

TDK-dolgozat

Váradi Kristóf

2024.

A kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében, illetve a TKP2021-EGA-02 számú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, a TKP2021-EGA pályázati program finanszírozásában.



Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Artificial Intelligence and Systems Engineering

Interpretability of Molecular Language Models for Drug Development

Scientific Students' Association Report

Author:

Kristóf Váradi
Year IV.

Advisor:

Márk Marosi,
PhD Student

2024.

Contents

Kivonat	i
Abstract	ii
1 Introduction	1
1.1 Relevance	3
1.2 Structure	3
2 Preliminaries	5
2.1 Machine Learning for Molecular Design	5
2.1.1 Representing Chemical Structures	5
2.1.2 Evaluation Metrics	6
2.2 Attention-Based Sequence Modeling	6
2.3 Mechanistic Interpretability	8
2.3.1 Dictionary Learning	9
2.4 Model Control	9
2.4.1 Fine-Tuning	10
2.4.2 Activation Steering	10
3 Proposed Methods	11
3.1 Autoencoder Configuration	11
3.2 Automated Interpretability	12
3.3 Restrictive Activation Steering	13
4 Experiments	15
4.1 Feature Interpretability	15
4.1.1 Effects on Sample Quality	15
4.1.2 Sparsity	16
4.2 Activation Steering	16
4.3 Experimental Setup	19

4.3.1	Language Models	19
4.3.2	SAEs	19
5	Results	20
5.1	Language Model Performance	20
5.2	Feature Analysis	21
5.2.1	Modes of Activation	21
5.2.2	High-fidelity Features	21
5.3	Automated Annotation	24
6	Discussion	27
6.1	Applications	27
6.1.1	Scientific Understanding	27
6.1.2	Conditional Generation	28
6.2	Related Work	28
6.3	Limitations	29
6.4	Future Work	30
	Acknowledgements	31
	Bibliography	32
	Appendix	39
A.1	List of Annotated Features	39
A.2	Additional Generated Molecules	51

HALLGATÓI NYILATKOZAT

Alulírott *Váradi Kristóf*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2024. december 23.

Váradi Kristóf
hallgató

Kivonat

Az autoregresszív transzformerekhez hasonló, neurális hálózatokon alapuló szekvenciális modelleknek számos alkalmazása ismert a számítógépes gyógyszerfejlesztésben, beleértve a de Novo molekulagenerálást területspecifikus nyelvi modellek segítségével. Az ilyen modellek feketedoboz jellegű viselkedése ennek ellenére korlátozza működésük magyarázhatóságát és kimeneteik irányíthatóságát, amely tényezők kritikusnak számítanak a farmakológiai jellemzők optimalizálásában és a kondicionális generálási feladatok esetében. A dolgozatban a mechanisztikus interpretálhatóság módszereit alkalmazzuk a kémiai nyelvi modellek aktivációinak magyarázásához, továbbá új megközelítéseket ismertetünk az ilyen modellek dekódolás-idejű irányítására. Sor kerül az elkészített eszközök teljesítményértékelésére, valamint kitérünk a tárgyalt módszerek gyógyszeripari folyamatokba történő integrálásának lehetőségeire is.

Abstract

Neural network-based sequence modeling architectures, such as autoregressive transformers, have numerous applications in computational drug discovery, including de Novo generation of drug-like molecules through domain-specific language modeling. However, the black-box nature of these models limits interpretability and controllability — critical for conditional molecule generation and optimization of pharmacological properties. We apply tools from mechanistic interpretability to discover interpretable features in network activations of chemical language models and present new approaches for using them to control molecular generation during inference. We investigate the performance of the proposed methods and further discuss their integration into industrial drug development processes to accelerate in silico drug design.

Chapter 1

Introduction

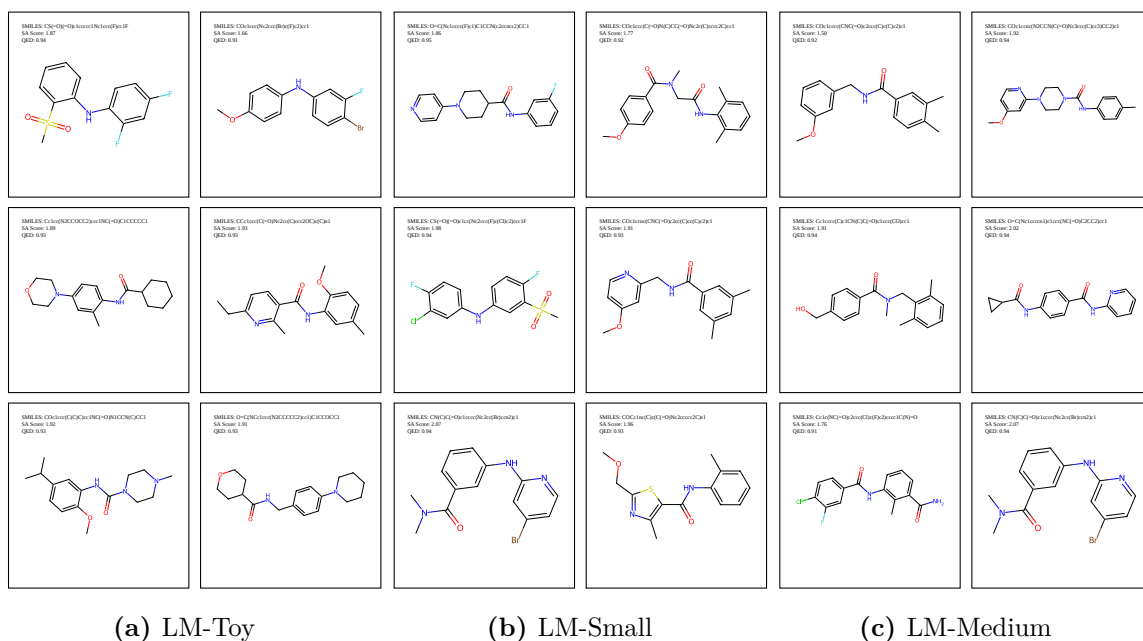


Figure 1.1: Novel SMILES strings and corresponding molecules hand-picked from 10K generated samples at different model sizes, annotated with drug-likeness and synthetic accessibility scores.

In drug development, the path from identifying a potential therapeutic target to getting a safe and effective drug to market is long, intricate, and costly. [Scannell et al. \(2012\)](#) states that the cost of developing a new drug has been doubling approximately every nine years. The process of developing a new therapeutic compound involves extensive collaboration among industrial, academic, and regulatory partners, with a low probability of success. A significant portion of this cost occurs at the early stages of drug discovery, where comprehensive libraries of drug-like compounds are synthesized and tested for desired biological activities. An analysis by [Simoens and Huys \(2021\)](#) indicates that drug discovery and preclinical phases of drug development account for approximately 42% to 50% of the overall research and development expenditure per new medicine. According to their study, estimates of total drug development costs range from 900 million to 2.8 billion US dollars. This significant investment is primarily due to the extensive synthesis and testing of large libraries of therapeutic compounds to identify potential drug candidates.

Computational methods for molecular design have emerged as promising tools to accelerate drug development through the *in silico* generation and evaluation of candidate molecules, reducing the reliance on manual labor and laboratory experimentation. For instance, computer-aided drug design (CADD, [Niazi and Mariam \(2024\)](#)) encompasses various computational approaches that assist in identifying and optimizing potential drug candidates, thereby streamlining the drug discovery pipeline.

Recent advancements in machine learning, particularly in natural language processing (NLP) and representation learning, have resulted in promising methods for molecular design. Applying language modeling techniques to linear sequences representing chemical structures can accurately capture the syntax and semantics of drug-like molecules. Attention-based models, such as transformers, have demonstrated remarkable capabilities in processing sequential data with long-range dependencies, making them suitable for generating novel therapeutic compounds.

Despite the success of neural chemical modeling, interpreting and controlling the behavior of deep generative models in molecular design remains challenging. Neural networks are considered black-box statistical models, meaning that one cannot easily identify causal patterns in its behavior. This effect is compounded by the increasing parameter size of such models, with many general-purpose transformers in production possessing parameters in the range of hundreds and thousands of billions.

