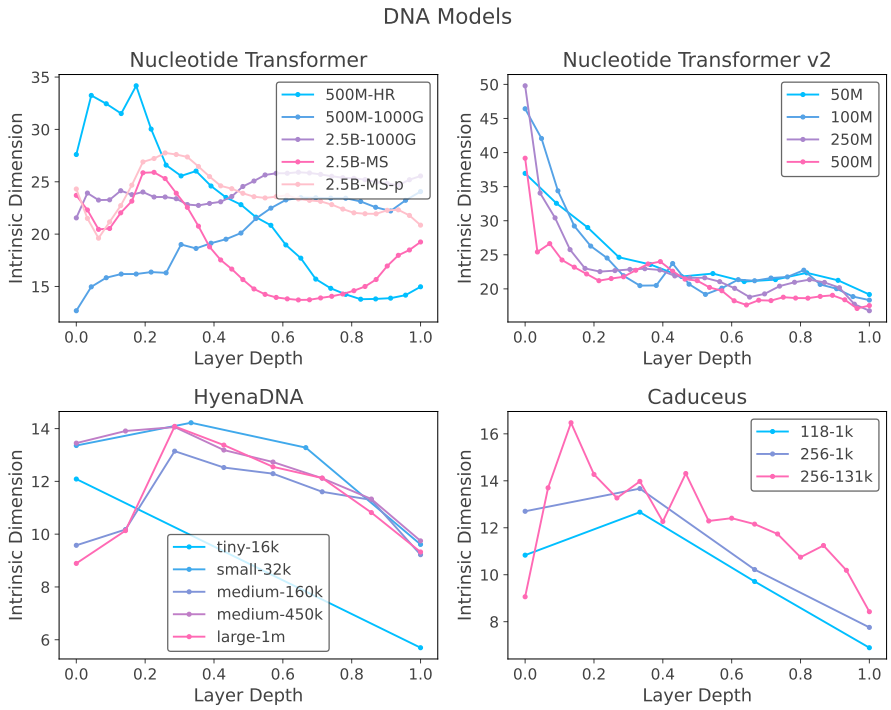


Figure 4.3: Intrinsic dimension estimates through the layers of DNA models (neighborhood size of $k = 16$). ***Nucleotide Transformer***: all of the curves are calculated on the human reference genome dataset, except *2.5B-MS-p*, which is the multi-species model calculated with the prokaryotic dataset. Models are identified by their parameter count and their training dataset (HR: human reference genome, 1000G: 1000 Genomes Project, MS: multi-species genomes). ***Nucleotide Transformer v2***: all curves are calculated on the prokaryotic dataset. Models are identified by parameter count. ***HyenaDNA***: curves are calculated on the human reference genome. Models are identified by their maximum sequence context length; their parameter count is detailed in the text. ***Caduceus***: curves are calculated on the human reference genome. Models are identified by their hidden size and context length.

We find that the HyenaDNA and Caduceus models, both non-transformer architectures, have intrinsic dimension curves that are consistent with the "hunchback" shapes observed by [8] in convolutional vision networks: an initial ascent and a peak followed by a slow descent in the curve.

The transformer models under study here behave much less consistently with one another. All the Nucleotide Transformer v2 models trained on multi-species genomes follow a systematic trend: a rapid ascent in the first layer (the intrinsic dimension of the input is not shown in the plot but is much lower than that of the first layer) followed by an exponential-type decay.

The Nucleotide Transformer v1 models follow a different trend for each case. It is worth noting that even though these models share the same architecture, they are trained on different datasets. The smallest model trained on the human

reference genome has an initial peak, a descent phase, and a slight final ascent reminiscent of the shape found by [139] for ESM2, a model with exactly the same architecture. The Nucleotide Transformer multi-species model shows a similar shape when evaluated on the human reference genome dataset but has a much less pronounced minimum when evaluated on the prokaryotic dataset. The Nucleotide Transformer models trained on the 1000 Genomes Project have a continuous ascending trend; the difference in this shape might be caused by the fact that this model was trained on a dataset of a slightly different nature (several different variants of the human genome instead of a single one), although evaluating the intrinsic dimension on a dataset with more genomic variety (the prokaryotic multi-species dataset) gave us qualitatively the same results.

Proteins

All of the models used for the protein section are transformers. In general, they all seem to follow the same overall trend described previously by [139], with a peak followed by a minimum and a final ascent phase, although some models have their own quirks (Figure 4.4). Not all models present the extended plateau phase that ESM2 models have, and some have more extended peaks. It is remarkable that models that contain structural information during their training also exhibit the same overall trend.

All the plots shown were calculated with a value of $k_{max} = 4$, which determines the size of the neighborhood. We find that for protein models, it is difficult to decide on a single value of the neighborhood size that provides a stable intrinsic dimension estimate for all layers. Additionally, and contrary to other data modalities, the qualitative shape of the intrinsic dimension curve varies drastically across different scales (see Figure 4.5 for the scale analysis of ESM2-3B). At larger scales, the plateau consistently turns into a peak for all models. This indicates a strong curvature in the data: locally, the manifold is low-dimensional, but at higher scales, the apparent dimension increases. We think that this could be due to two different possibilities: 1. a low-dimensional but extremely curved meandering manifold or 2. a representation space with a highly clustered structure.

The shape of the intrinsic dimension evolution for different representative models can be found in Section B of the appendix.

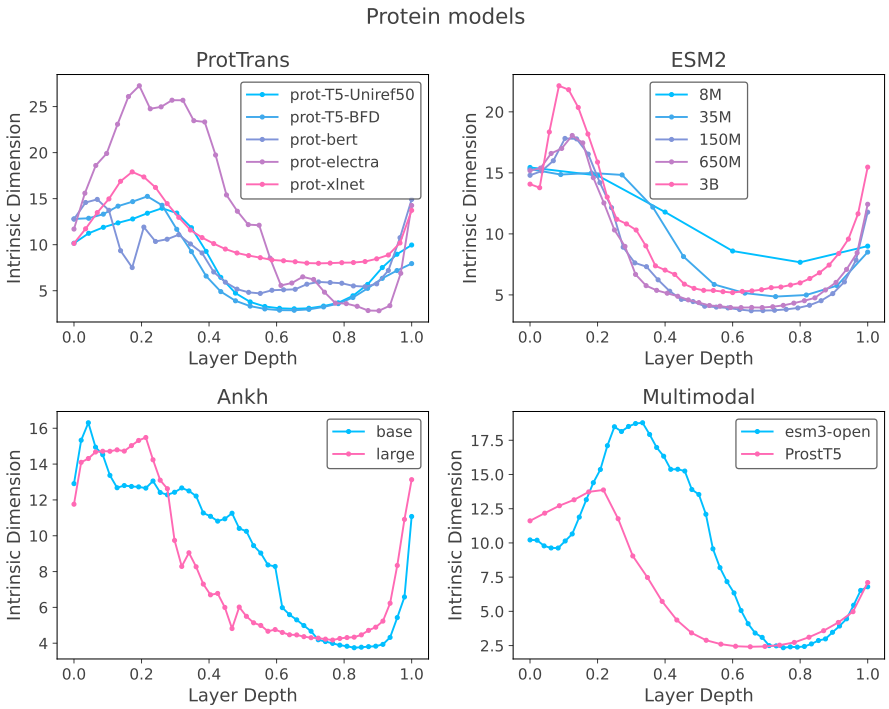


Figure 4.4: Intrinsic dimension estimates through the layers of the protein language models (neighborhood size of $k = 4$). All of them are evaluated on protein sequences from the prokaryotic dataset. ***ProtTrans:*** models are identified by their names (which include the base architecture and, in the case of T5, the training dataset — all other models are trained on Uniref50). ***ESM2:*** models are identified by their parameter count. ***Ankh:*** models are identified by their name. ***Multimodal:*** models are identified by their name; both models are from different families and training objectives.

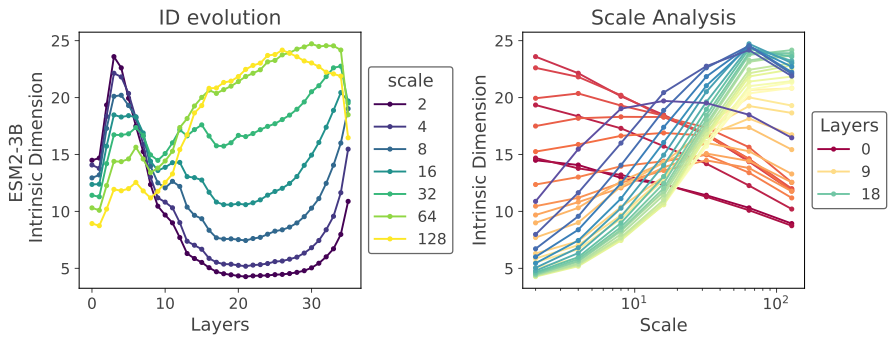


Figure 4.5: Neighborhood scale analysis for the ESM2 — 33-layer model. ***ID Evolution***: This plot shows the evolution of intrinsic dimension through the layers for different choices of neighborhood size (scale). ***Scale Analysis:*** This plot shows how the intrinsic dimension estimate changes with the choice of scale (neighborhood size) for each one of the layers of the model. Usually, this helps find a choice of scale where the estimate is stable; however, here it is difficult to pinpoint a single scale choice for all layers.

Images

For images, we compare a series of Vision Transformer model variants (see figure 4.6). Although less varied in architecture, these models also present a consistent pattern of intrinsic dimension evolution, reminiscent of the hunchback shape found by [8] for convolutional vision models. We note that CLIP and ViT have the same architecture but different training objectives, while ViT and Swin (which are more similar in intrinsic dimension evolution shape) have the same training objectives but slightly different architectures. We show the scale analysis for some representative models in section B of the appendix.

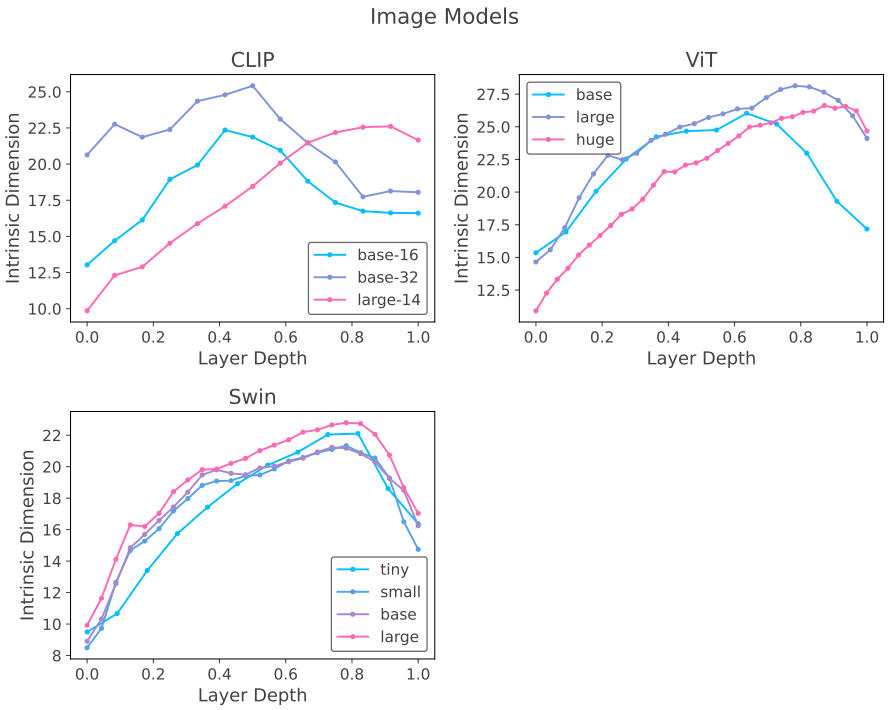


Figure 4.6: Intrinsic dimension estimates through the layers of the vision models (neighborhood size of $k = 8$). Models are identified by their names in each family. For the CLIP models, the number after the hyphen is the image patch size used by the model. The intrinsic dimension is estimated from the mean-pooled representation of an image.

Language

Here, we compare two families of transformer models (Pythia and LLaMA 3), as well as a Mamba model and a hybrid hyena-transformer model (Striped Hyena). All the results are shown in figure 4.7, with additional scale analysis in B. The Pythia models show a distinct pattern consisting of an initial peak followed by a slow descent (which becomes a second peak when using larger neighborhood sizes for the analysis) and a minimum. This pattern seems quite similar to the one shown by protein language models, but not exactly the same as the one found by [28]. This difference from previous works could be due to the fact that the dataset used to calculate these representations contains very short sequences compared to the maximum lengths supported by the model.

The Mamba model also seems to have a distinctive pattern with one main central peak at around halfway through the model. The LLaMA and Striped Hyena models have inconsistent shapes when calculated with the dataset used in this study, which again could be due to the short sequence length.

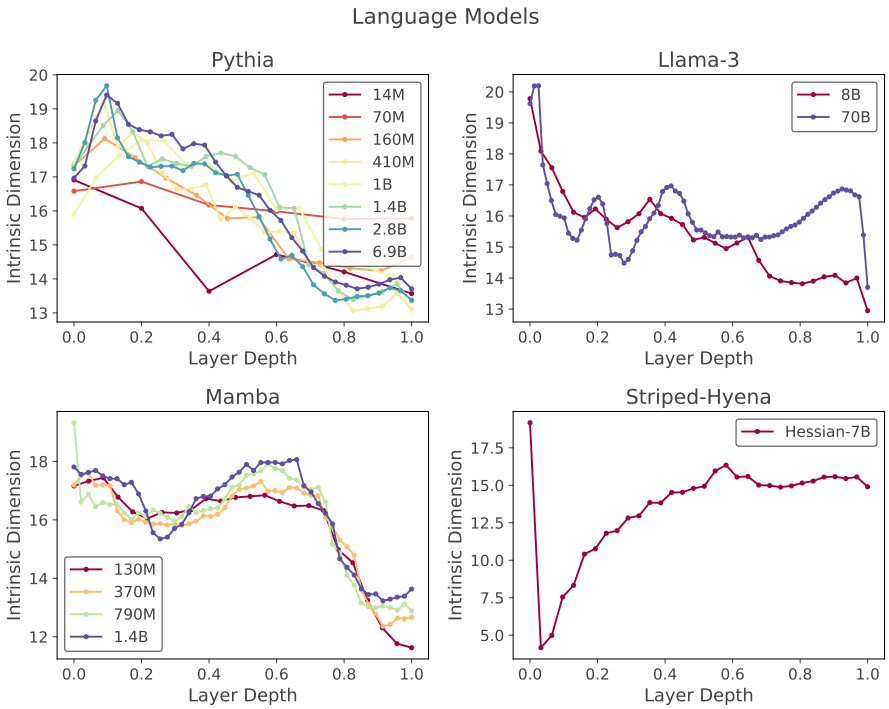


Figure 4.7: Intrinsic dimension estimates through the layers of the natural language models (neighborhood size of $k_{max} = 32$). Models are identified by their parameter count in each family. The intrinsic dimension is estimated from the mean-pooled representation of a sentence.

4.3.4 Conclusions

In this chapter, we studied the evolution of the intrinsic dimension through the layers of neural networks with different architectures and trained on different data modalities. We find some consistent patterns that have been reported by previous works, such as the hunchback pattern in vision models and the characteristic shape in protein language models. So far, these patterns seem less consistent for DNA and language data, and remarkably robust for protein data: all models, including multimodal ones, seem to have converged on a similar pattern. We conclude from this that protein language models are interesting case studies for interpretability research, given that statements about one model seem to be more likely to generalize to another.

Chapter 5

Interpretability and Sparse Autoencoders

Abstract

This chapter introduces the main ideas of mechanistic interpretability of neural networks and sparse autoencoders as a tool to extract interpretable concepts from a model (section 5.1.1). Later on we apply sparse autoencoders to study the representations of protein language models, specifically the ESM2-8M model (section 5.4). By performing a statistical analysis of each neuron in the sparse autoencoder, we link them to relevant protein annotations and identify potential interpretations such as transmembrane regions, binding sites, and specialized structural motifs. We further demonstrate that manipulating selected activations during inference (section 5.4.2) enables steering of sequence generation toward target motifs. Intervening on zinc-finger–associated neurons yields sequences containing zinc-finger structural elements, as verified by automated annotation and structure prediction (figure 5.7). Overall, our results show that sparse autoencoders have the potential for disentangling and controlling the internal representations of protein language models, offering a mechanistic tool for both interpretability and protein design applications.

5.1 Introduction

In spite of the massive success of large neural networks, it is extremely difficult to understand why and how a particular model works internally. This black box nature poses practical problems for model usage, as it makes it difficult to guarantee that models are safe, trustworthy and reliable as well as free from different biases [16]. Understanding how a model works can help with debugging, designing better and more efficient model architectures [105, 63] and provide new insights into problems that had not been solved before by humans [5]. Moreover, studying the inner mechanisms of artificial intelligence provides researchers with unique opportunities and a level of access that is not possible in biological systems [125].

The field of *mechanistic interpretability* aims to understand the internal mechanisms of neural networks with an approach similar to reverse engineering a computer program. In this approach, we can first try to understand what *features* are learned by a neural network and then how these are combined together in *circuits* to produce the final output. The term was first introduced by [91] in a series of articles (the *Circuits thread*) that aimed to intensively study vision in the Inception model [132]. The claim is that the features and circuits found in a model are somewhat generalizable to similar models.