Mechanistic interpretability seeks to understand how specific components within a model contribute to its overall function, which is crucial for applications where safety, compliance, and scientific insight are priorities.

In this work, we focus on the applications of mechanistic interpretability to the *de novo* generation of therapeutic compounds. Specifically, we investigate whether sparse autoencoders (SAEs) can be employed as a dictionary learning method on molecular language models.

By training SAEs on the activations of pre-trained language models, we obtain interpretable, sparse representations that can be manipulated to steer the model’s output toward generating molecules with desired properties.

We propose a contrastive steering technique that utilizes the most important SAE features for every contrastive pair and the method’s ability to influence molecular attributes.

We propose to annotate SAE neuron activations using general-purpose language models. In our experiments, we find that large language models can create convincing explanations of similarities between SMILES strings but fail to assess the interpretability of SAE neurons.

The results of our experiments show that methods of mechanistic interpretability achieve promising performance on the task of generating valid and high-quality SMILES strings and argue that the integration of sparse autoencoders can explain and accelerate processes at various stages of the drug development pipeline.

We release code and weights¹ for our language models as well as SAEs.

¹Code is available at github.com/leakedweights/molSAE.

Interactive Colab Notebook is available at [MolSAE.ipynb](#).

Checkpoints can be found at [MolSAE - Drive](#).

1.1 Relevance

The drug development pipeline encompasses a series of methodical and regulated steps that transform a potential therapeutic target into a marketable medication. This pipeline is typically divided into five main stages: Drug Discovery, Preclinical Testing, Clinical Development, Regulatory Review and Approval, and Post-Marketing Surveillance (U.S. Food and Drug Administration, 2022). At each stage, responsible actors must ensure the safety, efficacy, and quality of the final pharmaceutical product. Integrating molecular language models into each stage of the drug development pipeline can enhance efficiency and decision-making.

In the early stages of drug development, models may help researchers select and refine molecules based on potential bioactivity, specificity, and safety. Despite the potential of machine learning methods for therapeutics, in the case of black-box models such as neural networks, the cause of a favoring certain molecular features or interactions cannot be determined exactly. This creates uncertainty, as compounds that seem promising might harbor hidden, unfavorable characteristics not initially apparent.

Interpretability enables practitioners to verify that the model’s predictions are based on known, desirable pharmacological features, reducing the risk of advancing poor candidates through later stages of development. During the drug discovery phase, these models can accelerate the identification and optimization of lead compounds by predicting molecular interactions and bioactivity.

In preclinical testing, chemical language models can help anticipate pharmacokinetic and pharmacodynamic profiles, reducing the reliance on extensive in vitro and in vivo experiments. Throughout clinical development, molecular language models can assist in patient stratification and the identification of biomarkers for better trial design. In Post-Marketing Surveillance, these models can analyze large datasets to detect patterns in adverse reactions and therapeutic outcomes, facilitating rapid responses to safety concerns.

As the process moves into preclinical and clinical testing, interpretability becomes even more critical. Drug candidates must be thoroughly evaluated for safety, efficacy, and alignment with regulatory requirements. Molecular language models that lack transparency cannot easily demonstrate the mechanisms by which a drug achieves its therapeutic effects or avoids off-target interactions, creating obstacles for regulatory approval. An interpretable model, however, could reveal critical insights into how a molecule interacts with specific biological targets or pathways, making it easier to anticipate safety and efficacy issues before costly in vivo studies or clinical trials.

While molecular language models hold the potential to transform drug development, their lack of interpretability remains a major limitation. High explainability would enable researchers to make confident, informed decisions throughout drug development, ensuring that only the most promising, safe, and effective candidates advance. This transparency is essential not only for scientific rigor but also for regulatory compliance.

1.2 Structure

This work is organized into the following chapters and sections:

Preliminaries: Chapter 2 provides an overview of foundational concepts in machine learning for molecular design and the methods used to represent chemical structures. Machine Learning for Molecular Design (Section 2.1) introduces key aspects of applying ma-

chine learning to generating novel chemical structures and discuss representing molecules as linear sequences. We discuss popular metrics for evaluating molecular attributes, including synthetic accessibility (SA) scores, QED, LogP, molecular weight, and Tanimoto similarity. Attention-Based Sequence Modeling (Section 2.2) explains the attention mechanism and reviews the most important aspects of training autoregressive transformers. In Section 2.3 we introduce basic concepts related to mechanistic interpretability, such as linearity, superposition, features, and circuits, and discuss the relevance of interpretability in therapeutics. We briefly introduce sparse dictionary learning and its approximation through sparse autoencoders to obtain interpretable, sparse features from internal representations of language models. In Section 2.4, we describe common methods for controlling language model outputs after the pre-training phase, including fine-tuning and latent space editing. We give an overview of activation engineering, the practice of using steering vectors to modify model activations during inference.

Proposed Methods: Chapter 3 details the configuration and use of SAEs for interpretability and control in MLMs. We describe the architecture and training of conventional sparse autoencoders on transformer activations, focusing on sparsity constraints to obtain interpretable representations. Section 3.2 Outlines methods for automated neuron annotation using a general-purpose large language model, providing a scalable approach to feature interpretation. In Section 3.3, we introduce an iteration of activation addition steering approach that applies constraints on activation SAE features at different scales depending on their importance, aiming to generate molecules with targeted chemical properties.

Experiments: Chapter 4 discusses experimental setups and methodologies for evaluating the interpretability and control mechanisms in MLMs. Section 4.1 investigates the interpretability of features extracted by SAEs, analyzing sample quality and feature sparsity. We examine how sparsity affects the validity and QED of generated SMILES sequences and analyze the sparsity of the average SAE feature, as well as the impact of the SAE sparsity coefficient on the interpretability of molecular representations. In Section 4.2, we assess the performance of restrictive activation steering in generating molecules with specific pharmacological and structural properties on the contrastive pair Diazepam (+) and Gentamicin C₁ (−). Section 4.3 provides details on model architectures, tokenization, training parameters, and SAE configurations.

Results: Chapter 5 analyzes experimental results, evaluating the performance of different transformer configurations and the effectiveness of interpretability and steering methods. We summarize language model performance across different model sizes, focusing on validity, QED, and SA scores. We explore the interpretability of SAE-derived features, including operational modes and high-fidelity features. We discuss the distinct operational modes within SAE neurons that contribute to both interpretability and controllability. We point to a number of high-fidelity SAE neurons in our 2048-latent toy model and provide a detailed analysis of interpretable features and their role in generating chemically valid structures. Section 5.3 evaluates automated feature annotation using large language models and discusses their advantages and limitations for annotating neuron activations in chemical transformers.

Chapter 2

Preliminaries

2.1 Machine Learning for Molecular Design

Molecular design involves creating new chemical compounds with specific desired properties, which is fundamental in fields like drug discovery, materials science, and agrochemicals. Traditionally, this process has been empirical and time-consuming, relying heavily on laboratory experimentation. Advances in computational chemistry accelerated molecular design by enabling the *in silico* generation and evaluation (Terstappen and Reggiani, 2001) of candidate molecules before synthesis.

2.1.1 Representing Chemical Structures

The Simplified Molecular Input Line Entry System (SMILES) is widely used for representing chemical structures as character sequences (Weininger, 1988). It encodes molecules by converting atoms and the connectivity between them into one-dimensional strings, while retaining structural information for further processing. SMILES allows the direct use of computational techniques developed for natural language processing to model and generate chemical structures in a natural way.