In these series of articles, researchers found that visual features learned by the network get progressively more complex through the layers, starting from simple edge and line detectors to textures and then eyes and heads [90]. The features in one layer are combined through *circuits* that create more complicated features (see figure 5.1), for example curve detectors are created from simple line features in a way that can be understood and manually re-created [25].

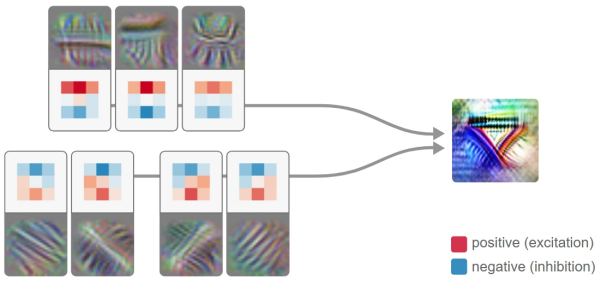


Figure 5.1: A simplified version of the circuit that creates a triangle detector based on simpler line detector features. Taken from [90].

Inspired by these findings, Anthropic created a similar series of articles for natural language models called the *Transformers Circuits Thread*. Initial exploration revealed that a big percentage of the capabilities of transformer models to learn from context seem to stem from a specific type of circuit termed an *induction head* [47, 95]. As already mentioned in previous sections, this circuit consists of two attention heads in successive layers that allow the model to complete patterns of the type $AB..A \rightarrow B$ and this insight has already been used to design more efficient language model architectures [56, 105, 63].

A fundamental problem for interpretability research in transformers is that "neurons" in this model do not have a one to one correspondence with definite "features" or concepts that we understand. Instead, features are thought to be encoded as vector directions in representation space: linear combinations of model neurons. For example, it has been found that there is a direction corresponding to "gender" in the representation space of language and word embedding models [98]. This idea -that concepts are encoded as linear directions in space- is known as the *linear representation hypothesis*, and there seems to be plenty of evidence to support it (although there are known counterexamples, such as cyclical features for the days of the week and months of the year found in [50]). This partially explains why transformer neurons are *polyse-*

mantic (having more than one meaning): as a lot of transformer operations are rotationally invariant, there is no incentive for the basis axis or "neurons" of the model to align with particular feature directions. These concepts are illustrated in figure 5.2.

In addition to this, it has been observed that language models seem to learn many more facts and concepts than they have parameters. [46] showed using toy models that transformers are incentivized to learn concepts in *superposition* to maximize parameter use. Concepts are encoded in directions that are almost orthogonal but cannot be recovered by a simple rotational transformation (see figure 5.2 (C)). This further complicates the problem of trying to interpret polysemantic neurons in transformers.

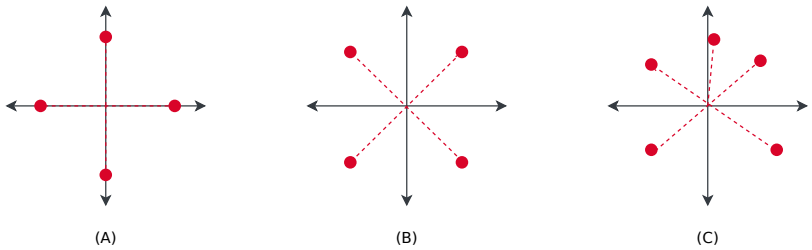


Figure 5.2: Illustration of feature directions in a neural network representation space. The axes are two independent "neurons" in the model.(A) In an ideal case, feature directions would align perfectly with the basis vectors, which would make neurons directly interpretable. (B) In a real neural network, the feature directions might be rotated with respect to the basis axes, which spreads concepts across different components and gives rise to polysemanticity. (C) Features have also been shown to be packed in *almost orthogonal* directions, where neural networks try to learn more concepts than they have parameters.

5.1.1 Sparse Autoencoders

To solve the problem of interpreting models with polysemantic neurons, we would like to have a method that first learns where feature directions are in representation space (ideally in an unsupervised manner). [46] first proposed dictionary learning to solve this problem in toy models that presented superposition, and then [22, 33] applied *sparse autoencoders* to small models trained on natural language data. This was then expanded to larger models by [133]. They found that, by using sparse autoencoders (SAEs), we can extract features that are much more easily interpretable than individual neurons and correspond to highly abstract human-level concepts like sarcasm, sycophancy and empathy. Moreover, [133] found that these features are actionable: if we artificially activate them during inference, we can *steer* or guide the model to produce an output that contains those features.

We will describe the original sparse autoencoder architecture and training objectives in section 5.2. Since the first papers using this simple architecture, other works have proposed several improved versions like the top-k activation SAE by OpenAI [59] and the gated SAE by Google Deepmind [110]. The

SAE approach has been applied to several different language models, including non-transformer architectures like Mamba [146]. It has also been applied to convolutional vision models [61], multi-modal vision-language models [75] and more domain specific data, like x-rays [3] and medical text [96]. Apart from interpretability and steering, some research have applied SAEs to model editing: trying to "delete" specific information from a trained model [54].

5.1.2 Sparse Autoencoders on Biological Sequence Data

Given that sparse autoencoders can be used not only to discover which features a model is learning, but also to guide model generation towards specific types of output, their potential applications to biological modeling are immense. Applying SAEs to protein language models can, for example, help us learn more about biological mechanisms by telling us which features a model uses to make a certain prediction [5]. They could also be used in protein sequence design pipelines. And lastly, for editing out unsafe information from models, if necessary.

At the time of starting our project, there were no published articles applying SAEs to protein language models. Two articles appeared on the topic [127, 5] before ours [60]. [5] examined each SAE feature by hand to find possible biological interpretations, while [127] applied an automated pipeline to interpret each SAE latent. Ours [60] was the first to apply steering to generate a sequence containing a simple structural motif.

Later on, several other articles appeared on steering protein language models: using matryoshka SAEs to increase structure solvent accessibility [99], steering based on functionally predictive latents to obtain a protein sequence [137] and directly using neuron activations [13]. [87] performed the first study on how simpler motif detector latents are used to construct domain detectors that are ultimately used for protein contact prediction circuits in protein language models. There were also works expanding protein SAE labels to GO annotations [66] and even small gene language models like HyenaDNA [65].

5.2 Sparse Autoencoder Architecture

Sparse autoencoders that are used in interpretability research are usually single layer models which are trained to reconstruct the token representations of the language model being studied. The sparse autoencoder can potentially disentangle the features learned by the larger model because its hidden representation space is much larger than the original. SAEs have an additional sparsity constraint which forces the model to keep only a few components active at a time, which seems to make each component far more interpretable than in a standard model [134, 22, 33].

The autoencoder is constituted by a single layer encoder followed by a single layer decoder. The encoder is given by:

$$z = f_{enc}(x) = \mathrm{ReLU}(W_{enc}(x - b_{dec}) + b_{enc}) \tag{5.1}$$

where $x \in \mathbb{R}^d$ is an input token that has been embedded into the representation space of a given language model under study, and the output $z \in \mathbb{R}^n_{\geq 0}$ is

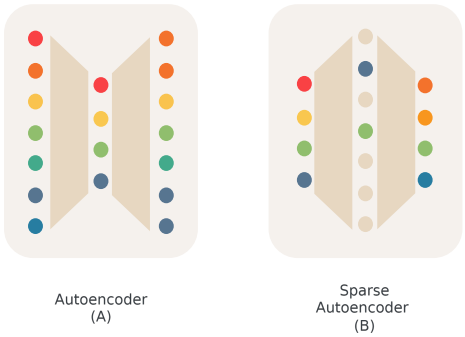


Figure 5.3: Comparison of the architecture of a standard autoencoder vs. a sparse autoencoder. Both autoencoders take a continuous representation as input and try to reconstruct it in the output after passing through a bottle-neck layer. (A) In the standard autoencoder architecture, the hidden layer has a smaller size than the input, so information has to be compressed. (B) In the sparse autoencoder architecture, the hidden layer is bigger in size but a sparsity constraint is enforced so only a few neurons can be active at any given time which forces the model to learn meaningful concepts.

the latent autoencoder representation. The hidden dimension n is m-times bigger than the original dimension d. W_{enc} are the encoder weights and $b_{dec} \in \mathbb{R}^d$ are the $b_{enc} \in \mathbb{R}^n$ biases.

The decoder is:

$$\check{x} = f_{dec}(z) = (W_{dec} \cdot z + b_{dec}) \tag{5.2}$$

Here $\check{x}$ is input reconstructed through the decoding matrix $W_{dec} \in \mathbb{R}^{n \times d}$.

The loss function used during training is a combination of the reconstruction error $\mathcal{L}_{MSE}$ and an additional sparsity constraint $\mathcal{L}_{L_1}$:

$$\mathcal{L}(x) = \mathcal{L}_{MSE} + \mathcal{L}_{L_1} = \sum_d (x_d - \check{x}_d)^2 + \lambda \sum_n z_n \tag{5.3}$$

Arbitrarily small latents could satisfy the L_1 constraint without having to be sparse. To avoid this from happening, we normalize W_{dec} to be unit norm periodically during the training, as in [110].

Several improvements have been proposed for training sparse autoencoders, both in terms of sparsity contraints (top-k proposed by [59]) and additional terms to the loss function (KL divergence proposed by [21]), but these will be left for future work.

5.2.1 Evaluation Metrics

There are several hyperparameters that we need to decide on while training a sparse autoencoder. For a simple one, the main hyperparameters are the learning rate, the size of the hidden layer and the sparsity penalty λ. To decide which sparse autoencoder architecture and combination of hyperparameters

gives the best results, ideally we would have to know the ground truth directions that we have to learn. Instead, we rely on a combination of sparsity and reconstruction error to measure the quality of the trained SAE:

- *Sparsity*: We measure the sparsity of our autoencoder as L_0 of z, the average number of non-zero latents for a single token.
- *Reconstruction error*: This measures how good the SAE is at reconstructing the activations of the original model. We can use the mean squared error for this. Usually, instead, we use a metric based on the cross entropy (performance) of the model, which we call the cross entropy increase Δ_{CE}.

  $$\Delta_{CE} = CE_{original} - CE_{reconstructed} \tag{5.4}$$

  This is the difference in cross entropy of the model with the original activations vs the model with the reconstructed ones and indicates how much of the model's performance the SAE can explain.
- *Number of dead latents*: This metric is generally used for SAE quality. It is defined as the number of latents that are never active (non-zero) over a large number of tokens (here we take $\sim 10^5$ tokens).

We will choose a SAE that reaches a good compromise between sparsity and reconstruction error, while keeping the number of dead latents as an additional quality check.

5.3 Interpretability and Steering

5.3.1 Interpreting "Neurons"

Several methods have been proposed to interpret the parameters of a neural network. In vision models, some works have proposed directly optimizing a synthetic image that maximizes the activation value of the "neuron" that we want to interpret [92]. A simple approach is to collect example images from our dataset that activate that particular "neuron", as done by [91]. Then, we can try to figure out what these examples have in common, for example, by having a human look at them and annotate them with a possible explanation. Another approach is to create manually curated datasets for certain labels (python code, french text, politician names), and then try to find neurons in the model that activate when the label is present by training a simple logistic regressor [67]. However, besides being time consuming, this approach requires constructing positive and negative examples in the right way, so that we could distinguish a "politician name" neuron from a general "person name" or "political" neuron [67].

To automatically annotate a large number of "neurons" in a language model, other researchers have suggested using large language models themselves [17]. Here we give the model a few examples of text that activates the neuron and their corresponding activations and ask it to give a brief explanation on what the neuron is detecting. By conditioning the model on this explanation and asking it to score a new example with a "predicted activation", we can also measure the quality of the explanations by comparing the predicted and real activations.

In Protein Language Models

For the case of protein language models, InterProt [5] adopts an approach where they build a visualizer for top activating proteins with their sequence and structure, and then try to figure out by manual inspection if there are any obvious patterns. InterPLM [127] uses different methods, including taking protein annotations from UniProt as labels and looking for neurons that have a good performance on them as binary classifiers; as well as using textual protein descriptions to give automated interpretations based on a large language model.

5.3.2 Steering

We can use SAE latents that correspond to specific features to steer or guide the model towards generating output with those features, or removing them from the output. This has been demonstrated for natural language [134] as well as image generation [77].

The approach we use in this work is outlined in figure 5.4.

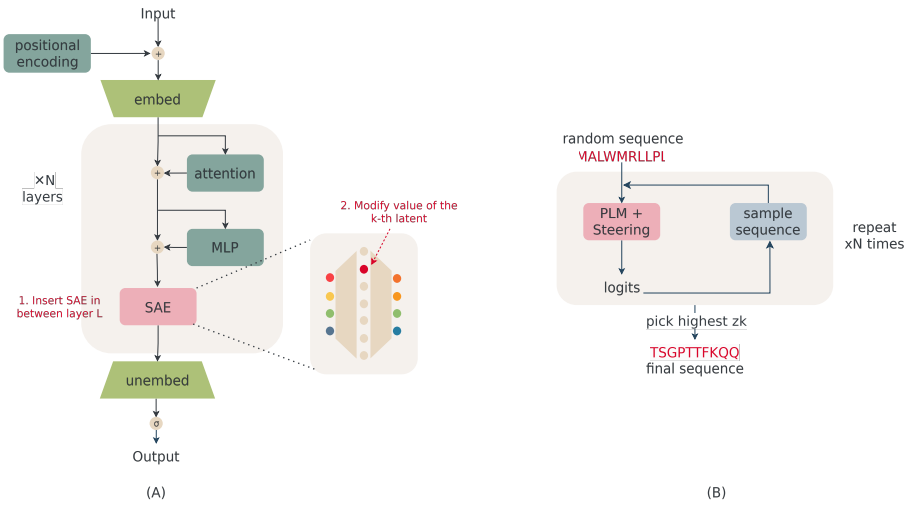


Figure 5.4: Procedure for steering the outputs of a Protein Language Model (PLM). (A) We introduce the trained sparse autoencoder in between one of the original transformer layers and modify the value of a target latent component during inference. (B) Starting from a random sequence, we perform inference with the modified model and sample an output sequence. We repeat this process iteratively and select a final sequence that maximizes the target component z_k.

We start with a randomly generated sequence of tokens of fixed length. We input the sequence through the model and the encoder layer of the SAE, and extract the latent representation of the sequence z. Given that the k-th component of this representation corresponds to our target feature, we modify z_k by scaling and shifting its value to increase its magnitude, as in (5.5). We then pass the modified value z_k^* through the decoder layer f_{dec} and add back

the original reconstruction error of the embedding x before passing it through the rest of the model (5.6) to obtain our final output probabilities.

$$z_k^* = a \cdot z_k + b \tag{5.5}$$

$$x^* = f_{dec}(z_k^*) + x_{err} \tag{5.6}$$

For each position in the sequence, we randomly sample a token according to the probability distribution predicted by the model under the intervention outlined previously and in 5.4 (A). We then repeat this process iteratively (as outlined in 5.4 (B)), starting from the predicted sequence and performing 100 repetitions of inference-prediction to refine the sequence. We select the sequence at the iteration where the value of the target latent z_k is maximum.

5.4 Sparse Autoencoders on Protein Language Models

In this section, we describe the details of our work training a SAE for interpretability of a protein language model and guiding sequence generation.

5.4.1 Training Details

We train our SAE on embeddings of the smallest model of the ESM2 family (ESM2 8M) [81], which has a hidden dimension of size 320, and extract activations from the final output of the transformer block. Our SAE consists of the simplest single layer architecture described in section 5.2, with the simple L_1 sparsity constraint and a ReLU activation function. The model is written in PyTorch [100] and trained on an NVIDIA A100 GPU with 40 GB of RAM. The time to train and evaluate each model was 30 minutes. The training dataset, hyperparameter and layer selection are described in the following sections.

Training Dataset

We train our model using the Astral SCOPe 2.08 dataset, filtered to 40% sequence identity, which includes approximately 15k highly non-redundant protein sequences [27]. This dataset contains a manageable number of tokens, which allows us to perform model training faster and iterate over different hyperparameters; but it still spans a wide range of protein structural domains. This diversity allows the autoencoder to learn a broad set of features.

Layer Selection

We adopt a principled strategy to select the layer from which we extract representations for the sparse autoencoder. The initial intuition, taken from previous works, [134, 59], is to choose a layer towards the middle/end of the model. We assume that at this stage the model has already developed features that are abstract enough but are not yet focused on the output reconstruction task. However, unlike these prior works, we move beyond mere intuition by incorporating a quantitative measure based on intrinsic dimension.

Specifically, we select one layer in the plateau of the intrinsic dimension curve, as described in section 4.1. Selecting a layer within this plateau, which provides more compact semantically-rich representations, increases the likelihood that our SAE finds meaningful abstract features.

Hyperparameter Selection

We use the Adam optimizer [78] with default parameters ($\beta_1 = 0.9, \beta_2 = 0.999$) for the training. We use the Kaiming random initialization [70] for the weights of the network.

We perform a hyperparameter sweep with the following values: learning rate ($5e^{-4}$, $1e^{-4}$, $1e^{-3}$); λ penalty ($0.0003, 0.001, 0.005$) and dictionary size multiplier ($32, 10, 5$).