In SMILES, a molecule M is represented as a sequence $S = (s_1, s_2, \dots, s_n)$, where each s_i is a token corresponding to atoms, bonds, branches, or ring closures. Chemical structures are converted to SMILES strings according to the following rules:

- **Atoms, Bonds:** Atoms are denoted by their atomic symbols, with special handling for non-alphanumeric characters and multi-character symbols. Bonds are represented by specific symbols, e.g. '-' for single bonds, '=' for double bonds, '#' for triple bonds, '\$' for quadruple bonds, and ':' for aromatic bonds. Implicit single bonds do not require the use of special characters and are inferred from adjacency.
- **Branches:** Branches are indicated using parentheses. For a branching point at position t , the branch starting at t is enclosed as $S_{[1:t-1]} s_t (S_b) S_{[t+1:]}$, encoding that substructure S_b branches off from the main chain at s_t .
- **Rings:** Ring structures are represented by matching digits. To denote the n -th ring closure between atoms at positions t_k and t_m , the SMILES string includes the same digit at both positions:

$$S = S_{[1:t_k-1]} s_{t_k} n S_{[t_k+1:t_m-1]} s_{t_m} n S_{[t_m+1:]}$$

- **Stereochemistry:** Optional stereochemical information is encoded using symbols placed after the chiral center atom to indicate its configuration:

Clockwise stereochemistry (R): @, Counterclockwise stereochemistry (S): @@

Though SMILES is the most popular linear encoding method, commonly used implementations can often generate multiple valid representations for a single chemical structure. To address representational ambiguity, newer approaches (Weininger et al., 1989; O’Boyle, 2012) attempt to develop algorithms to obtain canonical SMILES representations by enforcing pre-defined orders of traversal. SMILES encodings also lack three-dimensional spatial information, which is essential for certain molecule processing and generation tasks.

2.1.2 Evaluation Metrics

Determining the quality and properties of chemical structures involves complex evaluations due to the diverse and interrelated molecular characteristics. The following metrics provide standardized quantitative measures to assess key chemical and pharmacological attributes. Here, we only share the metrics that are most relevant to our examination.

- **Synthetic Accessibility (SA) Score:** The SA score estimates the ease of synthesizing a molecule in a laboratory setting (Ertl and Schuffenhauer, 2009). It combines fragment contributions and structural complexity into a score ranging from 1 (easy to synthesize) to 10 (difficult to synthesize).
- **Quantitative Estimation of Drug-likeness (QED):** QED evaluates how closely a molecule aligns with known drug-like properties (Bickerton et al., 2012). It integrates multiple molecular attributes—such as lipophilicity, molecular weight, and hydrogen bond donors—into a single value between 0 (undesirable) and 1 (ideal drug-likeness).
- **Octanol-Water Partition Coefficient (LogP):** LogP quantifies a molecule’s lipophilicity by measuring the logarithm of its partition coefficient between n-octanol and water. LogP values play an important role in predicting a compound’s absorption and distribution characteristics.
- **Molecular Weight:** The sum of the atomic weights of all atoms in a molecule. Molecular weight affects a compound’s pharmacokinetic properties, including absorption and distribution. It is a fundamental parameter in drug design constraints.
- **Tanimoto Similarity:** The Tanimoto coefficient measures the similarity between two molecules based on their binary fingerprint representations (Rogers and Hahn, 2010). Ranging from 0 (no similarity) to 1 (identical), it is used to assess how closely a generated molecule resembles known compounds.

2.2 Attention-Based Sequence Modeling

Attention-based language models, particularly those utilizing the transformer architecture (Vaswani et al., 2023), have revolutionized language processing by capturing long-range dependencies in sequential data in a highly parallelizable fashion.

The key component of these models is the self-attention mechanism, which computes a weighted representation of other relevant tokens in the sequence when processing each token, formally:

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

where d_k is the dimensionality of the key vectors. The attention computation allows language models to focus on different parts of the input sequence dynamically, enabling models to generate coherent and contextually relevant samples.

Decoder-style transformers, such as GPT variants (Radford and Narasimhan, 2018), are usually trained to predict the next token in a sequence given all previous tokens by classifying over the model’s vocabulary, a paradigm termed causal language modeling. Formally, given a sequence of tokens $x = (x_1, x_2, \dots, x_n)$, the model estimates the probability of the sequence as the product of conditional probabilities:

$$P(x) = \prod_{t=1}^n P(x_t \mid x_{<t}),$$

where $x_{<t} = (x_1, x_2, \dots, x_{t-1})$. The conditional probability $P(x_t \mid x_{<t})$ is modeled using the softmax function over the vocabulary V :

$$P(x_t \mid x_{<t}) = \frac{\exp(\mathbf{w}_{x_t}^\top \mathbf{h}_t)}{\sum_{k=1}^{|V|} \exp(\mathbf{w}_k^\top \mathbf{h}_t)},$$

where $\mathbf{h}_t$ is the hidden state at position t , $\mathbf{w}_k$ is the weight vector corresponding to token k in the output vocabulary, and $|V|$ is the size of the vocabulary.

In decoder-style CLMs, transformer blocks usually consists of:

- **Masked Self-Attention:** Causality is achieved by applying a causal mask to ensure that the prediction for each token only depends on previous tokens. Causal masks prevent the model from accessing future tokens during inference. Masking is achieved by adding a large negative value to $QK^\top$ where tokens should not be considered. Self-attention refers to the choice of queries, keys, and values. Specifically, we refer to self-attention when $Q = K = V$ and cross-attention when $Q \neq K$ and $K = V$. Transformer blocks are often configured with multiple heads – sublayers that calculate attention on different parts of token representations, allowing them to attend to different parts of a sequence.
- **Position-Wise Feed-Forward Networks (FFN)**
- **Residual Connections and Normalization Layers:** Residual connections serve as shortcuts between different computational elements in neural networks. Along with normalization techniques such as LayerNorm (Ba et al., 2016) and RMSNorm (Zhang and Sennrich, 2019), they are used to improve training stability.

For a high-level illustration of decoder-style transformers, see Figure 3.1.

CLMs are primarily trained with Maximum Likelihood Estimation (MLE). In practice, MLE is implemented by minimizing cross-entropy across N training examples:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \log P(x_t = x_{i,t} \mid x_{i,<t}; \theta),$$

where T denotes the token length of the i -th sequence and θ are the parameters of the transformer.

2.3 Mechanistic Interpretability

Interpretability refers to the ability to understand the behavior of a machine learning model. As statistical models become more complex, like in the case of modern language models, interpreting their operation becomes challenging.

Interpretability is particularly important in pharmacological applications including molecule generation for several reasons, such as:

1. **Safety and Compliance:** Understanding how a model generates molecules can help prevent the creation of compounds with undesirable properties, such as toxicity or low bioavailability.
2. **Chemical Understanding:** Interpretable models can provide insights into underlying chemical principles, potentially leading to the discovery of new chemical knowledge.
3. **Sample Quality:** Identifying failure modes of a chemical language model through its internal representations can guide the development of more effective architectures and training strategies.
4. **Controlled Generation:** Circuits and feature vectors can be manipulated during inference to generate more promising drug candidates and molecules conditioned on inputs such as molecular weight or atomic properties.

In broad terms, mechanistic interpretability (MI) (Elhage et al., 2021) aims to dissect model internals to understand how specific components of an architecture contribute to its overall function, primarily through discovering or engineering causal mechanisms (Saphra and Wiegrefe, 2024). Current MI research is centered around analyzing features and circuits. Although there is no consensus on the exact definitions of these terms, a feature can be imagined as a meaningful property of the training data captured by the model and a circuits can be interpreted as the algorithmic subsystems connecting specific features.

Mechanistic Interpretability commonly operates under two assumptions: linearity and superposition. The linear representation hypothesis (Park et al., 2024; Piantadosi et al., 2024) states that neural networks encode interpretable attributes as linear directions within their activation space. In other words, specific features or concepts correspond to particular vectors, allowing for linear operations to manipulate these features.

The superposition hypothesis (Arora et al., 2018; Elhage et al., 2022) operates under the assumption of linearity and suggests that neural networks represent more features than they have capacity to store by compressing a number of features into single neurons through superposition. Superposition leads to polysemanticity, where neurons may respond to multiple, unrelated features, complicating their explainability.

We direct the reader to [Rai et al. \(2024\)](#) for an introduction to mechanistic transformer interpretability.

2.3.1 Dictionary Learning

Sparse dictionary learning ([Chen et al., 2017](#); [Cunningham et al., 2023](#)) is a method in machine learning where the goal is to find a set of basis functions (the dictionary) that can sparsely represent input data, where each datapoint can be approximated by a linear combination of only a few dictionary elements.

Recently, sparse autoencoders (SAEs) ([Cunningham et al., 2023](#)) emerged as a promising method approximating dictionary learning on language models in an unsupervised fashion. SAEs cast activations to a higher-dimensional space for creating monosemantic features. They are trained to reconstruct the low-dimensional representations while enforcing a sparsity constraints, thus separating entangled representations.

Sparse autoencoders are able to successfully decompose entangled representations of language models at various scales, ranging from one layer toy models ([Bricken et al., 2023](#)) to production-grade language models with tens of billions of parameters ([Templeton et al., 2024](#); [Gao et al., 2024](#); [Lieberum et al., 2024](#)). Interpretable SAE representations can be used to steer text generation at inference ([Templeton et al., 2024](#)) or classify text sentiments, such as harmful prompts([Bricken et al., 2024](#)).

2.4 Model Control

Methods to control language decoding are essential for achieving high-quality outputs after pre-training a language model on large datasets. Generally, control methods can be categorized into three primary types: inference-time control, post-training modification, and fine-tuning.

The most common way guide a language model is through prompting – specifying instructions for the model to follow. Prompts can steer models to generate text in a particular style, avoid certain topics or words, or apply advanced reasoning techniques to solve tasks. Although prompting is a widely used method to influence language model outputs, it requires considerable manual effort and can yield inconsistent results across different contexts. Moreover, prompting is vulnerable to adversarial attacks, where instructions are manipulated to exploit the model’s tendency to follow instructions.

Further training on specialized datasets has been shown to improve model performance across various tasks. By conditioning the model on specific response styles or content preferences, this approach aligns the model’s responses more closely with desired behaviors. Fine-tuning is used in every large-scale production language model, though its success is limited by the high training costs and rigidity during inference.

Post-training modifications, for example latent space editing in [Meng et al. \(2023\)](#), modify learned parameters without retraining on a specialized dataset. This approach can yield various benefits, though its success, similar to fine-tuning, is limited by rigidity during inference.

2.4.1 Fine-Tuning

Fine-tuning is a common approach for adapting pre-trained language models to specific tasks or domains. This method updates the model’s parameters using task-specific data, allowing the model to perform functions that go beyond its initial training.

Reinforcement Learning from Human Feedback (RLHF, [Ouyang et al. \(2022\)](#)) aligns language models with human preferences by incorporating human evaluations into the training process. In this approach, human annotators rate model outputs, and these ratings are used to train a reward model. The language model is then fine-tuned using reinforcement learning algorithms, such as Proximal Policy Optimization (PPO, [Schulman et al. \(2017\)](#)) or Direct Preference Optimization (DPO, [Rafailov et al. \(2024\)](#)), to optimize the reward model’s feedback, producing outputs that align more closely with human expectations.

Reinforcement Learning from AI Feedback (RLAIF, [Lee et al. \(2024\)](#)) builds on RLHF by relying on feedback from AI systems instead of human evaluators. In RLAIF, the reward model is trained on preference data generated by language models. RLAIF addresses the scalability issues of human feedback by automating the evaluation process, reducing the need for human involvement while still allowing for effective fine-tuning.

Low-Rank Adaptation (LoRA, [Hu et al. \(2021\)](#)) improves the efficiency of fine-tuning by adding low-rank matrices into the model’s architecture. By keeping the original model weights frozen and training only these additional low-rank matrices, LoRA reduces the number of parameters that need to be adjusted, which makes fine-tuning less computationally demanding.

2.4.2 Activation Steering

A recently developed approach for guiding language model outputs during inference is the use of steering vectors. Steering vectors ([Subramani et al., 2022](#)) are specific modifications applied to the internal activations of generative neural networks during inference to produce specific characteristics in generated samples. Steering vectors can be a zero-cost method to influence aspects such as sentiment, style, or factual accuracy without retraining the model.

[Turner et al. \(2024\)](#) introduce Activation Engineering, a term referring to the practice of applying systematic changes to model activations to control the behavior of generative models. They propose to guide activations using a linear combination of pairs of contrastive prompts. Formally, given the i -th transformer layer l_i and natural language prompts $\mathbf{x}^+$ and $\mathbf{x}^-$ with contrastive sentiments, the steering vector is composed of the scaled difference of the layer’s activations $\mathbf{z}^+$, $\mathbf{z}^-$ on the positive and negative instructions:

$$\hat{\mathbf{z}} = \mathbf{z} + \sigma(\mathbf{z}^+ - \mathbf{z}^-)$$

The authors evaluate activation addition on a steering vector composed of the contrastive pair "love" and "hate" to steer sentiments in various contexts. They report state of the art performance on negative-to-positive sentiment shift and detoxification.

Chapter 3

Proposed Methods

3.1 Autoencoder Configuration

To obtain interpretable representations of the language model’s internal activations, we employ SAEs trained on the activations of pre-trained language model at different capacities. For simplicity, we use standard ReLU SAEs with pre-encoder bias from [Bricken et al. \(2023\)](#) in all of our experiments.

Let $\mathbf{x}$ denote the input activation vector from the language model. The SAE architecture consists of an encoder and a decoder with weight matrices $\mathbf{W}_{\text{enc}}$ and $\mathbf{W}_{\text{dec}}$, respectively. The encoder transforms the input into a latent representation $\mathbf{f}$ using a ReLU activation function:

$$\mathbf{f} = \text{ReLU}(\mathbf{W}_{\text{enc}}\tilde{\mathbf{x}} + \mathbf{b}_{\text{enc}}) \quad (3.1)$$

Here, $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{b}_{\text{dec}}$ represents the input vector adjusted by the decoder bias $\mathbf{b}_{\text{dec}}$. The decoder then reconstructs the input as:

$$\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{dec}} \quad (3.2)$$

The autoencoders are trained with a composite objective consisting of an L_2 reconstruction term on the SAE output and an L_0 sparsity term on the latent activations:

$$\mathcal{L} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\mathcal{L}_{\text{rec.}}} + \underbrace{\lambda \|\mathbf{f}\|_1}_{\mathcal{L}_{\text{sparsity}}} \quad (3.3)$$

In this work, we primarily focus on training SAEs at the residual stream of language models. The residual stream refers to the sum of activations that flow through previous transformer blocks. Residual streams are the most common sites for sparse autoencoders due to a track record of high quality features ([Cunningham et al., 2023](#); [Templeton et al., 2024](#)) and their potential ability to decrease cross-layer superposition ([Templeton et al., 2024](#)) in larger models. Additionally, we train and evaluate post-norm Multi-Layer Perceptron (MLP) autoencoders as a comparative baseline. Unlike residual stream SAEs, post-norm MLP autoencoders use the normalized activations of feed-forward networks. Figure 3.1 illustrates the dynamics of transformer blocks and SAEs considered in this work.