We use the evaluation metrics detailed in section 5.2.1 to decide on the best combination of these hyperparameters. Specifically, we aim to find a model that balances reconstruction error and sparsity by focusing on the "elbow" part of the CE increase against L_0 plot, where increasing the density of active latents does not significantly reduce the reconstruction error, indicating an optimal trade-off.

Dead Neuron Handling

We check how frequently each of the latents activates over a subsample of tokens (50 batches of 4096 tokens) at regular intervals during training (every 500 batches). When this frequency is close to zero ($< 10^{-5}$), we consider that the latent is "dead" and we randomly re-initialize its weights to "revive" it.

5.4.2 Interpretability and Steering

Interpretability

In this work, we follow a similar strategy as [127]. We extract several different protein annotations from the UniProt database (described in section 3.3.2). We convert the amino-acid-level annotations into binary sequences matching the length of the corresponding protein sequences and use them to detect SAE latents that act as detectors for those annotations.

Mathematically, let A be the set of all amino acids and A_{ϕ^+} the set of amino acids that have been annotated with the feature ϕ. Considering a SAE latent k to be active (k^+) for a given amino acid when its value z_k exceeds a certain threshold τ_z, we have:

$$\pi = P(\phi^+|k^+) = \frac{\left|\{a \in A_{\phi^+} : z_k > \tau_z\}\right|}{|\{a \in A : z_k > \tau_z\}|} \tag{5.7}$$

$$\rho = P(k^+|\phi^+) = \frac{\left|\{a \in A_{\phi^+} : z_k > \tau_z\}\right|}{\left|A_{\phi^+}\right|} \tag{5.8}$$

This gives us a value of precision and recall for each pair of k, ϕ. We consider a latent component to be associated with a specific feature if its precision π or recall ρ exceed a predefined threshold that we set to 0.80.

In the case of protein-level annotations, we use mean pooling over the latent SAE representations.

Steering

We use the methodology described in 5.3.2 for steering the model during inference. We test the sequence generation procedure with SAE latents that were selected to have a clear association with zinc finger region annotations (high recall). We probe different values of starting sequence length ($[22, 27, 30, 35, 40, 60]$), scaling value a ($[2, 5, 10, 20, 30]$) and shift b ($[0.1, 1, 10, 50, 100, 200]$), for a total of 180 combinations, obtaining one sequence for each. We use the automatic zinc finger detection tool by [119] to check the resulting sequences.

5.4.3 Results

Model Training and Selection

Training and evaluation of a sparse autoencoder for the ESM2-6M model took around 30 minutes on an A100 nvdia GPU with 40GB of memory. The model is selected to have a good trade-off between its reconstruction error and sparsity level, as shown in figure 5.6, with hyperparameters $lr = 5e^{-4}$, $\lambda = 0.001$, $m = 10$. The loss curves for the training of this autoencoder are shown in figure 5.5.

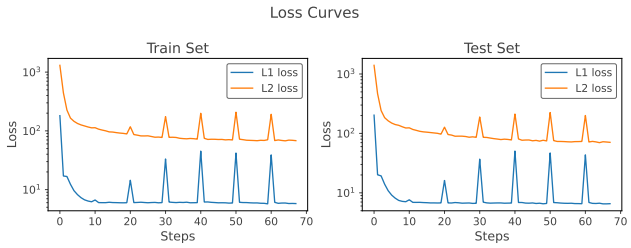


Figure 5.5: Loss curves for the training of the selected sparse autoencoder with hyperparameters $lr = 5e^{-4}$, $\lambda = 0.001$, $m = 10$, on a train set and a test set. We show two types of losses: a loss based on the $L1$ norm which represents sparsity and the $L2$ loss which represents the reconstruction error. The periodic peaks in the loss happen due to our dead neuron handling where we reset some of the parameters (details in section 5.4.1 in the main text).

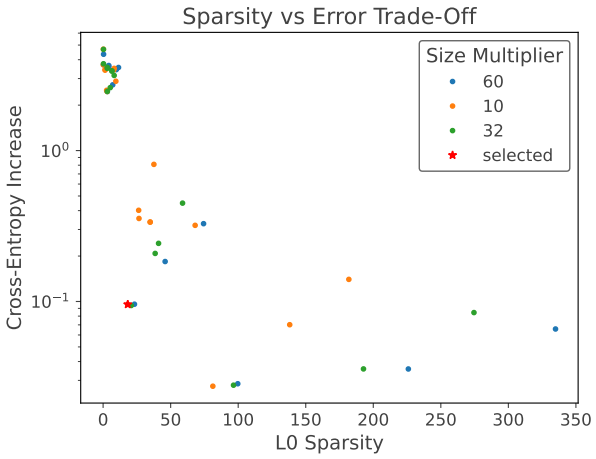


Figure 5.6: Trade-off between the sparsity and reconstruction error of sparse autoencoders with different hyperparameters. The dictionary size multiplier is indicated in different colors, but the models have also varying learning rates and sparsity coefficients. We select a model with a low reconstruction error and a reasonable level of sparsity, shown as a red star. This model had a learning rate of $5e^{-4}$, a sparsity coefficient λ of 0.001 and a dictionary size multiplier of 10.

Interpretability

For the interpretability analysis, we focus on the autoencoder that provides the best trade-off between sparsity and reconstruction quality. We compute recall and precision for all latent-feature pairs, using three increasing thresholds of latent activation. This allows us to assess the robustness of the identified features.

We identify 395 putative latent–feature associations. Among these, several latent components are associated with distinct binding sites, cellular regions, and motifs such as zinc fingers. We also identify many less stronger potential associations (lower values of precision or recall).

Intuitively, a latent component that perfectly matches an annotation type should exhibit both high precision and recall, resulting in a high F1-score. However, the model is trained in an unsupervised way without access to any of these annotations; so the features that it learns do not necessarily correspond to those in our manually curated dataset. We expect the model to learn some of these features anyways as they are relevant protein characteristics, and protein language models are successful at a lot of protein downstream tasks. For instance, a latent k might encode a more specific subcategory of a dataset feature ϕ, such as identifying the starting amino acid of a helix rather than the entire helix structure (as seen by [5] on some features). In such cases, the association between k and ϕ would likely have high precision but low recall.

Similarly, a high recall but low precision may indicate that the model has learned a more coarse-grained feature than those defined in the dataset. This is evident in cases such as latent $k = 610$, which activates across various types

of zinc fingers. We also find a latent that activates on zinc fingers, disulfide bonds and different binding sites (such as iron-sulfur cluster binding sites).

At a protein-wide level, we find latents associated to different protein characteristics. For example, we find latents related to location: they activate in internal (cytoplasmic, intravirion or in the mitochondrial matrix) or in extracellular/periplasmic proteins. We also find a latent that seems to activate only for eukaryotic organisms, which are a minority in the dataset used for the sparse autoencoder training.

To avoid selecting latents with spuriously high recall (such as those activating indiscriminately across all amino acids) we also evaluate the proportion of times a latent is active on amino acids lacking a given label, denoted as $P(k^+|\phi^-)$, before confirming an association.

Steered Sequences

We test the model steering procedure described in previous sections and activate sparse autoencoder latents that are associated with the presence of zinc finger motifs, to try to guide the model towards generating sequences that contain the motif.

We generate 180 sequences, as detailed in section 5.4.2. Using an online tool for automatic annotation of zinc finger regions [119], we look for known zinc finger motifs and Pfam family matches in our generated sequences. We find 24 matching regions (out of 180 sequences) when we simultaneously intervene on the two most prominent latents (high recall) associated to zinc finger motifs. In contrast, intervening only on the most prominent latent produces 3 matches, while generating sequences from the baseline model or intervening on a random latent or a random pair of latents produces no matches. Among the matched sequences, the highest percent similarity was 48%, with an average sequence similarity of 31%, indicating a good level of diversity among the generated sequences.

We also predicted the structures of the generated protein sequences using the ESMFold model for visual inspection. Some examples are shown in 5.7. As it can be seen, this structures show a patterns typical of a zinc finger motif: a beta hairpin in the vicinity of an alpha helix.

While this sequence generation pipeline requires carefully choosing the values of the a, b and the initial sequence length to improve the success rate, the process can be automated by introducing appropriate heuristics to search the parameter space (i.e. by performing a grid search with several combinations of these parameters).

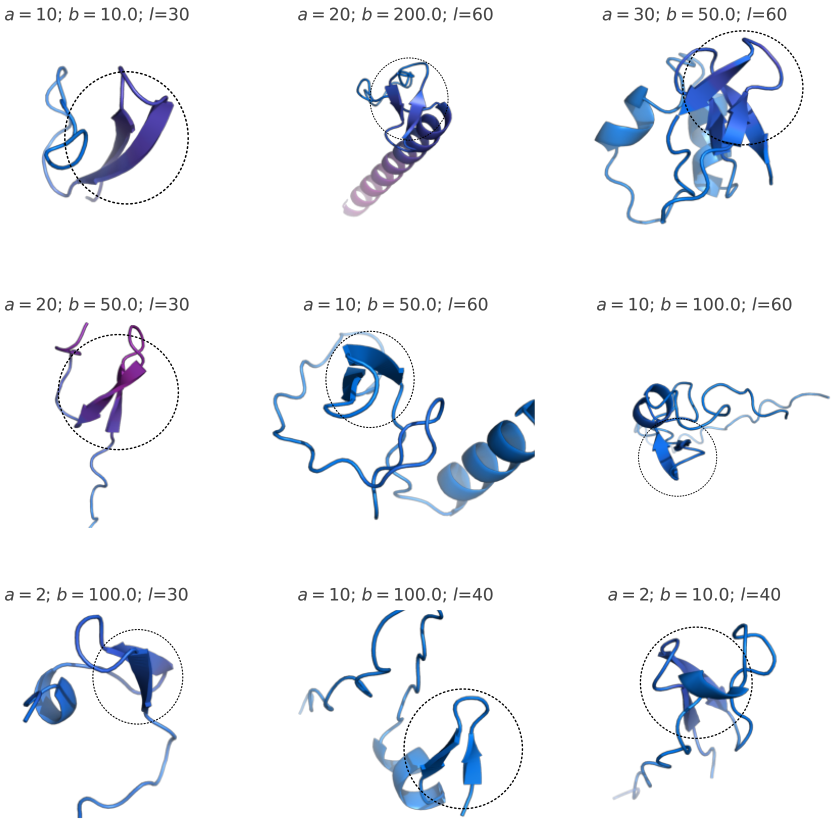


Figure 5.7: Examples of sequences generated with the sparse autoencoder steering procedure, using different parameters and sequence length and activating latents associated with detecting zinc finger structures. The structure of the sequence that is shown here was predicted with the ESMFold model. With the intervention, the model tends to generate sequences exhibiting structural elements characteristic of zinc fingers: β-hairpins in proximity to an α-helix. The β-hairpin like structures are highlighted with dashed circles.

5.4.4 Conclusions

In this study, we demonstrate the potential of sparse autoencoders (SAEs) for interpreting and manipulating the internal representations of protein language models. By training a SAE on the ESM2-8M parameter model, we identified and interpreted latent components associated with various protein annotations, including transmembrane regions, binding sites, and zinc finger motifs.

We have also demonstrated that these latent components can be used to steer the model towards generating amino acid sequences containing specific structural patterns like zinc finger motifs. These results highlight how sparse autoencoders can be useful in studying the representations of protein language

models, with promising potential applications for protein design and engineering.

Future work can expand on this by training sparse autoencoders on the representations of more modern protein language models like ESM3, expanding the protein training dataset and incorporating recent improvements in the sparse autoencoder architecture and training. Additional tests are required to see how well this methodology extends to other protein motifs and domains.

Chapter 6

Interpretability of Model Neurons

Abstract

This chapter explores the interpretability of protein language models through two complementary analyses: sparse probing for protein domain recognition and the characterization of outlier dimensions. The layers exhibiting peak performance coincide with the local minimum of the intrinsic dimension curve, reinforcing the notion that these intermediate representations capture the most evolutionarily informative features and are best for downstream tasks. Using embeddings from ESM2-3B, we show that information relevant to specific protein domains (such as the death domain PF00531), although not encoded by a single model neuron, is concentrated in a small subset of neurons, with classifier performance saturating rapidly as the number of input neurons increases. This is described in more detail in section 6.1. We further demonstrate that protein language models exhibit outlier dimensions analogous to those described in natural language models. A single outlier component shows activation patterns strongly associated with intrinsically disordered protein regions, suggesting it acts akin to punctuation in natural language models. This is detailed in section 6.2.

6.1 Sparse Probing for Protein Domain Recognition

We have seen previously that individual neurons in language models are hard to interpret: the models tend to encode concepts using several different components at the same time. However, how many components are needed to encode a certain concept is still an open question. The model could use a few select components or encode using the whole population of neurons. Here, we follow the methodology in [67] to do sparse probing and see how many neurons are needed to clearly detect a specific label in the data (in this case, a specific protein domain).

We find that although adding input neurons to a classifier consistently increases performance, this will quickly saturate with only a few neurons: it seems that most of the information is contained in a few components. Additionally, and in line with observations in previous works ([139]), we see that this performance peaks in the same layers where we find the local minimum of the intrinsic dimension, further supporting the claim that this is where the model most efficiently encodes evolutionary information about the proteins.

6.1.1 Methods

Datasets

For this experiment we take a very simple dataset: we extract proteins from the Interpro database that contain one specific protein domain: PF00531, also called the death domain, which is present in apoptosis and inflammation regulation pathways. The structure of this domain consists mainly of a bundle of six alpha helices and is described in more detail in [97].

For this analysis, we extract embeddings from all the layers of ESM2-3B for all the amino acids in these proteins. We also assign a binary label to each amino acid depending on whether it is part of a death domain region (1) or not (0).

Sparse Probing

We train logistic regression models to classify amino acids included in the PF00531 domain in two stages. First, we use the whole embedding (all components/neurons in one layer) as input to a regressor, and we extract the absolute value of the coefficients for each component as a measure of their relative importance for the prediction.

On a second stage, we train k-sparse regressors, we only take the first k most important components/neurons (according to the previous classification) and train a logistic regressor model with this data. We restrict this analysis to the 50 most important components in each layer due to computational costs. This is also the reason we avoid doing a stepwise selection, which would provide a more accurate ordering of the importance.

We use the F1 score as a measure of performance for all the classifiers, and observe trends with varying sparsity (k) and across layers in the model.

The models are trained on labeled amino acids from protein P98150 and tested on amino acids from protein O70510, which both contain PF00531 but have different architectures (different combinations of domains).

6.1.2 Results

With a simple logistic regression model using the protein representations from ESM2-3B we are able to predict which amino acids are part of a region containing the PF00531 domain with almost perfect accuracy.

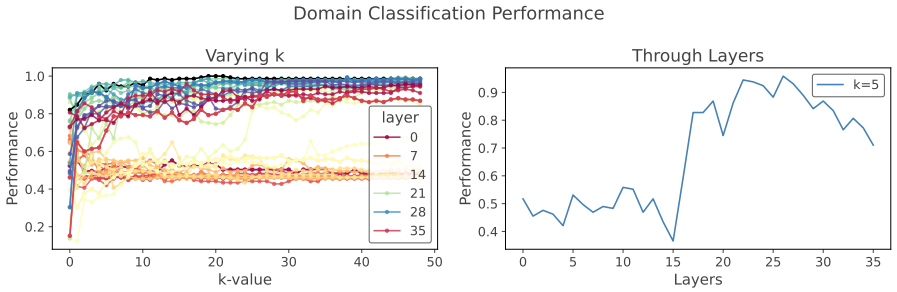


Figure 6.1: Classification performance (accuracy) for PF00531 prediction with logistic regression. The models are trained on different layers of ESM2-3B, using an increasing number of neurons k. *Varying k*: This plot shows the classification accuracy for each layer (different colors) with increasing number of neurons. We mark in black the curve corresponding to layer 24, which has a good performance already for small k and is located in the local minimum region of the intrinsic dimension evolution. We see that most of the information for this task is contained in a few most relevant neurons. *Through Layers*: The classification accuracy using 5 neurons as input for the logistic regressor, shown through the layers of ESM2-3B. The accuracy peaks in the region of the intrinsic dimension minimum, as shown previously for a homology task by [139].

Afterwards, we observe how this performance changes when varying the number of neurons used for the classification and through the layers of the model. This is shown in figure 6.1. We see that although the performance increases consistently as more neurons are added, each neuron gives diminishing returns. Most of the information seems to be concentrated in the first most important neurons, and performance saturates pretty early as well. Another phenomenon that we note is that early layers give relatively low performance at this task, and the performance peaks around layer 24 (different layers for different values of k - number of neurons).

We also plot the performance across layers for a value of $k = 5$, and we note that the evolution seems to be divided into two main phases: in early layers the performance is low, and it suddenly peaks in intermediate layers before slightly decreasing in the last layers. We observe that these two phases occur in the same layers that mark the two phases of intrinsic dimension evolution in protein language models, and peak performance seems to occur at the location of the minimum. This seems to be consistent with the observation by [139], that evolutionary relationships between proteins are easier to identify at the intrinsic dimension minimum. In this work we are additionally able to to note that this evolutionary information is contained in a very small number of components and that this sparsity also peaks in the layers of the intrinsic dimension minimum. This points to a highly structured representation space that is organized by evolutionary representations in these layers.

We also plot the activation values of the three most important neurons (the neurons where the absolute value of the coefficients in the logistic regression is the largest) in layer 24 along the protein length and show them in figure

6.2. Here we can easily see how these components contain already enough information to visually classify which amino acids are part of the domain.

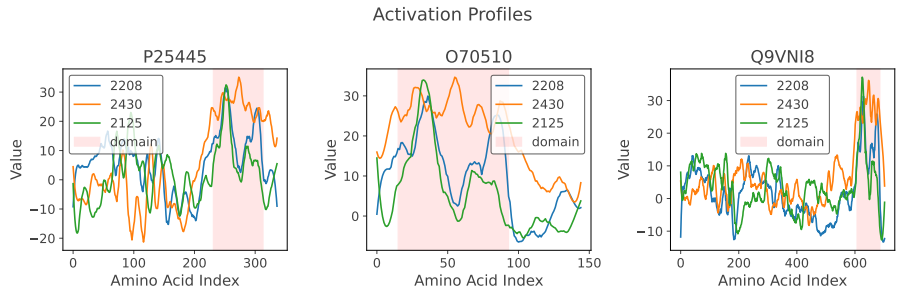


Figure 6.2: Neuron activation profiles for proteins containing the PF00531 domain. We plot the activation value of selected neurons through the protein length (amino acid index), for a subset of proteins that contain the PF00531 domain. The amino acids that span PF00531 are marked with a red coloring. We select the top 3 neurons that are the most important to predict the presence of the domain (in terms of the highest coefficients in a logistic regressor). We see that high values of the activation in these neurons are indicative of the presence of the domain.

6.2 Massive Activations and Outlier Dimensions

In this section, we demonstrate that the phenomenon of outlier dimensions, previously described in natural language models by [79], is also present in protein language models.

Outlier dimensions are components in the embeddings of natural language models that have activation values that are consistently much larger than the average component. They seem to play an important role in models, as deleting one of these components has a significant effect on model performance, unlike an average component [79].

Massive activations, on the other hand, are a related phenomenon where some dimensions in natural language models activate by orders of magnitude more for specific tokens: usually the token that marks the beginning of a sequence or other very common tokens [131]. They seem to act as bias terms in the model, and concentrate attention on the tokens that activate them.

Studying ESM2-3B, we find an outlier dimension in the model that seems to be related to intrinsically disordered protein regions. These regions are protein segments that lack a a well defined structure and are very common in proteins. Because of their flexibility, they can bind to multiple partners which gives them numerous functional roles [93]. We note that the component occurs at the same index across all layers in the model - this consistency is likely cause by the residual connections in the transformer.

6.2.1 Methods

Dataset

We extract a dataset of 3513 proteins with disorder annotations from MobiDB [103]. MobiDB contains annotations for disordered regions in proteins curated from several different sources. These annotations are aggregated in different ways. We selected the "Gold" level of evidence (annotations containing manually curated data and their homologues), and downloaded the dataset on March 2025.

For each amino acid in a protein in our dataset, we have a series of 0s or 1s indicating whether the amino acid is part of a disordered region or not. This label is a "merge" of all the different annotation sources included in the dataset.

We extract protein embeddings from all the layers of ESM2-3B for the analysis.

Existence of an Outlier Component

We analyze the average activation value of embedded amino acids in each of the components, as well as their standard deviation. We compare the value of our putative outlier component to the average value across all other components.

Relationship to Disorder

We perform a binary classification for each amino acid that indicates if it is contained in a disordered region or not. At a first stage, we perform this classification using the whole embedding to establish a baseline. Afterwards, we also perform this classification with each individual component, across all layers, and take note of where the performance is the maximal.

6.2.2 Results

To demonstrate the existence of an outlier dimension in the model, we analyze the average activation values across all components, which are shown in figure 6.3. We see that there is a component with a value that is consistently several orders of magnitude above the average value of other components in the model.

We also plot activation profiles for the outlier component for all the layers of ESM2-3B. An example is shown in figure 6.4. The average component through the whole sequence is several orders of magnitude above the normal value (which is around 0), consistent with the definition of an outlier dimension. Additionally, even though massive activations are thought of as separate phenomena in natural language models, this component also seems to show some characteristics of massive activations: the value of its activation is significantly higher for the beginning and end-of-sequence tokens. We also see that the behavior of this component seems to be particularly different in regions of the protein where there is no specific structural domain: there is more separation in the activation values through the last layers, and the frequency of variation from one token to another seems to be lower. This can be seen in figure 6.4.

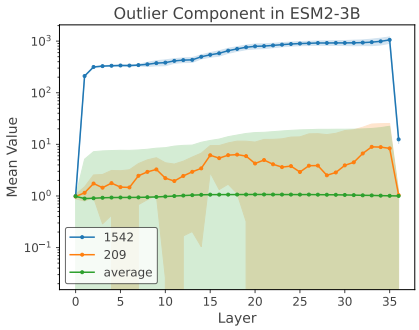


Figure 6.3: Plot illustrating the presence of an outlier component in ESM2-3B. We show the average value and standard deviation of the activation (in logarithmic scale) through the layers of the model for a dataset of several thousand proteins for: 1 - the outlier component 1542, 2 - a "normal" component 209, and 3 - the average of all components. Given that the average value for most components is close to zero, we add 1 to all values to be able to show them on this logarithmic scale.

As intrinsically disordered regions are present in a large percentage of proteins, it makes sense that protein language models could use these regions as anchors the same way the "." or "and" tokens can be used in natural language models. To test the hypothesis that component 1542 is somehow related to disordered regions, we train a few logistic regression probes on this data.

First, we use the whole embedding from ESM2-3B to classify amino acids from intrinsically disordered regions, and manage to achieve a decent level of performance in terms of F1 score (figure 6.5).

We then test the same logistic regression probe on each individual component of the embeddings (figure 6.6) and find that the outlier component 1542 reaches better performance than other components in recognizing intrinsically disordered regions. We also note that the performance of the outlier component in this task peaks at around layer 20, near the location of the intrinsic dimension minimum. This further supports the idea that the representations in this location are rich linearly separable information.

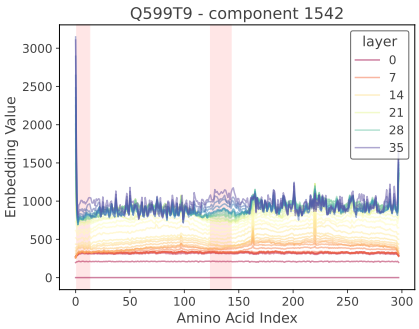


Figure 6.4: Activation values of the outlier component of ESM2-33B (1542) for a protein containing two disordered regions. The values are shown through the length of the protein (amino acid index) and the amino acids spanning the disordered regions are colored in red. The disorder annotation here is from Alphafold-RSA. Values in different layers are shown with different colors. We can see that the activation values in amino acids from disordered regions follow a particular evolution through the layers that is markedly different from the amino acids in the rest of the protein, particularly in later layers.

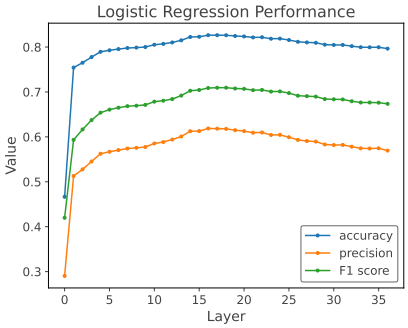


Figure 6.5: Logistic regression performance for predicting disorder annotations in proteins taken from MobiDB (accuracy, precision and F1 score). The regressor takes all components of a particular layer of ESM2-3B embeddings as input. The performance is shown through the layers. We see here that the performance increases dramatically in the first layers, but continues to increase and peaks between layers 15-20 (corresponding to the local minimum of intrinsic dimension evolution).

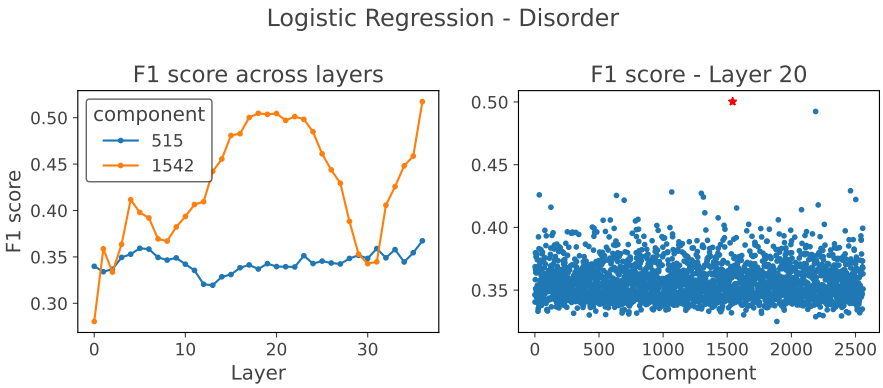


Figure 6.6: Logistic regression performance for predicting disorder annotations using individual components of ESM2-3B. *F1 score across layers*: we show the performance for an average component and for the outlier component. The outlier component reaches much better performances on its own, and peaks around layer 20 before going up again for the final layer. *F1 score - layer 20*: performance of each individual component for predicting disorder with embeddings from layer 20 of ESM2-3B. We mark the outlier component with a red star, and note that it is the one with the highest individual performance.

6.3 Conclusions

In this chapter, we study two phenomena related to interpretability of protein language models aided by two different classification tasks (protein domain and disordered region classification). We see that the performance on both of these tasks tends to peak in layers located at the local minimum region of the intrinsic dimension curve, further supporting the idea that representations from these layers are the most useful for downstream tasks.

Additionally, we show that the phenomenon of outlier dimensions is also present in protein language models, where they seem to be particularly related to disordered regions, which might play a role similar to punctuation in natural language models.

We also see that although “neurons” in protein language models don’t encode for single concepts such as protein families, most of the information related to these concepts seems to be concentrated on a few neurons in the model. Concepts in the model do not seem to be encoded through the whole population of neurons uniformly.

Chapter 7

Protein Interaction Interface Prediction

Abstract

This chapter presents a study of protein interface prediction using pre-trained protein language models. The work focuses on identifying the amino acid residues that constitute interaction interfaces, a fundamental problem in structural bioinformatics with implications for understanding protein function and guiding therapeutic design. To ensure the reliability of the evaluation, special attention was given to data curation, particularly to the prevention of homology-induced data leakage between training and test splits. We used representations from model in the ESM family for interface classification across multiple interaction types. The results demonstrate that finetuned ESM2 models substantially outperform previous classical feature and embedding based baselines on benchmark datasets, achieving improvements of up to 0.04 in the Matthews Correlation Coefficient for interface prediction. Using embeddings from ESM3 additionally increases performance without the need for finetuning. These findings highlight the capacity of large scale language models to construct sequence representations with biologically meaningful information, and the relevance of a diverse pre-training dataset including different data modalities (sequence, structure and function) for model learning.

7.1 Introduction

Protein–protein interactions (PPIs) lie at the heart of nearly every cellular process, from signal transduction and metabolic regulation to structural assembly and immune response. Alterations in PPIs (in particular, perturbations of the interacting residues) can lead to various diseases. As such, identifying the specific residues that mediate these interactions is not only of fundamental biological interest, but also of direct relevance in clinical diagnostics and drug design. The biochemical and structural characterization of PPI interfaces is a major task in computational structural biology [150].

At the atomic level, a PPI interface is defined by the set of amino acid residues on one protein that are in contact with a partner protein. These residues contribute to binding affinity, specificity, and functionality of the protein complex. From a therapeutic perspective, these interfaces themselves are ideal targets for drug design: specific molecules can be engineered to disrupt or modulate specific interactions. However, experimentally determining protein-protein interfaces is a very expensive and laborious process. This gap has motivated a rich body of computational work on predicting interface residues from protein sequence and structure [44, 152]. Computational methods can accelerate the annotation of interaction networks and provide hypotheses for experimental validation, which is especially important given the explosion of sequencing data.

Early efforts in computational prediction used hand crafted features such as surface accessibility, evolutionary conservation and physicochemical characteristics to identify these interaction interfaces [130, 12]. With advancements in artificial neural networks and deep learning, other methods were proposed to predict them by directly training an algorithm on protein sequences and interaction labels [30]. More recently, progress in unsupervised representation learning through language modeling presents a unique opportunity to approach this problem. Some researchers have already started efforts to predict interaction interfaces using the rich representations learned by protein language models [136]. However, critical challenges remain: for example, data leakage (very similar proteins in test and training sets) seems to be common in the literature when evaluating such models, which means their performance scores are unreliable.

In this chapter, we describe how we used pre-trained protein language models for predicting which amino acids are part of a protein interaction interface, taking particular precautions to prevent data-leakage from our training to our test datasets. It is organized as follows: in section 7.2 we describe in more detail the deep learning model and training used for the prediction (section 7.2.1) and the data curation methodology (section 7.2.1). The results are outlined in section 7.3.

7.2 Methods

7.2.1 Data Curation

For training the models described in this chapter, we used the BioDL dataset curated in [130],, described in more detail in the following section.

BioDL dataset

The BioDL dataset was constructed by [130] by taking protein structures from the PDB with more than 2.5 Å resolution. Residues were annotated as interacting when the distance between one of their atoms and any atom of the interacting molecule was less than the sum of their Van der Waals radii plus 0.5 Å. Using BLASTClust, sequences with more than 25% similarity were removed, and the dataset was split into 95% for the training set and 5% for the test set.

The BioDL dataset contains interactions of proteins with nucleotides (n), small molecules (s) and other proteins (p).

Models

We finetune models from the ESM2 family for the protein interaction interface classification task. We use a pre-trained protein language model as our base model and add an additional linear layer before binary classification. We use the binary cross entropy loss for the training. The optimizer used was Adam with weight decay for regularization and default parameters (betas: 0.9, 0.999). We also perform the training using the ESM3-open model, but in this case, we freeze all model layers and train only the last layer (equivalent to extracting the embeddings from the model).

We perform hyperparameter sweeps with learning rates: ($5e^{-5}, 1e^{-4}$), weight decay: (0.01, 0.05, 0.1) and batch size (2) with gradient accumulation over (1, 4, 8, 16) steps. We perform the training over 8 NVIDA A100 GPUs with 80 GB of memory each. The effective batch size is: batch_size $\times$ gradient_accumulation $\times$ number_devices. The training time for the biggest model was around 30 minutes.

We train the models on the BioDL datasets for interactions with nucleotides, small molecules and proteins, and additionally test the most performing model on the ZK448 dataset from [152].

We measure the performance of the models with two main metrics that are less sensitive to the imbalance in the labels of the dataset (many more negatives than positives): the F1 score and the Matthews Correlation Coefficient (MCC) score, which are both standard metrics for this task.

7.3 Results

Following the described methodology, we finetune all models in the ESM2 family up to the 3B size. The performance with increasing model size is shown in figure 7.1, for models trained with a standard set of hyperparameters. We note how, as expected, a larger starting size for the base model results in a higher MCC score, given that they have learned a richer protein representation and have more expressive capacity (parameters) to tune.

The performance of the best models for each type of classification task (protein-nucleotide: n, protein-small molecule: s and protein-protein interaction: p) are shown in table 7.1 and compared to the best performing models from the PIPENN family [130, 136]. These consist of an ensemble of neural networks trained on the BioDL dataset for each specific task, starting from classical features. For protein-protein interactions, there is also a model trained on the embeddings of another protein language model, namely ProtT5-XL. In previous tests, [136] saw that ProtT5-X embeddings worked better than ESM2. With the finetuning of ESM2, we achieve a higher score than theirs by more than 0.04 points, and 0.05 using the embeddings from the ESM3 model.

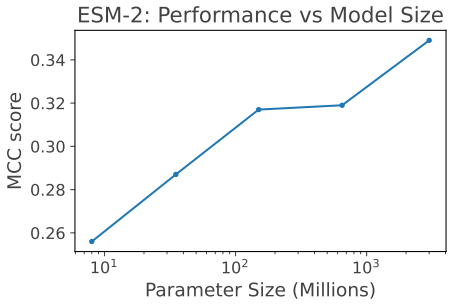


Figure 7.1: Evolution of the performance on the protein-protein interface classification task (only proteins), given by the Matthews Correlation Coefficient with different sizes of the starting ESM-2 model. All models were finetuned with the following hyperparameters: (learning rate: $5e^{-5}$, weight decay: 0.01, effective batch size: $2 \times 8 \times 8$). Bigger starting models produce a higher final performance (notice the log scale on the plot).

Table 7.1: Performance of different models on various data modalities for the BioDL [130] dataset. The models trained in this work are marked with *.

Model Name	**F1**	**MCC**
Nucleotide		
PIPENN ensnet_n [130]	0.42	0.38
*esm2_t36_3B	**0.59**	**0.57**
Small Molecule		
PIPENN ensnet_s [130]	0.41	0.38
*esm2_t36_3B	**0.55**	**0.53**
Protein		
PIPENN ensnet_p [130]	0.34	0.25
PIPENN ensnet + EMB [136]	0.40	0.31
*esm2_t36_3B	**0.44**	**0.35**
*esm3-open	**0.44**	**0.36**

For comparison with other models and scores previously reported in the literature, we also evaluate the model on the ZK448 dataset [152], these comparisons are shown in table 7.2. With ESM2, we obtain a better performance score than most models, starting only from the sequence information, except for the Seq-InSite model [72]. However, the performance of this model significantly decreases when the level of homology between its training set and the test set decreases, as shown by [136]. The ESM3 based model further improves performance beyond what we obtain with models trained without structural data.