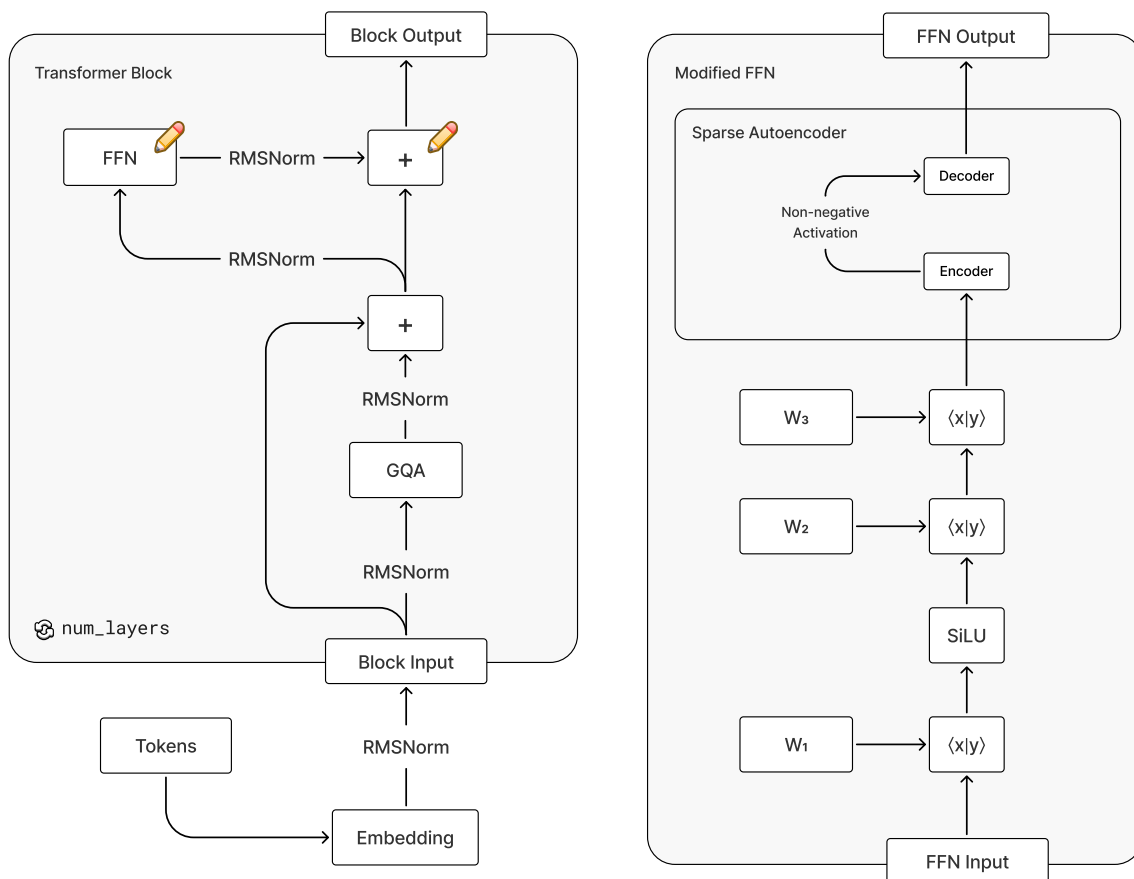


Figure 3.1: Layout of transformer blocks (left) and feed-forward networks augmented with SAEs (right). Pencil icons indicate sparse autoencoder sites.

3.2 Automated Interpretability

Manually evaluating latent activations of SAEs with thousands of neurons at multiple sites is labor-intensive and infeasible for transformers at a practical scale. Interpreting SMILES strings and drawing relationships between activations on different molecules is much harder than identifying contexts of the most prominent feature activation in natural language. It demands accurate visualization and its success depends on the annotator’s chemical understanding and pharmacological knowledge.

We argue that large language models can be used to explain chemical compounds purely identified by corresponding SMILES strings. Transformers are efficient at capturing dependencies between sequences and do not require visual representations to understand complex linear structures, even in situations where they are incomprehensible by humans. Automatic annotation can also be parallelized and is much more cost-effective than hiring multiple human annotators with a high level of chemical understanding.

Following [Bills et al. \(2023\)](#), we propose to search for chemically relevant SAE features using generative language models to label neuron activations on SMILES tokens and automatically evaluate feature interpretability. We prompt a general-purpose language model to output a single sentence explaining the relationship between top activations, and an

interpretability score ranging 1-5. We instruct the model to use multi-step reasoning (Lightman et al., 2023; Wei et al., 2023) to identify dependencies between the top 20 activations of sparse autoencoder features. A natural choice for this task is the OpenAI o1 family of models (OpenAI, 2024b) because of their reasoning-oriented training. Figure 3.2 outlines our approach to feature extraction using o1-mini, a cost-effective choice considering per-token pricing. As the API access to OpenAI o1-mini is limited at the time of completing this work, we resort to using GPT-4o (OpenAI, 2024a) while enforcing structured generation, where each response contains an array of strings explaining the model’s reasoning. We find that language models can be used to extract molecular information from linear representations efficiently.

Prompt	OpenAI o1-mini Response
Context Your task is to examine sparse autoencoder neurons on a molecular language model to interpret which properties and sequences they fire on. Task #1 Then, output a single sentence explaining the causal factor and a single value between 1-5 determining whether this feature is interpretable. Task #2 You have to reason about the molecule and the similarities between the sequences. Grading Criteria 1 means not interpretable, 2 means somewhat interpretable (there is a vague connection between the top sequences), 3 means interpretable (there is a close connection, some examples might not be connected), 4 means interpretable (context can be clearly identified), 5 means the neurons exactly identify a pattern in the strings. Molecule Data Sequence 407727 - 11.9080: <chem>Cc1ccn(-c2ccc(CNC3C(=O)O)CC1)C3</chem> Sequence 131177 - 11.6716: <chem>COc1ccc(-n2nc(-c3ccc(C(F)(F)F)nc3)nc2)cc1</chem>	Reasoning Identifying key patterns I’m analyzing the top 20 SMILES sequences that activated the neuron, noticing a frequent presence of fluorine atoms, sulfonyl groups, nitrile or nitrogen-containing rings, and carbonyl groups. Laying out the options I’m thinking through a neuron’s response: fluorinated groups, nitrogens, aromatic rings, sulfonyl groups, nitrogen heterocycles, chiral centers, and perfluoroalkyl chains. Response The neuron is activated by fluorinated aromatic compounds containing nitrogen heterocycles. Interpretability Score: 4

Figure 3.2: Demonstrative example of automated neuron annotation using a general-purpose language model.

We find that the model generates convincing explanations for the relationship between the most active sequences but fails to differentiate between uninterpretable and highly interpretable features. Almost all of the neurons we annotated got an interpretability score of 4, with only a few occurrences of scores 3 and 5, and no occurrences of 1’s and 2’s. Our results suggest that language models fail to discard unconvincing explanations as uninterpretable, therefore more effective grading strategies are required.

3.3 Restrictive Activation Steering

Early results from Nanda et al. (2024) indicate that sparse autoencoder features can also be used to steer language models. Instead of modifying the activations of the language model, they combine base SAE magnitudes with activations of contrastive pairs. We argue that, even if this approach comes with a slight decrease in performance in comparison to

regular activation steering, the surplus information about the nature of these modifications may be worth it in pharmacological applications.

Let $\mathcal{A}$ denote a sparse autoencoder applied at transformer block $\mathcal{B}$. The activations produced by the SAE are represented by the vector $\mathbf{z}_{\mathcal{A}}$ and the transformer block’s activations are represented by $\mathbf{z}_{\mathcal{B}}$. Suppose we have a sequence $\mathbf{x}^+$ with desired properties and $\mathbf{x}^-$ with undesired properties, and according to Equation 3.1, their SAE activations are calculated as:

$$\mathbf{z}_{\mathcal{A}}(\mathbf{x}^+) = \text{ReLU}(\mathbf{W}_{\text{enc}} \tilde{\mathbf{z}}_{\mathcal{B}}(\mathbf{x}^+) + \mathbf{b}_{\text{enc}})$$

$$\mathbf{z}_{\mathcal{A}}(\mathbf{x}^-) = \text{ReLU}(\mathbf{W}_{\text{enc}} \tilde{\mathbf{z}}_{\mathcal{B}}(\mathbf{x}^-) + \mathbf{b}_{\text{enc}})$$

We select k features with the largest activation on contrastive pairs:

$$\mathcal{K} = \text{TopK}(\mathbf{z}_{\mathcal{A}}(\mathbf{x}^+) - \mathbf{z}_{\mathcal{A}}(\mathbf{x}^-))$$

Here, $\mathcal{K}$ represents the set of indices corresponding to the top- k features with the highest autoencoder activation differences, and $\text{TopK}(\cdot)$ corresponds to the indices of the k largest elements from the input vector. We construct a binary mask $\mathbf{m} \in \{0, 1\}^n$, where each element m_i is set to 1 if the corresponding feature index i is within the top- k set $\mathcal{K}$, and 0 otherwise.

To control the influence of the SAE activations on the language model output, we introduce a scaling factor σ and the top- k importance coefficient α . The scaled SAE activations $\mathbf{z}_{\mathcal{A}}^{\text{scaled}}$ are computed by applying the scaling factor α to the top- k features and $1 - \alpha$ to the remaining features:

$$\mathbf{z}_{\mathcal{A}}^{\text{scaled}} = \alpha \cdot \mathbf{m} \odot \mathbf{z}_{\mathcal{A}} + (1 - \alpha) \cdot (\mathbf{1} - \mathbf{m}) \odot \mathbf{z}_{\mathcal{A}},$$

where $\odot$ denotes element-wise multiplication, and $\mathbf{1}$ is a vector of ones.