Table 7.2: Comparison of different methods for protein-protein interface prediction on the ZK448 [152] dataset. The models trained in this work are marked with *.

Method	**F1**	**MCC**
Seq-InSite46	0.54	0.46
PIPENN-EMB	0.51	0.39
EnsemPPIS35	0.39	0.29
*esm2_t36_3B	0.54	0.42
*esm3-open	0.61	0.52

7.4 Conclusions

In this chapter, we investigated the use of pre-trained protein language models for predicting residues involved in protein interaction interfaces. By carefully constructing the training and evaluation datasets to avoid homologous data leakage, we obtained a robust assessment of model generalization. Finetuning ESM2 models led to consistent improvements across interaction types, with the largest model achieving the best performance. The observed scaling trend suggests that larger representations encode progressively richer biochemical and structural information.

Compared to traditional feature based methods and previous embedding based models, the finetuned ESM2 architecture achieved higher F1 and MCC scores on both the BioDL and ZK448 datasets. These results demonstrate that sequence based models can provide accurate residue level predictions of interaction sites.

Additionally, using ESM3 embeddings further increases the performance without the need for finetuning. This is likely due to the fact that the ESM3 model has seen both sequence and structural information during pre-training, and this gives it an advantage over the EM2 model for this task. This holds true even though we are only using the protein sequence as input for both ESM2 and ESM3, without having to include any additional structural information during inference.

Chapter 8

Risk Stratification via Unsupervised Learning

Abstract

In this chapter, we describe how we applied unsupervised learning techniques (clustering) to clinical data from patients with a specific type of leukemia and found that the resulting groups corresponded to different risk categories in terms of treatment-free survival of the disease. We describe this type of leukemia (B-cell Chronic Lymphocytic Leukemia), as well as the clinical problem, in more detail in section 8.1.1. The methodology for our analysis and the results are detailed in sections 8.2 and 8.3, respectively. We provide additional background on survival data analysis in section 8.1.2, and on the type of clinical data used in this study in section 8.1.1.

8.1 Background

8.1.1 B-Cell Chronic Lymphocytic Leukemia

B-cell Chronic Lymphocytic Leukemia (BCLL) is a type of cancer affecting the white blood cells responsible for secreting antibodies, the B-cells. It is the most common type of leukemia in adults in Western countries, with an incidence of 4.1 cases per 100,000 inhabitants [68].

This type of leukemia is chronic because the disease progresses slowly over several months or years, and many patients are asymptomatic. Treatment is only required when patients start showing symptoms (anemia, swollen lymph nodes, etc.) or for advanced stages of the disease [68]. The time from diagnosis to first treatment (treatment-free survival time) is our main target variable here.

Patients diagnosed with BCLL are clinically heterogeneous in terms of disease progression and prognosis. Specific biomarkers (measurable biological quantities) for this disease help provide more personalized care to each patient. Some biomarkers of known diagnostic and prognostic value in BCLL are spe-

cific gene mutations, chromosomal aberrations, and flow cytometry variables described in the following section.

Selected Biomarkers

- ***Flow Cytometry Markers***: Flow cytometry is a technique used to analyze individual cells flowing through a small tube by using light. Antibodies with fluorescent dyes can be used to detect the presence of specific receptors on the cell surface. In this project, it is used to identify B-cells and detect the presence of receptors like CD49d (involved in cell adhesion) and CD38 (involved in B-cell activation), in terms of the percentage of cells presenting the marker [104].
- ***Fluorescence In-Situ Hybridization (FISH) markers***: This technique uses specific DNA sequences (probes) that are labeled with a fluorescent dye. The DNA probe binds to its complementary sequence in the cell and is used to detect its presence, absence, or duplication [32]. Here, we use it to look at deletions in chromosomes 17 and 11, as well as trisomy (repetition) of chromosome 12 (measured as the percentage of nuclei that present the abnormality).
- ***Genetic Mutations***: We look at genetic mutations in terms of Variant Allele Frequency: roughly the proportion of times a certain mutation appears during a DNA sequencing experiment. In this project, we mainly look at the TP53 gene, a known tumor suppressor gene [15].
- ***IGHV (Immunoglobulin Heavy Chain Variable region) mutation***: The IGHV region is a variable region in the gene that codes for antibodies in B-cells. This helps the antibody bind to different antigens (parts of biomolecules), and it undergoes *somatic hypermutation* in mature B-cells. Mature B-cells (i.e., those having a mutated IGHV) are associated with less aggressive forms of leukemia [68]. The % mutation is calculated as the % difference with respect to the germline.

8.1.2 Survival Analysis

Survival analysis deals with a specific type of data called *censored data*. Originally, it dealt with measuring the lifespan of individuals, but it can be used for analyzing any dataset where our target variable is the time it takes for a particular event to occur. This event could be death, but it could also be a patient undergoing a particular treatment or showing signs of a disease. At the time of the analysis, it is possible that not all events have happened yet: this is why we call it *censored data* — part of the information is missing. Our data will have two components: the total time duration and whether the event has happened or not.

We can explore several different aspects of our data. We can see how survival probability or hazard rates evolve through time in the population, or we can see how different variables improve or worsen these probabilities. Survival probabilities are usually estimated with the Kaplan-Meier estimator (section 8.1.2), while there are several different estimators for the hazard ratio. A simple

way of seeing how different variables affect survival outcomes is the Cox model described in section 8.1.2.

Kaplan-Meier Estimator

The survival probability at time t is defined as the probability that the death event has not yet occurred. This can be estimated with:

$$\hat{S}(t) = \prod_{t_i \leq t} \left(1 - \frac{d_i}{n_i}\right) \tag{8.1}$$

Where $\hat{S}(t)$ is the estimated survival probability at time t, d_i is the number of deaths at time t_i, and n_i is the number of individuals still at risk at time t_i [43].

Cox Proportional Hazards

The Cox Proportional Hazards model is the simplest regression model for survival data. It allows us to estimate how particular variables change the baseline hazard function of the population [114].

The hazard function $H(t)$ is defined as the probability of a death occurring at time t, and is related to the survival probability through:

$$S(t) = e^{(-H(t))} \tag{8.2}$$

The Cox model is fitted to output the hazard given a set of input variables $h(t \mid \mathbf{x})$:

$$h(t \mid \mathbf{x}) = h_0(t) \exp\left(\beta \cdot (\mathbf{x} - \bar{\mathbf{x}})\right) \tag{8.3}$$

Where $h_0(t)$ is the baseline hazard of the population, $\bar{\mathbf{x}}$ are the average input values of the population, and β are the coefficients for each variable.

8.1.3 Clustering

Clustering is a machine learning technique used to find structure in datasets and group together similar data points. Here, we briefly describe two clustering algorithms used for this project: k-means (a partition algorithm) and Advanced Density Peaks (DPA), a density-based clustering algorithm.

K-means

The k-means algorithm partitions the data into k different clusters by minimizing the Sum of Squared Errors (SSE) between data points and their assigned cluster centers (called centroids) [10]. Mathematically:

$$\arg\min_{C}; \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{8.4}$$

where C_i is the set of all data points x assigned to cluster i, and μ_i is its centroid (average x values).

Advanced Density Peaks

Advanced Density Peaks (DPA) is an improvement over the original density peaks clustering algorithm. In the original algorithm, cluster centers are found as data points that are both in regions of high density and far away from other points with high density. DPA automatically finds these cluster centers, given a specific level of statistical confidence Z [45]. The value of Z is a measure of how confident we are that a cluster actually exists, instead of being just a random fluctuation. Setting a minimum value of Z controls how many clusters the algorithm considers real (i.e. a higher value of Z means less clusters will meet the criteria).

8.2 Methods

8.2.1 Data Pre-processing

The dataset we used for our analysis consists of observations from peripheral blood samples of 746 patients diagnosed with B-cell leukemia from January 2003 to January 2020. These data come from the *Centro di Riferimento Oncologico di Aviano*, and the dataset has been described in detail by [34]. We applied several pre-processing steps before the analysis, which we detail in the following paragraphs. A visual overview of the dataset and the original variables is provided in Figure 8.1.

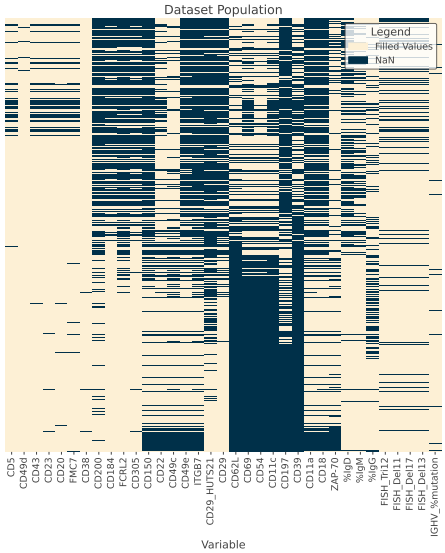


Figure 8.1: Population for each variable in the original dataset. The x-axis represents different columns in our dataset, while the y-axis represents different samples. We mark missing values in blue.

- ***Sample Selection***: We select the first sample for each patient in cases

where patients have multiple samples.

- ***Treatment-Free Survival Time***: To calculate the treatment-free survival time, we count the days between the blood sample and the date of first treatment, or the date of the last follow-up appointment for untreated patients. We exclude patients whose last follow-up appointment was less than 12 months after diagnosis.
- ***Missing Values***: We discard variables that have more than 25% of missing values.

8.2.2 Variable Selection

To identify the most clinically significant variables for predicting the treatment-free survival time, we use the Cox model. We fit a univariate Cox model for each of the variables under study and keep the ones with coefficients that have the smallest p-values.

8.2.3 Clustering and Survival Analysis

Clustering

After selecting our variables and samples, we perform k-means clustering on our dataset. We use the elbow method to select the number of clusters to partition our dataset and plot the results as a 2D projection using the UMAP algorithm [85].

To study the characteristics of each cluster in more detail, we look at the average values of each variable for each cluster (the centroids) and compare them across clusters.

Survival Analysis

We use the Kaplan-Meier estimator and plot the survival curves (treatment-free survival probabilities) for each of the resulting clusters. We combine the clusters into three risk groups (high, intermediate, and low risk) to make the classification more convenient to use in practice.

8.3 Results

8.3.1 Variable Selection

We calculate the proportion of missing values in the dataset and drop variables with fewer than 75% valid values. We show the results of the univariate Cox model fitting in Figure 8.2. We choose to select the variables with the smallest p-values before the gap, shown with the dashed red line. This selection results in the following variables: IGHV_%mutation (percentage of mutation of the IGHV gene); FISH_Del11, FISH_Del17, FISH_Tri12 (FISH: Fluorescence in Situ Hybridization variables detecting the presence of deletions in chromosome

11, chromosome 17, and trisomy in chromosome 12, indicated as a percentage of cells); CD49d, CD38 (flow cytometry variables indicating the presence of specific protein receptors in the cell membrane, indicated as a percentage of cells); and TP53 (VAF: Variant Allele Frequency of mutations in the TP53 gene).

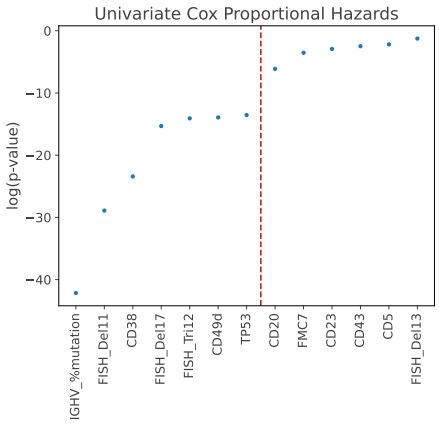


Figure 8.2: Plot of the p-values (on a log scale) of the coefficients in each univariate Cox proportional hazards model. The p-values are calculated with the lifelines package [36], using the *Wald* test. We select the lowest p-values.

8.3.2 Clustering and Survival Curves

Using the elbow method, we find an optimal value of 6 clusters for our dataset with the k-means clustering algorithm. This plot is shown in the appendix (figure B.1). We visualize the resulting clusters in a UMAP projection (figure 8.3) and plot the cluster centroids in figure 8.4. From this, we recognize that there are clusters with distinct clinical characteristics in terms of the selected variables. We plot the Kaplan–Meier survival probabilities for each cluster in Figure 8.5.

Clusters 4, 5, and 6 are each characterized by high values in one of the FISH variables (corresponding to trisomy of chromosome 12, deletion of chromosome 11, and deletion of chromosome 17). Cluster 6 is additionally characterized by a high presence of TP53 mutations, which tend to be correlated with chromosome 17 deletion [151]. All these are severe disruptions at the chromosomal level and are therefore associated with an overall lower value of the treatment-free survival probability.

Cluster 1 is marked by higher values of the IGHV_%mutation variable, which is associated with more specialized B-cells and a milder form of the disease [143], indicating very low risk. Cluster 2, on the other hand, is distinguished by low values of IGHV_%mutation as well as CD49d and is intermediate in terms of risk. Cluster 3 seems to be associated mainly with elevated values of CD49d and is also intermediate in terms of risk, but slightly worse than cluster 2.

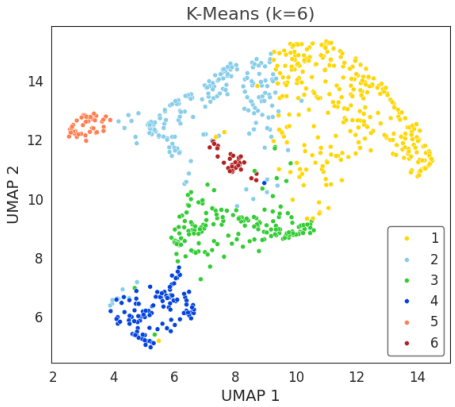


Figure 8.3: UMAP projection in 2D of the clustered dataset.

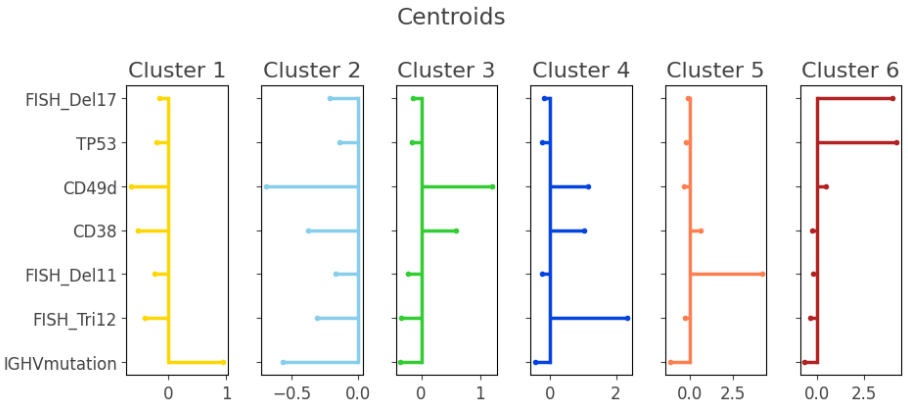


Figure 8.4: Cluster centroids (average values) for the k-means clustering of our dataset. Each cluster is characterized by a particular profile and seems to be associated with one or two particularly marked variables.

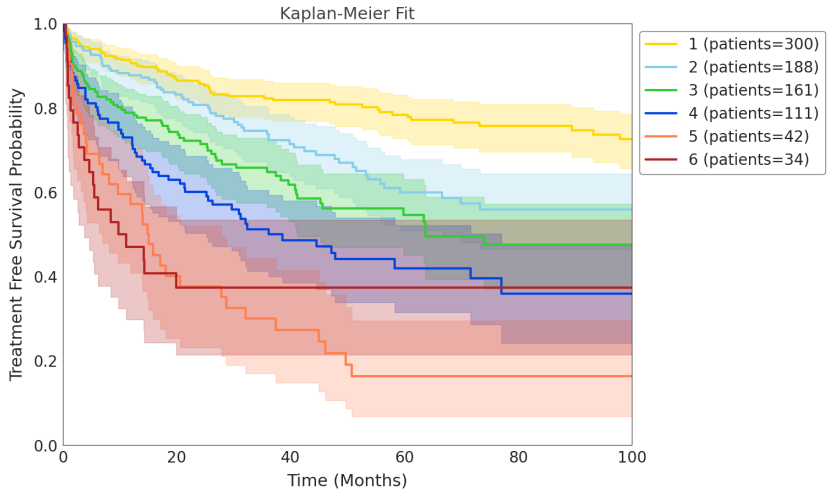


Figure 8.5: Treatment-free survival probabilities for patients in different clusters.

For convenience in clinical practice, we group the clusters into low (cluster 1), intermediate (2, 3, and 4), and high-risk groups (5 and 6). We show the survival curves for each group in Figure 8.6.

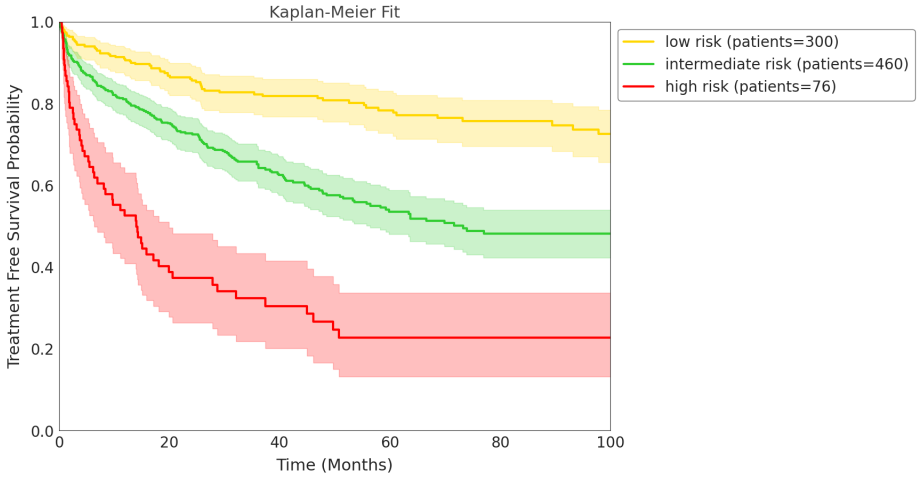


Figure 8.6: Treatment-free survival probabilities for patients in different risk groups.

We note that each of the clusters seems to be characterized by an extreme value in a particular variable: cluster 1 and 2 by IGHV_%mutation (high and low values), cluster 3 by CD49d, cluster 4 by FISH_Tri12, cluster 5 by FISH_Del11, and cluster 6 by FISH_Del17 and TP53. Guided by this finding, we devise a way of recreating the original clustering by applying simple thresholds to the data. This is useful for applying the results of the clustering in clinical practice.

This classification algorithm (described in Figure 8.7) allows us to recover the original clustering classification with 92% accuracy. To decide the values of the thresholds, we take the minimum value of the representative variable in each particular cluster (excluding outliers: values that are more than 40% away from the average value). We then rank them in order from highest to lowest to get our final algorithm. We show the distributions of each of the variables along with the selected thresholds in Figure 8.8.

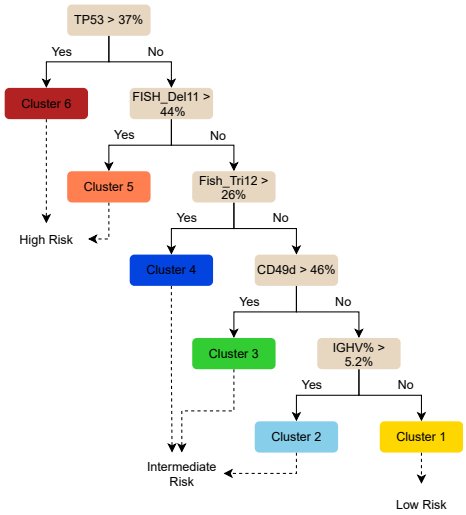


Figure 8.7: Decision tree algorithm to classify each new patient into one of the clusters and corresponding risk groups (high, intermediate, and low) described previously.

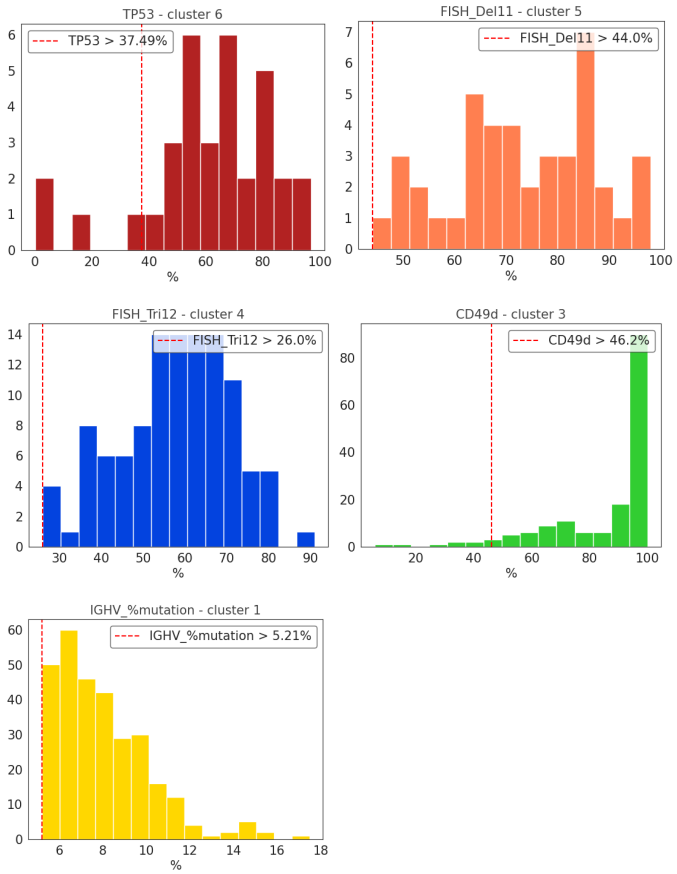


Figure 8.8: Histograms showing the distribution of value counts for different variables that characterize each cluster. We mark our selected thresholds with a dashed red line.

8.3.3 Advanced Density Peaks Clustering

For comparison, we perform the same clustering on our dataset using the DPA. We perform the clustering with several values of z and notice that there seems to be a region where the number of clusters remains stable, so we decide to keep a z value in that range (z from 0.75 to 1.25). This is shown in figure B.2 in the appendix. Clustering with $z = 1$ leads to the assignments shown in Figure 8.9. We also plot the centroids of each cluster in Figure 8.10.

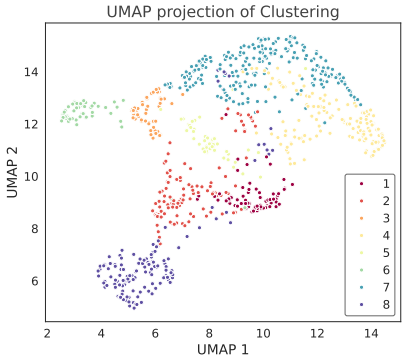


Figure 8.9: UMAP projection in 2D of the original dataset with the clustering assignments of Density Peaks Advanced.

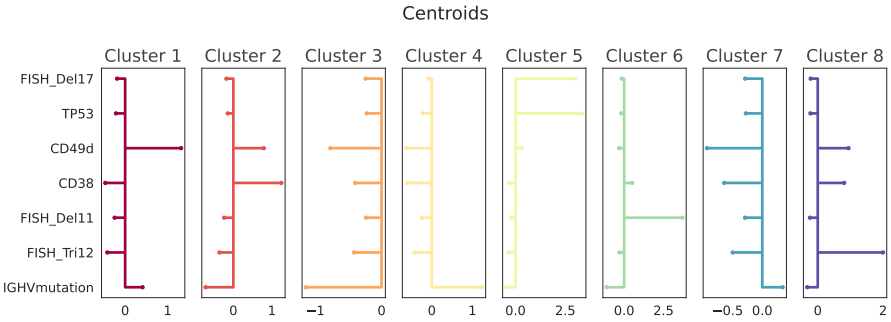


Figure 8.10: Centroids of the DPA clusters.

Looking at the centroid profiles and the UMAP, we see that the original clusters (OC) from k-means with the extreme FISH variables (FISH_Tri12, FISH_Del11, and TP53 & FISH_Del17) have an almost exact correspondence with some new clusters (NC) obtained from DPA. NC5 is associated with high values of TP53 and FISH_Del17, like OC6; NC6 with FISH_Del11, like OC5. These clusters are both associated with low survival probabilities. NC8 corresponds to OC4, with high FISH_Tri12 values.

OC3 splits into two smaller clusters, NC1 and NC2. Like OC3, NC2 has high CD49d and CD38 values and a low value of IGHV mutation. NC1, on the

other hand, has high CD49d but low CD38 and IGHV mutation, with a slightly better survival outcome than NC2.

NC3 seems to be a subset of OC2, with similar clinical profiles (low IGHV mutation, CD49d, and CD38). NC4 is a subset of OC1, mainly characterized by high values of IGHV mutation. NC7 lies at the boundary region between OC1 and OC2, with a profile similar to OC1. Both NC4 and NC7 have quite high survival rates, probably due to their high IGHV mutation values.

NC1 has intermediate survival, while NC3 and NC2 have low survival rates, which could be associated with their low IGHV mutation values.

We see that the DPA clustering shares some key similarities (the FISH-related clusters) and differences with the k-means clusters. In the end, we decided to keep our original k-means clustering, given that we were able to extract a simple algorithm that clinicians can use to manually classify new patients into distinct risk groups, as this was our main goal.

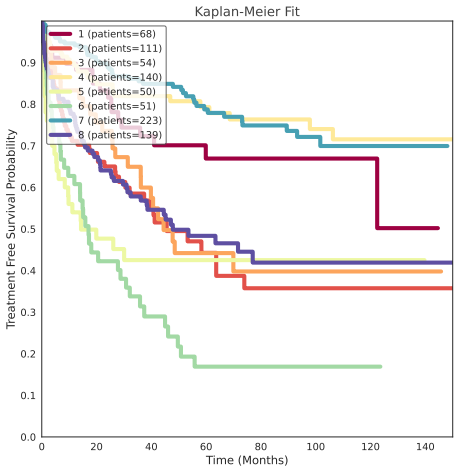


Figure 8.11: Kaplan–Meier survival curve estimates for the DPA clustering.

8.3.4 Conclusions

Using an unsupervised machine learning algorithm, we were able to extract clinical profiles (cluster centroid values) for our B-cell leukemia patients that correspond to distinct risk groups with low, intermediate, and high treatment-free survival probabilities. We devised a simple decision tree that can reproduce the original clustering with high accuracy and can be easily implemented to classify new patients in clinical practice. Our groupings reflect well-known information in the field, such as high values of chromosomal aberrations being predictive of a more aggressive form of the disease, and IGHV mutation being a strong prognostic marker. What sets our approach apart is that these clinical profiles emerge in an unsupervised manner, directly reflecting the inherent structure of the patient population.

Chapter 9

Conclusions

This thesis has presented a comprehensive and multi-layered investigation into the interpretability and practical utility of Protein Language Models (PLMs). The work spans from foundational analyses of the geometric structure of neural representations to advanced methodologies for disentangling and manipulating them, culminating in their application to biologically relevant problems.

A key observation, discussed in **chapter 4.1**, concerns the consistent trajectory of the intrinsic dimension (ID) in PLMs compared to models trained on other data modalities. All PLMs examined displayed a reproducible pattern characterized by two phases: 1. an initial peak and 2. a mid-layer plateau with an ID minimum at a local neighborhood scale and a second peak in the ID at larger neighborhood scales. This behavior suggests strong curvature and highly structured representations in the second phase. Models in other modalities also show similarities to each other (particularly the image models which show a characteristic hunchback pattern seen before in convolutional networks), as well as some inconsistencies (the patterns in DNA models are remarkably diverse in this analysis). The robustness of the ID evolution pattern in PLMs suggests that they converge towards a universal and efficient representation of protein space, largely independent of specific architectural or training details. Such universality positions PLMs as an ideal framework for mechanistic interpretability: insights obtained from one model are highly likely to generalize across others, substantially enhancing the reproducibility and impact of interpretability research in computational biology.

Building on this foundation, **chapter 5** demonstrated the potential of Sparse Autoencoders (SAEs) as an effective means of probing and interpreting PLM representations. By mapping latent dimensions to biologically meaningful features derived from UniProt annotations (such as zinc finger motifs and transmembrane domains) we provided a concrete correspondence between neural activations and molecular functions. Notably, this interpretability proved useful for model steering. Through targeted interventions on SAE latents associated with zinc finger motifs, we successfully guided the ESM2 model to generate novel sequences exhibiting the desired structural pattern. This result illustrates how mechanistic interpretability can help us enable directed protein design and, ultimately, AI-assisted protein engineering.

The analyses presented in **chapter 6** further underscored the structured nature of PLM representations. Focusing on the death domain (PF00531), we

found that information encoding is sparsely distributed across neurons, with most relevant signal concentrated in a small subset of them instead of through the whole neuron population. This sparsity is particularly marked within the middle layers, coinciding with the ID minimum. These observations reinforce the notion that intermediate representations have particularly rich structure. Additionally, the identification of an outlier dimension correlated with intrinsically disordered regions provides a compelling parallel to findings in large language models, where outlier dimensions and massive activations seem to be related to particularly frequent tokens.

In **chapter 7**, we move from interpretability to application by finetuning PLMs for protein interface prediction. The resulting models achieved high predictive performance, underscoring the transferability and biological relevance of representations learned during pre-training. Given the centrality of protein interactions to cellular processes and therapeutic discovery, this contribution demonstrates how PLMs can serve as robust computational tools for molecular-level inference and drug design.

Lastly, and in parallel to this work, **chapter 8** investigated the use of unsupervised learning approaches in a clinical setting. By grouping patient profiles from B-cell leukemia cohorts, we uncovered distinct and clinically meaningful categories that corresponded with observed risk. This study resulted in a small and interpretable decision tree for risk assessment, easily applicable to clinical practice.

In summary, this thesis contributes to the growing paradigm shift in computational biology: from using deep learning as a predictive instrument to leveraging it as a mechanistic lens through which biological processes can be examined. Through the integration of intrinsic dimensionality estimation, sparse autoencoders, guided steering and neuron-level analysis we have started to bring some light into the inner information processing of protein language models, as well as demonstrating their applicability to biologically relevant problems and their potential for protein engineering.

Bibliography

[1] Protein data bank: the single global archive for 3d macromolecular structure data. *Nucleic acids research*, 47(D1):D520–D528, 2019.

[2] Uniprot: the universal protein knowledgebase in 2023. *Nucleic acids research*, 51(D1):D523–D531, 2023.

[3] Ahmed Abdulaal, Hugo Fry, Nina Montaña-Brown, Ayodeji Ijishakin, Jack Gao, Stephanie Hyland, Daniel C Alexander, and Daniel C Castro. An x-ray is worth 15 features: Sparse autoencoders for interpretable radiology report generation. *arXiv preprint arXiv:2410.03334*, 2024.

[4] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[5] Etowah Adams, Liam Bai, Minji Lee, Yiyang Yu, and Mohammed AlQuraishi. From mechanistic interpretability to mechanistic biology: Training, evaluating, and interpreting sparse autoencoders on protein language models. *bioRxiv*, 2025.

[6] Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, et al. The gene ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, 2023.

[7] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[8] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[10] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[11] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[12] A Selim Aytuna, Attila Gursoy, and Ozlem Keskin. Prediction of protein–protein interactions by combining structure and sequence conservation in protein interfaces. *Bioinformatics*, 21(12):2850–2855, 2005.

[13] Arjun Banerjee, David Martinez, Camille Dang, and Ethan Tam. Automated neuron labelling enables generative steering and interpretability in protein language models. *arXiv preprint arXiv:2507.06458*, 2025.

[14] Federico Barone, Elena Tea Russo, Edith Natalia Villegas Garcia, Marco Punta, Stefano Cozzini, Alessio Ansuini, and Alberto Cazzaniga. Protein family annotation for the unified human gastrointestinal proteome by dpcfam clustering. *Scientific Data*, 11(1):568, 2024.

[15] Evan H Baugh, Hua Ke, Arnold J Levine, Richard A Bonneau, and Chang S Chan. Why are there hotspot mutations in the tp53 gene in human cancers? *Cell Death & Differentiation*, 25(1):154–160, 2018.

[16] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

[17] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html`, 2023.

[18] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[19] Matthias Blum, Antonina Andreeva, Laise Cavalcanti Florentino, Sara Rocio Chuguransky, Tiago Grego, Emma Hobbs, Beatriz Lazaro Pinto, Ailsa Orr, Typhaine Paysan-Lafosse, Irina Ponamareva, et al. Interpro: the protein sequence classification resource in 2025. *Nucleic acids research*, 53(D1):D444–D456, 2025.

[20] Emmanuel Boutet, Damien Lieberherr, Michael Tognolli, Michel Schneider, and Amos Bairoch. Uniprotkb/swiss-prot: the manually annotated section of the uniprot knowledgebase. In *Plant bioinformatics: methods and protocols*, pages 89–112. Springer, 2007.

[21] Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning. *Advances in Neural Information Processing Systems*, 37:107286–107325, 2024.

[22] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex

Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[24] Elena Buscaroli, Azad Sadr Haghighi, Riccardo Bergamin, Salvatore Milite, Edith Natalia Villegas Garcia, Arianna Tasciotti, Alessio Ansuini, Daniele Ramazzotti, Nicola Calonaci, and Giulio Caravagna. A bayesian framework to infer and cluster mutational signatures leveraging prior biological knowledge. *bioRxiv*, pages 2024–09, 2024.

[25] Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 2021. https://distill.pub/2020/circuits/curve-circuits.

[26] Claudio Ceruti, Simone Bassis, Alessandro Rozza, Gabriele Lombardi, Elena Casiraghi, and Paola Campadelli. Danco: An intrinsic dimensionality estimator exploiting angle and norm concentration. *Pattern recognition*, 47(8):2569–2581, 2014.

[27] John-Marc Chandonia, Naomi K Fox, and Steven E Brenner. Scope: classification of large macromolecular structures in the structural classification of proteins—extended database. *Nucleic acids research*, 47(D1):D475–D481, 2019.

[28] Emily Cheng, Diego Doimo, Corentin Kervadec, Iuri Macocco, Jade Yu, Alessandro Laio, and Marco Baroni. Emergence of a high-dimensional abstraction phase in language transformers. *arXiv preprint arXiv:2405.15471*, 2024.