The scaled SAE activations are then integrated with the model activations. The combined activations, represented as $\mathbf{z}_{\mathcal{B}}^{\text{mod}}$, are calculated according to Equation 3.2. Given an input sequence $\mathbf{x}$, the modified activations at the SAE site are computed as:

$$\mathbf{z}_{\mathcal{B}}^{\text{mod}}(\mathbf{x}) = \mathbf{z}_{\mathcal{B}}(\mathbf{x}) + \sigma \cdot \mathbf{W}_{\text{dec}} \mathbf{z}_{\mathcal{A}}^{\text{scaled}} + \mathbf{b}_{\text{dec}}.$$

We believe that applying greater scaling factors to top- k contrastive activations helps retain information that is important in the context of the steered sequence. By intuition, modifying only the most significant features in the contrastive activations could help the model retrain its representational power while influencing useful properties. In Section 4.2, we demonstrate that top- k steering is a promising tool for conditional generation and can also achieve high validity and generation quality in comparison with regular SAE activation steering.

Chapter 4

Experiments

4.1 Feature Interpretability

The sparsity term in Equation 3.3 penalizes joint neuron activations. Enforcing higher sparsity increases the likelihood of finding interpretable latent neurons, but creating disentangled representations at the same capacity comes with a tradeoff in sample quality, therefore selecting appropriate values of λ during training is essential.

4.1.1 Effects on Sample Quality

We investigate the quality of generated SMILES sequences in molecular language models at different sparsity coefficients. Figure 4.1 presents effect of changes in λ with respect to SMILES validity and drug-likeness. We compare SAEs at the residual stream and feed-forward layer to the unmodified toy model.

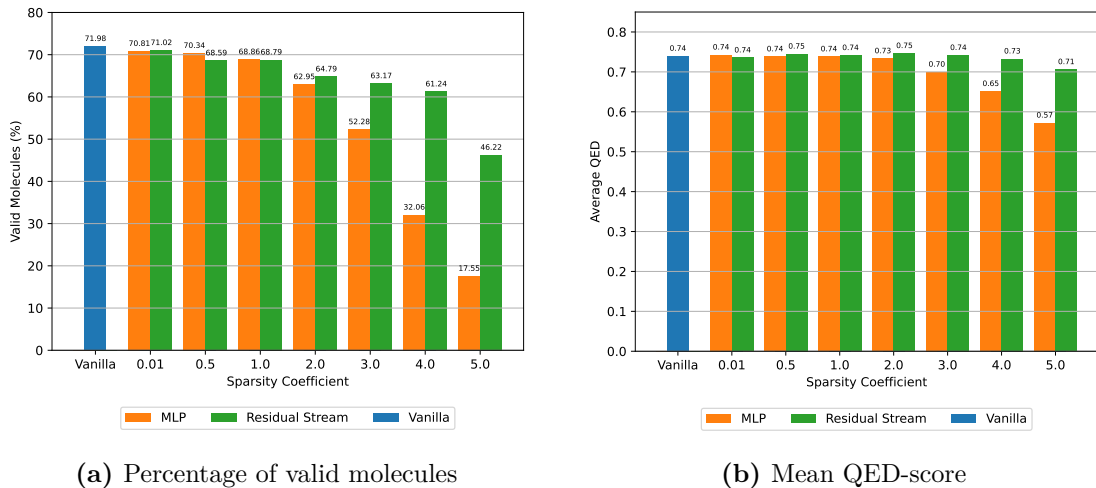


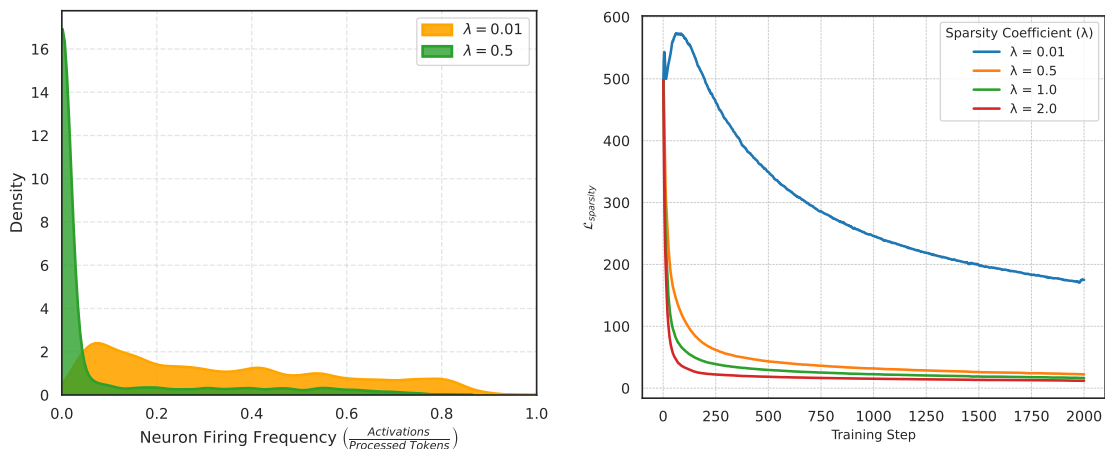
Figure 4.1: Quality of SAE-modified representations on the one-layer toy model with 16x SAEs across different values of λ , calculated from 10K samples generated at unit temperature.

High sparsity levels constrain the network’s capacity to capture complex molecular features, leading to a reduction in both the validity and quality of the generated molecules, though the decline is minimal with values of λ under 1.0. The observed decline in the

validity and QED scores of generated samples is particularly pronounced in post-norm MLP SAEs. This behavior is expected, given that feed-forward activations play a more significant role in transformers.

4.1.2 Sparsity

We train SAEs with different sparsity coefficients and analyze the sparsity and capacity of the resulting models. We test values of λ ranging from 0.001 to 5.0 and monitor the sparsity and the decrease in molecule quality resulting from swapping the transformer activations to those reconstructed by sparse autoencoders. A coefficient below 0.5 resulted in saturated activations with a sparsity loss above 20 and λ 's above 1.0 significantly hindered the SAE's ability to reconstruct transformer representations.



(a) Kernel density estimates of SAE firing frequency at different values of λ . (b) Sparsity loss during training at different values of λ .

Figure 4.2a depicts the frequency of SAE neuron activity at two different sparsity coefficients. While the average neuron at $\lambda = 0.01$ is active on hundreds of thousands of tokens, a normal neuron at $\lambda = 0.5$ is only active on a couple thousand tokens without a significant degradation in molecule quality. For a possible explanation of this phenomenon, see Section 5.2.1.

Figure 4.2b depicts the sparsity loss when training a 2048-latent autoencoder for 10 epochs on 1M transformer activations. We don't include runs with λ below 0.01 and above 2.0 since they either produce oversaturated activation vectors or do not reconstruct transformer representations accurately enough.

4.2 Activation Steering

We argue that activation steering can be used for conditional molecule generation in chemical language models. We provide a demonstrative example of restrictive activation steering on a pair that has significantly contrasting molecular and pharmacological properties.

We consider the following contrastive pair:

- **Diazepam:** CN1C(=O)CN=C(C2=C1C=CC(=C2)C1)C3=CC=CC=C3 – a benzodiazepine derivative characterized by a diazepam ring fused to a chlorinated benzene ring,

and it is employed as an anxiolytic, anticonvulsant, and muscle relaxant through its enhancement of GABA_A receptor activity.

- **Gentamicin C₁**: C[C@H]([C@@H]1CC[C@H]([C@H](O1)O[C@@H]2[C@H](C[C@H]([C@@H]([C@H]2O)O[C@@H]3[C@@H]([C@H]([C@@H](CO3)(C)O)NC)O)N)N)NC – an aminoglycoside antibiotic featuring a complex aminocyclitol core with multiple amino and hydroxyl groups. Gentamicin C₁ it is used to treat bacterial infections by inhibiting protein synthesis.