[29] Hua Cheng, R Dustin Schaeffer, Yuxing Liao, Lisa N Kinch, Jimin Pei, Shuoyong Shi, Bong-Hyun Kim, and Nick V Grishin. Ecod: an evolutionary classification of protein domains. *PLoS computational biology*, 10(12):e1003926, 2014.

[30] Hanhan Cong, Hong Liu, Yi Cao, Cheng Liang, and Yuehui Chen. Protein–protein interaction site prediction by model ensembling with hybrid feature and self-attention. *BMC bioinformatics*, 24(1):456, 2023.

[31] UniProt Consortium. Uniprotkb statistics. Accessed: 24-08-2025. Available at `https://www.uniprot.org/uniprotkb/statistics`.

[32] Chenghua Cui, Wei Shu, and Peining Li. Fluorescence in situ hybridization: cell-based genetic diagnostic and research applications. *Frontiers in cell and developmental biology*, 4:89, 2016.

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

[34] Francesca Cuturello, Federico Pozzo, Edith Natalia Villegas Garcia, Francesca Maria Rossi, Massimo Degan, Paola Nanni, Ilaria Cattarossi, Eva Zaina, Paola Varaschin, Alessandra Braida, et al. An unsupervised machine learning method stratifies chronic lymphocytic leukemia patients in novel categories with different risk of early treatment, 2022.

[35] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, 2025.

[36] Cameron Davidson-Pilon. lifelines: survival analysis in python. *Journal of Open Source Software*, 4(40):1317, 2019.

[37] Warren L DeLano et al. Pymol: An open-source molecular graphics tool. *CCP4 Newsl. protein crystallogr*, 40(1):82–92, 2002.

[38] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[39] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[40] Francesco Denti, Diego Doimo, Alessandro Laio, and Antonietta Mira. The generalized ratios intrinsic dimension estimator. *Scientific Reports*, 12(1):20005, 2022.

[41] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[42] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[43] William N Dudley, Rita Wickham, and Nicholas Coombs. An introduction to survival statistics: Kaplan-meier analysis. *Journal of the advanced practitioner in oncology*, 7(1):91, 2016.

[44] Jesse Durham, Jing Zhang, Ian R Humphreys, Jimin Pei, and Qian Cong. Recent advances in predicting and modeling protein–protein interactions. *Trends in biochemical sciences*, 48(6):527–538, 2023.

[45] Maria d'Errico, Elena Facco, Alessandro Laio, and Alex Rodriguez. Automatic topography of high-dimensional data sets by non-parametric density peak clustering. *Information Sciences*, 560:476–492, 2021.

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

[47] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[48] Ahmed Elnaggar, Hazem Essam, Wafaa Salah-Eldin, Walid Moustafa, Mohamed Elkerdawy, Charlotte Rochereau, and Burkhard Rost. Ankh: Optimized protein language model unlocks general-purpose modelling. *arXiv preprint arXiv:2301.06568*, 2023.

[49] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

[50] Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. *arXiv preprint arXiv:2405.14860*, 2024.

[51] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140, 2017.

[52] Elena Facco, Andrea Pagnani, Elena Tea Russo, and Alessandro Laio. The intrinsic dimension of protein sequence evolution. *PLoS computational biology*, 15(4):e1006767, 2019.

[53] Mingyu Fan, Nannan Gu, Hong Qiao, and Bo Zhang. Intrinsic dimension estimation of data by principal component analysis. *arXiv preprint arXiv:1002.2050*, 2010.

[54] Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. Applying sparse autoencoders to unlearn knowledge in language models. *arXiv preprint arXiv:2410.19278*, 2024.

[55] Naomi K Fox, Steven E Brenner, and John-Marc Chandonia. Scope: Structural classification of proteins—extended, integrating scop and astral data and classification of new structures. *Nucleic acids research*, 42(D1):D304–D309, 2014.

[56] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

[57] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[58] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

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

[60] Edith Natalia Villegas Garcia and Alessio Ansuini. Interpreting and steering protein language models through sparse autoencoders. *arXiv preprint arXiv:2502.09135*, 2025.

[61] Liv Gorton. The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision. *arXiv preprint arXiv:2406.03662*, 2024.

[62] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[63] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[64] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[65] Haoxiang Guan, Jiyan He, and Jie Zhang. Sparse autoencoders reveal interpretable structure in small gene language models. *arXiv preprint arXiv:2507.07486*, 2025.

[66] Onkar Gujral, Mihir Bafna, Eric Alm, and Bonnie Berger. Sparse autoencoders uncover biologically interpretable features in protein language model representations. *Proceedings of the National Academy of Sciences*, 122(34):e2506316122, 2025.

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

[68] Michael Hallek. Chronic lymphocytic leukemia: 2020 update on diagnosis, risk stratification and treatment. *American journal of hematology*, 94(11):1266–1287, 2019.

[69] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 2025.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[71] Michael Heinzinger, Konstantin Weissenow, Joaquin Gomez Sanchez, Adrian Henkel, Milot Mirdita, Martin Steinegger, and Burkhard Rost. Bilingual language model for protein sequence and structure. *NAR Genomics and Bioinformatics*, 6(4):lqae150, 2024.

[72] SeyedMohsen Hosseini, G Brian Golding, and Lucian Ilie. Seq-insite: sequence supersedes structure for protein interaction site prediction. *Bioinformatics*, 40(1):btad738, 2024.

[73] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pages 8946–8970. PMLR, 2022.

[74] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

[75] Sonia Joseph, Praneet Suresh, Ethan Goldfarb, Lorenz Hufe, Yossi Gandelsman, Robert Graham, Danilo Bzdok, Wojciech Samek, and Blake Aaron Richards. Steering clip's vision transformer with sparse autoencoders. *arXiv preprint arXiv:2504.08729*, 2025.

[76] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[77] Dahye Kim and Deepti Ghadiyaram. Concept steerers: Leveraging k-sparse autoencoders for controllable generations. *arXiv preprint arXiv:2501.19066*, 2025.

[78] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[79] Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.

[80] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[81] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[82] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[83] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[84] Andrew G McDonald and Keith F Tipton. Enzyme nomenclature and classification: the state of the art. *The FEBS journal*, 290(9):2214–2231, 2023.

[85] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[86] Alexey G Murzin, Steven E Brenner, Tim Hubbard, and Cyrus Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

[87] Jatin Nainani, Bryn Marie Reimer, Connor Watts, David Jensen, and Anna G Green. Mechanistic evidence that motif-gated domain recognition drives contact prediction in protein language models. *bioRxiv*, pages 2025–08, 2025.

[88] Eric Nguyen, Michael Poli, Matthew G Durrant, Brian Kang, Dhruva Katrekar, David B Li, Liam J Bartie, Armin W Thomas, Samuel H King, Garyk Brixi, et al. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.

[89] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36:43177–43201, 2023.

[90] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 2020. https://distill.pub/2020/circuits/early-vision.

[91] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in.

[92] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[93] Christopher J Oldfield and A Keith Dunker. Intrinsically disordered proteins and intrinsically disordered protein regions. *Annual review of biochemistry*, 83(1):553–584, 2014.

[94] Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, 44(D1):D733–D745, 2016.

[95] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[96] Charles O'Neill, Mudith Jayasekara, and Max Kirkby. Resurrecting the salmon: Rethinking mechanistic interpretability with domain-specific sparse autoencoders. *arXiv preprint arXiv:2508.09363*, 2025.

[97] Hyun Ho Park, Yu-Chih Lo, Su-Chang Lin, Liwei Wang, Jin Kuk Yang, and Hao Wu. The death domain superfamily in intracellular signaling of apoptosis and inflammation. *Annu. Rev. Immunol.*, 25(1):561–586, 2007.

[98] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

[99] Nithin Parsan, David J Yang, and John J Yang. Towards interpretable protein structure prediction with sparse autoencoders. *arXiv preprint arXiv:2503.08764*, 2025.

[100] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[101] Typhaine Paysan-Lafosse, Antonina Andreeva, Matthias Blum, Sara Rocio Chuguransky, Tiago Grego, Beatriz Lazaro Pinto, Gustavo A Salazar, Maxwell L Bileschi, Felipe Llinares-López, Laetitia Meng-Papaxanthos, et al. The pfam protein families database: embracing ai/ml. *Nucleic acids research*, 53(D1):D523–D534, 2025.

[102] Zhangzhi Peng. Ptm-mamba: a ptm-aware protein language model with bidirectional gated mamba blocks. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 5475–5478, 2024.

[103] Damiano Piovesan, Alessio Del Conte, Damiano Clementel, Alexander Miguel Monzon, Martina Bevilacqua, Maria Cristina Aspromonte, Javier A Iserte, Fernando E Orti, Cristina Marino-Buslje, and Silvio CE Tosatto. Mobidb: 10 years of intrinsically disordered proteins. *Nucleic acids research*, 51(D1):D438–D444, 2023.

[104] A Graham Pockley, Gemma A Foulds, Julie A Oughton, Nancy I Kerkvliet, and Gabriele Multhoff. Immune cell phenotyping using flow cytometry. *Current protocols in toxicology*, 66(1):18–8, 2015.

[105] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

[106] Michael Poli, Jue Wang, Stefano Massaroli, Jeffrey Quesnelle, Ryan Carlow, Eric Nguyen, and Armin Thomas. StripedHyena: Moving Beyond Transformers with Hybrid Signal Processing Models, 12 2023.

[107] Protein Data Bank. Pdb data distribution by experimental method and molecular type. Accessed: 25-08-2025. Available at `https://www.rcsb.org/stats/summary`.

[108] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[109] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[110] Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

[111] Shahin Ramazi and Javad Zahiri. Post-translational modifications in proteins: resources, tools and prediction methods. *Database*, 2021:baab012, 2021.

[112] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International conference on machine learning*, pages 8844–8856. PMLR, 2021.

[113] Roshan M Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. *bioRxiv*, 2020.

[114] Laurent Remontet, Zoé Uhry, Nadine Bossard, Jean Iwaz, Aurélien Belot, Coraline Danieli, Hadrien Charvat, Laurent Roche, and CENSUR Working Survival Group. Flexible and structured survival model for a simultaneous estimation of non-linear and non-proportional effects and complex interactions between continuous variables: performance of this multidimensional penalized spline approach in net survival trend analysis. *Statistical methods in medical research*, 28(8):2368–2384, 2019.

[115] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*, 2019.

[116] Serena Rosignoli and Alessandro Paiardini. Boosting the full potential of pymol with structural biology plugins. *Biomolecules*, 12(12):1764, 2022.

[117] Alessandro Rozza, Gabriele Lombardi, Claudio Ceruti, Elena Casiraghi, and Paola Campadelli. Novel high intrinsic dimensionality estimators. *Machine learning*, 89(1):37–65, 2012.

[118] Elena Tea Russo, Federico Barone, Alex Bateman, Stefano Cozzini, Marco Punta, and Alessandro Laio. Dpcfam: unsupervised protein family classification by density peak clustering of large sequence datasets. *PLOS Computational Biology*, 18(10):e1010610, 2022.

[119] Chakkarai Sathyaseelan, L Ponoop Prasad Patro, and Thenmalarchelvi Rathinavelan. Sequence patterns and hmm profiles to predict proteome wide zinc finger motifs. *Pattern Recognition*, 135:109134, 2023.

[120] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *Proceedings of machine learning research*, 235:43632, 2024.

[121] Valerie A Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A Kitts, Terence D Murphy, Kim D Pruitt, Françoise Thibaud-Nissen, Derek Albracht, et al. Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5):849–864, 2017.

[122] Conrad L Schoch, Stacy Ciufo, Mikhail Domrachev, Carol L Hotton, Sivakumar Kannan, Rogneda Khovanskaya, Detlef Leipe, Richard Mcveigh, Kathleen O'Neill, Barbara Robbertse, et al. Ncbi taxonomy: a comprehensive update on curation, resources and tools. *Database*, 2020:baaa062, 2020.

[123] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[124] Damiano Sgarbossa, Cyril Malbranke, and Anne-Florence Bitbol. Protmamba: a homology-aware but alignment-free protein state space model. *Bioinformatics*, 41(6), 2025.

[125] Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

[126] Christian JA Sigrist, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Virginie Bulliard, Amos Bairoch, and Nicolas Hulo. Prosite, a protein domain database for functional characterization and annotation. *Nucleic acids research*, 38(suppl_1):D161–D166, 2010.

[127] Elana Simon and James Zou. Interplm: Discovering interpretable features in protein language models via sparse autoencoders. *bioRxiv*, pages 2024–11, 2024.

[128] Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

[129] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

[130] Bas Stringer, Hans de Ferrante, Sanne Abeln, Jaap Heringa, K Anton Feenstra, and Reza Haydarlou. Pipenn: protein interface prediction from sequence with an ensemble of neural nets. *Bioinformatics*, 38(8):2111–2118, 2022.

[131] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

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

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

[134] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet, 2024. `https://transformer-circuits.pub/2024/scaling-monosemanticity/` [Accessed: 2024].

[135] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[136] David PG Thomas, Carlos M Garcia Fernandez, Reza Haydarlou, and K Anton Feenstra. Pipenn-emb ensemble net and protein embeddings generalise protein interface prediction beyond homology. *Scientific Reports*, 15(1):4391, 2025.

[137] Darin Tsui, Kunal Talreja, and Amirali Aghazadeh. Sparse autoencoders for low-n protein function prediction and design. *arXiv preprint arXiv:2508.18567*, 2025.

[138] Eduard Tulchinskii, Kristian Kuznetsov, Laida Kushnareva, Daniil Cherniavskii, Sergey Nikolenko, Evgeny Burnaev, Serguei Barannikov, and Irina Piontkovskaya. Intrinsic dimension estimation for robust detection of ai-generated texts. *Advances in Neural Information Processing Systems*, 36:39257–39276, 2023.

[139] Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. *Advances in Neural Information Processing Systems*, 36:51234–51252, 2023.

[140] Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 2024.

[141] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

[142] Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. Bertology meets biology: Interpreting attention in protein language models. *arXiv preprint arXiv:2006.15222*, 2020.

[143] Andrea Visentin, Monica Facco, Carmela Gurrieri, Elisa Pagnin, Veronica Martini, Silvia Imbergamo, Federica Frezzato, Valentina Trimarco, Filippo Severin, Flavia Raggi, et al. Prognostic and predictive effect of ighv mutational status and load in chronic lymphocytic leukemia: focus on fcr and br treatments. *Clinical Lymphoma Myeloma and Leukemia*, 19(10):678–685, 2019.

[144] Karthik Viswanathan, Yuri Gardinazzi, Giada Panerai, Alberto Cazzaniga, and Matteo Biagetti. The geometry of tokens in internal representations of large language models. *arXiv preprint arXiv:2501.10573*, 2025.

[145] Vaishali P Waman, Nicola Bordin, Rachel Alcraft, Robert Vickerstaff, Clemens Rauer, Qian Chan, Ian Sillitoe, Hazuki Yamamori, and Christine Orengo. Cath 2024: Cath-alphaflow doubles the number of structures in cath and reveals nearly 200 new folds. *Journal of Molecular Biology*, 436(17):168551, 2024.

[146] Junxuan Wang, Xuyang Ge, Wentao Shu, Qiong Tang, Yunhua Zhou, Zhengfu He, and Xipeng Qiu. Towards universality: Studying mechanistic similarity across language model architectures. *arXiv preprint arXiv:2410.06672*, 2024.

[147] Maurice Weber, Dan Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492, 2024.

[148] David Whitford. *Proteins: structure and function*. John Wiley & Sons, 2013.

[149] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[150] Li C Xue, Drena Dobbs, Alexandre MJJ Bonvin, and Vasant Honavar. Computational prediction of protein interfaces: A review of data driven methods. *FEBS letters*, 589(23):3516–3526, 2015.

[151] Thorsten Zenz, Antonio Sarno, Sonja Häbe, Tina Denzel, Julia Mohr, Dominik Vollmer, Maria Heuberger, Dirk Winkler, Dirk Kienle, Andreas Bühler, et al. 17p deletion in cll: Detailed analysis of tp53 mutations, alternative mechanisms of p53 inactivation, clone size and clonal evolution. *Blood*, 112(11):782, 2008.

[152] Jian Zhang and Lukasz Kurgan. Review and comparative assessment of sequence-based predictors of protein-binding residues. *Briefings in bioinformatics*, 19(5):821–837, 2018.

Appendix A

Intrinsic Dimension Estimation

A.1 Scale Analysis Plots

This section contains the scale analysis of the intrinsic dimension estimation performed on some representative models for each modality (DNA, proteins, language and vision). These were kept of the main text for space concerns.

We can see that on most modalities the shape of the evolution curve remains the same, though the numeric values are do vary. The only exception to this is for protein language models, which present a marked second peak for the intrinsic dimension at larger scales of the estimation. Possible reasons for this behavior are explored in the main texts, but it is likely either due to highly curved or due to highly clustered representation spaces.