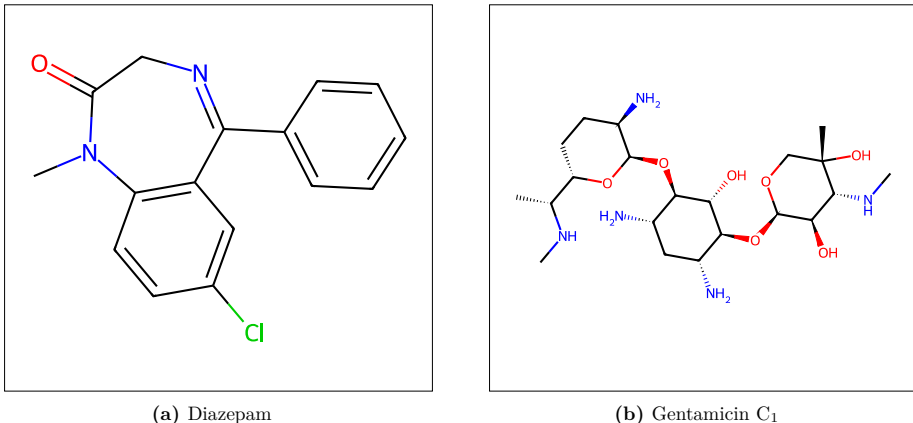


Figure 4.3: 2D visualizations of Diazepam and Gentamicin C₁ molecules.

Diazepam is moderately hydrophobic and highly lipophilic, allowing it to cross the blood-brain barrier effectively. Gentamicin is highly hydrophilic, limiting its ability to cross lipid membranes but facilitating its distribution in aqueous environments like blood and interstitial fluids.

Diazepam’s rigid structure contributes to its specific binding to GABA_A receptors, while Gentamicin’s flexibility allows it to adapt to binding sites on bacterial ribosomes. Diazepam’s aromatic rings enable $\pi-\pi$ stacking and specific molecular interactions necessary for its pharmacological effects, whereas Gentamicin’s lack of aromaticity is compensated by multiple hydrogen bonds and ionic interactions.

Figure 4.4 demonstrates an upwards trend in Tanimoto similarity as the guidance scale increases. This suggests that higher levels of guidance lead to molecular structures more similar to reference structures. We observe that the enhancement in Tanimoto similarity is accompanied by a reduction in validity, as indicated in Table 4.1. Specifically, as guidance strength σ increases from 0.1 to 0.5, the validity decreases from 0.657 to 0.444, indicating that a one-layer chemical transformer is very sensitive to small perturbations in activation. We acknowledge that determining the robustness of activation steering requires further analysis and additional evaluation on multi-layer models.

Table 4.1 also reveals that while uniqueness remains constant across different values of σ , key properties such as drug-likeness and mean LogP exhibit significant variations. Mean QED decreases from 0.7291 to 0.6553 as σ increases. Conversely, the mean LogP increases from 1.5841 to 2.0558, indicating a rise in lipophilicity, which is desired since Diazepam is highly lipophilic.

In comparison to regular SAE activation steering, we observe significant differences in performance. Restrictive steering results in higher validity, drug-likeness, and Tanimoto

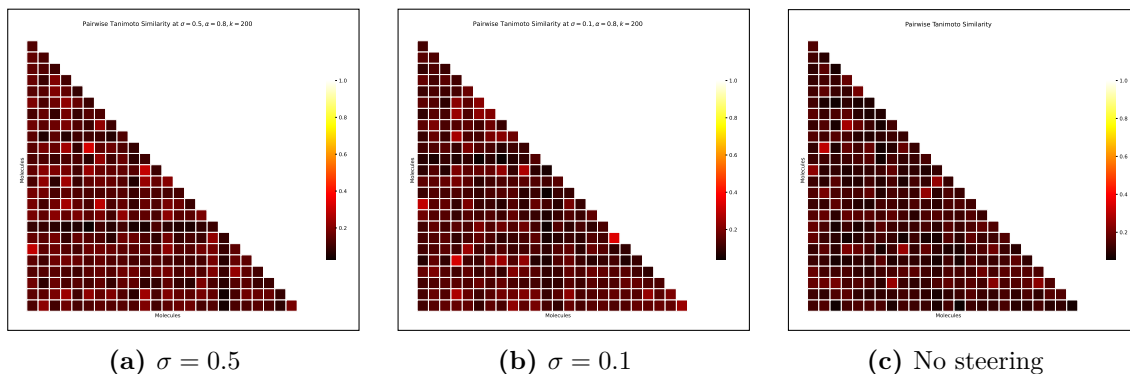


Figure 4.4: Pairwise Tanimoto similarities between generated molecules at different guidance scales using the molecule pair diazepam (+) and Gentamicin C₁ (−), compared to the baseline. Fixed hyperparameters are $\alpha = 0.8, k = 200$.

Metric	Base Model	$\sigma = 0.1$	$\sigma = 0.25$	$\sigma = 0.5$
($\uparrow$) Validity	0.6859	0.657	0.572	0.444
($\uparrow$) Uniqueness	1.0000	1.0000	1.000	1.000
($\uparrow$) Mean QED	0.7451	0.7291	0.7013	0.6553
($\downarrow$) Mean SA Score	3.2780	3.1584	3.1193	3.2152
Mean LogP	1.4182	1.5841	1.7860	2.0558

Table 4.1: Performance metrics for different values of σ , considering 1000 samples generated at unit temperature.

similarity scores, indicating higher quality samples while also conforming to guiding activations. Regular SAE steering results in significantly higher average LogP values, but reduces validity and produces less drug-like molecules. This suggest that restrictive steering can provide a more controlled approach for generating viable structures, although we acknowledge that systematic comparisons are required to determine whether the change in behavior is present across a large number of contrastive pairs.

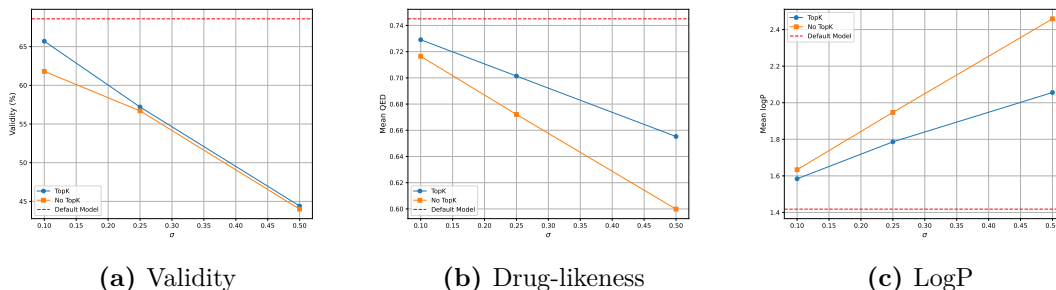


Figure 4.5: Comparison of sample quality between top- k and regular activation steering at different guidance scales using the molecule pair diazepam (+) and Gentamicin C₁ (−), compared to the baseline. Fixed hyperparameters are $\alpha = 0.8, k = 200$.

4.3 Experimental Setup

4.3.1 Language Models

Architecture: The language models used for molecule generation closely resemble the architecture of Gemma 2 (Team et al., 2024). The transformer layers use Grouped Query Attention (Ainslie et al., 2023), rotary positional encodings (RoPE) (Su et al., 2023) with $\theta = 10^4$, and apply a soft logit-cap (Bello et al., 2017; Team et al., 2024).

Model	d_{hidden}	d_{model}	Heads	KV-Heads	Layers	Parameters
Medium	4096	512	8	4	8	57M
Small	2048	256	4	4	4	7M
Toy	1024	128	4	4	1	0.5M

Table 4.2: Language Model configurations used in our experiments.

Tokenization: To convert SMILES strings into atomic representations, we use a tokenizer based on regular expressions from Schwaller et al. (2017). All token sequences are sandwiched between **<BOM>** (beginning of molecule) and **<EOM>** (end of molecule) tokens to enable variable length generation.

Training: We train all language models on a subset of the ZINC15 dataset (Sterling and Irwin, 2015) consisting of 10M molecules equating to approximately 386.6 million tokens, from which 10K samples (387K tokens) are used for evaluation only. All models are trained for 100K steps with a batch size of 4096 on a single TPU v2-8 node using the Adam optimizer (Kingma and Ba, 2017) with a constant learning rate of 0.0001. The learning rate was selected from 0.0001, 0.0005, and 0.001 based on stability and performance during sweeps. The batch size was optimized to fit the larger models into a single TPuv2-8 instance.

We do not apply transformer scaling laws when selecting dataset sizes and training hyperparameters, both for simplicity and because scaling laws for general language synthesis may not directly correspond to generating valid SMILES strings. We observe that large models quickly overfit on 10 million molecules across various learning rates, indicating that future training should utilize more extensive chemical datasets.