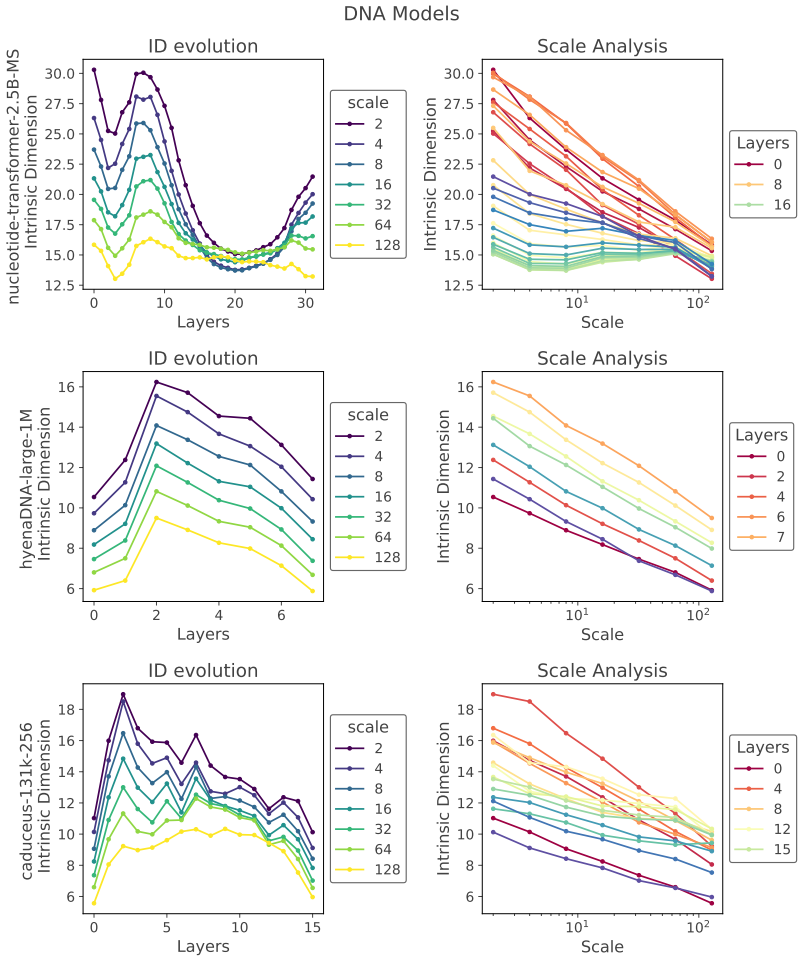


Figure A.1: Scale analysis for the intrinsic dimension estimation of selected DNA language models.

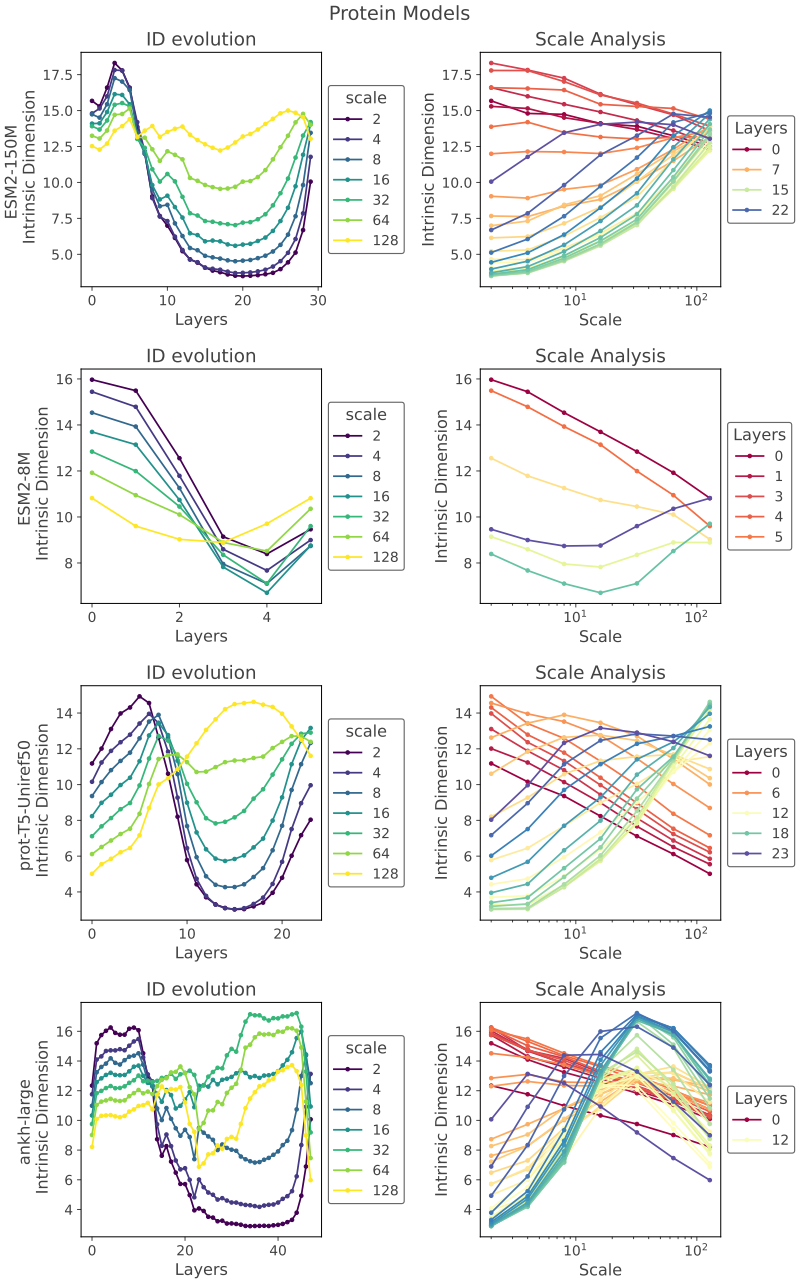


Figure A.2: Scale analysis for the intrinsic dimension estimation of selected protein language models.

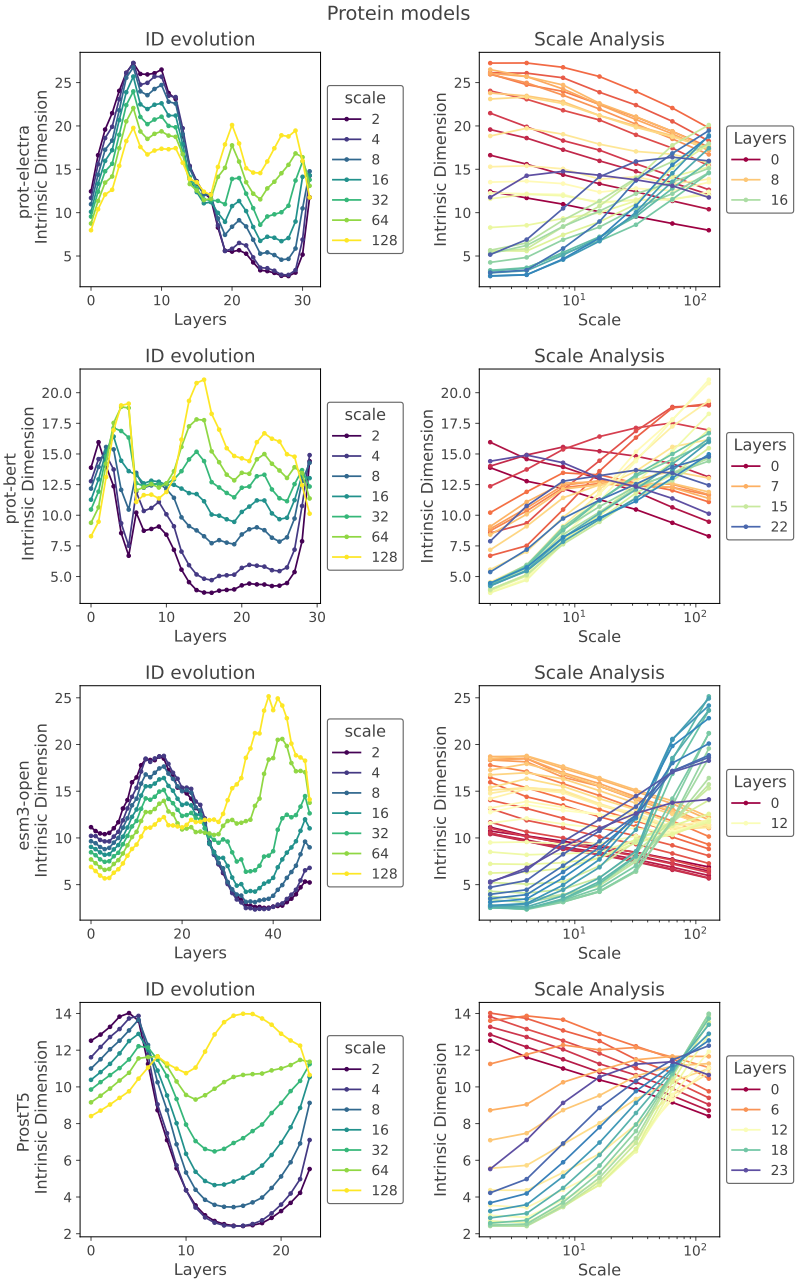


Figure A.3: Scale analysis for the intrinsic dimension estimation of selected protein language models.

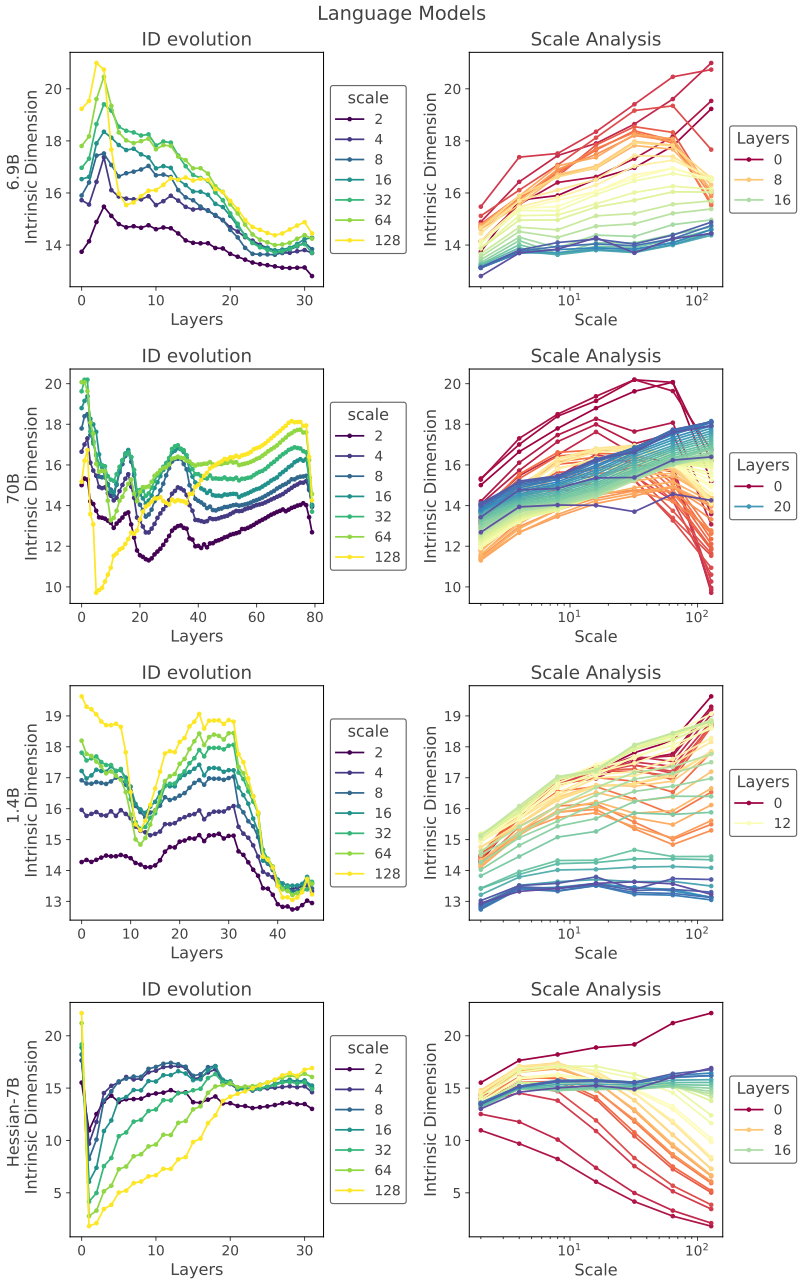


Figure A.4: Scale analysis for the intrinsic dimension estimation of selected language models.

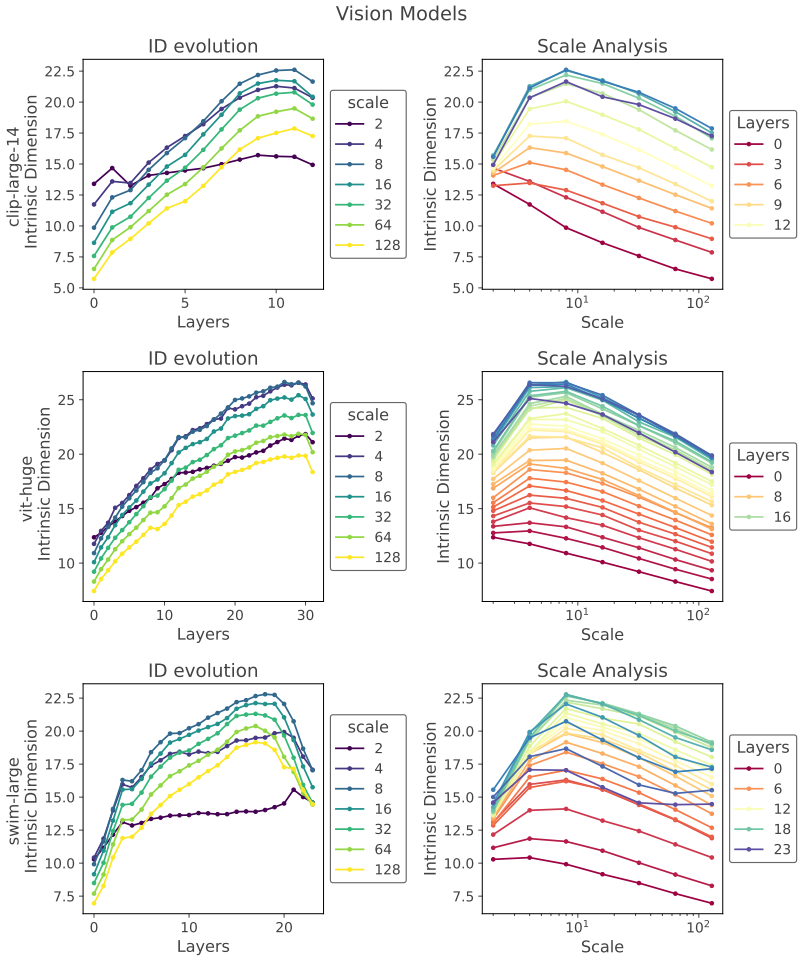


Figure A.5: Scale analysis for the intrinsic dimension estimation of selected vision models.

Appendix B

Risk Stratification via Unsupervised Learning

B.1 Cluster Number Selection Plots

This section contains supporting plots for the selection of the number of clusters k using both k-means and advanced density peaks. For k-means, we select the number of clusters using the elbow method. For advanced density peaks, we plot the number of clusters found by the algorithm against the given level of statistical confidence and we find the k number that is more consistent across different statistical confidence levels.

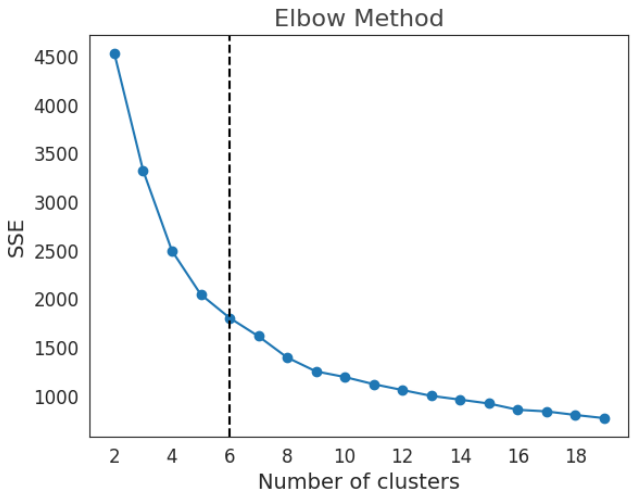


Figure B.1: Elbow plot for the selection of the number of clusters using k-means. We notice a bend around $k = 6$.

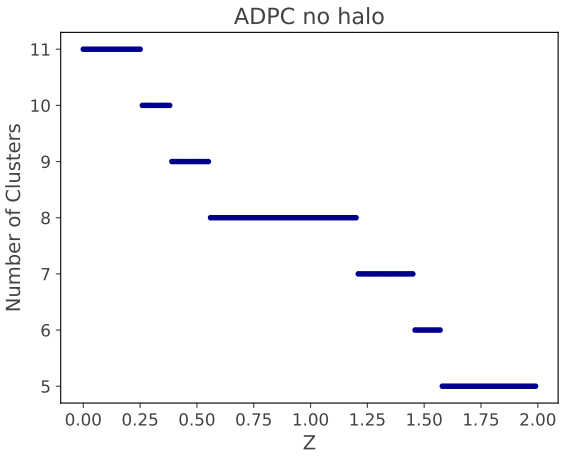


Figure B.2: Number of clusters found by advanced density peaks when varying the desired value of confidence Z. We see that the number of clusters seems to be $k = 8$ over several values of Z (it is persistent), so we select this as our number of clusters.