4.3.2 SAEs

All SAEs were trained for 10 epochs on 1M SMILES activations from the pre-trained molecular transformers. The molecules were selected from the training dataset. Toy model SAE representations are evaluated on an 500K subset of sequence-activation pairs, consisting of roughly 19M tokens, while SAEs on the 7M model are evaluated 200K molecules.

Following Bricken et al. (2023), we aim for a sparsity loss between 10 and 20 with an acceptable level of decrease in reconstruction accuracy. We train SAEs at different values of λ for each size and pick the ones that offer high sparsity at an acceptable level of decrease in sample quality.

Chapter 5

Results

5.1 Language Model Performance

We train autoregressive transformers at different parameter sizes according to Table 4.2.

Considering the performance metrics summarized in Table 5.1, the one-layer toy model achieves a significantly lower validity (0.7198) compared to the multi-layer small (0.9916) and medium (0.9924) configurations, suggesting that increased transformer capacity leads to better syntactic adherence in generated samples but saturates as the size of the language model increases. At unit temperature, models achieve 100% uniqueness uniformly, indicating that they can generate valid SMILES strings without excessive memorization.

The correlation between parameter size and SA-scores indicates that smaller models might produce structures that are marginally less synthetically accessible. We observe that QED scores are similar across models, with the toy model achieving 0.7397, the 7M model reaching 0.7417, and the medium model slightly higher at 0.7460. This suggests that even with limited parameters and transformer layers, molecular language models can capture chemical properties associated with drug-likeness, although the slight improvement in the 70M model could reflect a more complex image of chemical space.

Metric	LM-Toy	LM-Small	LM-Medium
(↑) Validity	0.7198	0.9916	0.9924
(↑) Uniqueness	1.0000	1.0000	1.000
(↑) Mean QED	0.7397	0.7417	0.7460
(↓) Mean SA Score	3.3061	3.2174	3.2312
Mean LogP	1.2235	1.1782	1.2130
Mean Molecular Weight	309.80	310.45	310.69
Mean Number of Atoms	21.822	21.904	21.919

Table 5.1: Performance Metrics for Different Models

Figure 5.1 depicts the kernel density estimates of synthetic accessibility and drug-likeness of the generated samples. The correlation between ZINC15 estimates and the KDEs of generated samples suggests that newly generated molecules closely match the training set in terms of sample quality. Although the one-layer transformer is more fragile than the multi-layer models and generates significantly fewer valid SMILES strings, the attributes of valid samples closely correspond to the training data.

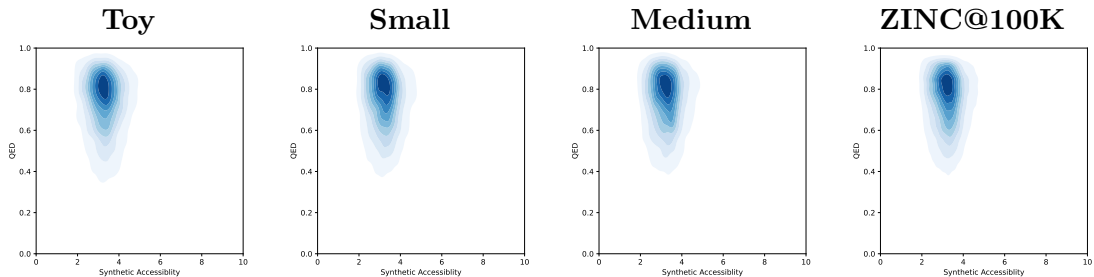


Figure 5.1: Kernel density estimates for QED and synthetic availability from 10K samples generated at unit temperature, compared to 100K samples from ZINC15.

5.2 Feature Analysis

During our experiments on the toy model, we identify a large number of highly interpretable features. We also observe that these features activate in two separate modes, indicating different functions in generating valid but diverse molecules. We test automated feature annotation using GPT-4o (see Section 3.2 for details) and find that stock language models can produce surprisingly coherent and convincing interpretations of activation patterns but fail to determine how interpretable the average feature is.

5.2.1 Modes of Activation

Our investigation identifies two distinct operational modes of SAE neurons within the 16x toy autoencoder. Firstly, we identify a significant number of low-density neurons, which activate exclusively in response to a limited number of tokens within specific contexts. Conversely, the second mode comprises high-density neurons that activate on various tokens across similar contextual environments.

We conjecture that this phenomenon has significant implications for both the interpretability and controllability of molecular language models. Low-density neurons can be effective in identifying molecular patterns during inference and can be potentially used to screen for desired or undesired behavior. Their selective activation possibly enables identifying precise structural motifs or functional groups. On the other hand, load-bearing high-density neurons may offer a mechanism for steering SMILES generation. By activating across diverse tokens, these neurons may allow for conditional molecule generation without being bound to overly specific features.

The coexistence of the two contrasting feature modes potentially enables us to explain model behavior and generate novel chemical entities at once. Low-density neurons capture very specific patterns, while high-density neurons provide the versatility needed to explore a wide chemical space. Neurons that activate in similar contexts but on a wide range of tokens may also explain the high uniqueness and validity in larger models observed in Table 5.1.

5.2.2 High-fidelity Features

A manual analysis of approximately 10% of the 2048 neurons resulted in a large number of highly coherent features that activate in specific contexts. In this section, we mostly consider features that activate on only a few tokens in specific settings. We find multiple

clusters of such low-density features, such as ring identifiers (some are active both on opening and closing IDs, and some are only active on either one or the other), rarely occurring elements (such as Bromine or Iodine), specific branches in molecules (substrings encapsulated by parentheses), identifiers of chiral centers, and substrings where the same atom (eg. carbon) is repeated a specific number of times.

Table 5.2: Neuron Annotations

Neuron ID	Summary
#1286	Detects backslashes, particularly in branches with single oxygen atoms.
#495	Identifies stereochemistry, especially oxygen atoms preceding R and S isomers.
#157	Recognizes start and end identifiers for fifth-membered rings.
#465	Detects nitrogen (n) atoms, especially in branches commencing with carbon, e.g., (C [C@] (O) (C n1 c c c c c 1)).
#785	Identifies closing identifiers of fifth-membered rings.
#1505	Detects iodine (I) atoms.
#1115	Identifies n atoms, particularly before closing rings.
#1172	Detects carbon atoms.
#895	Matches closing identifiers and chiral centers, e.g., [C@@H] 5 [C@@H] 4 [C@@H].
#407	Identifies identifiers of fifth-membered rings.
#1236	Recognizes ring identifiers, especially start IDs for fourth-membered rings.
#1200	Detects closing IDs for fifth-membered rings.
#1042	Identifies n atoms, particularly before starting first and second rings.
#903	Matches carbons with the pattern C 1 (C # N) »C« C.
#656	Detects branch-closing brackets.
#171	Identifies carbons following the pattern 5 c 4 C C C 5, indicating second rings.
#263	Detects double bonds before nitrogen in sequences like (/ C (O) = N.
#1174	Identifies identifiers for fourth and fifth-membered rings.
#1281	Detects carbon chiral centers, especially before first and second rings.
#942	Identifies fifth-membered rings consisting of five carbons, e.g., 5 c c c c c 5.
#519	Detects carbons, particularly before closing third-membered rings.
#767	Identifies bromine (Br) atoms, especially at the end of SMILES strings.
#2003	Recognizes ring identifiers, especially first rings (start and close) and start of fourth rings.
#115	Detects sequences of carbons following (C N C [C@@H] 2.

Continued on next page

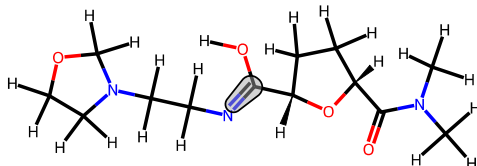
Table 5.2 – continued from previous page

Neuron ID	Summary
#1341	Identifies carbon atoms following N [C@H] 1.
#813	Detects sequences of carbons, especially after C 3 (C # N).
#1045	Identifies three or more carbons, especially around ring identifiers.
#159	Detects carbons, especially three or more sequentially, or triple bonds between two carbons.
#1170	Identifies carbons at the end of SMILES strings before the closing identifier of the first ring.
#1447	Matches N, C, =, O tokens in patterns like (N C (N) = O).
#16	Detects chiral centers.

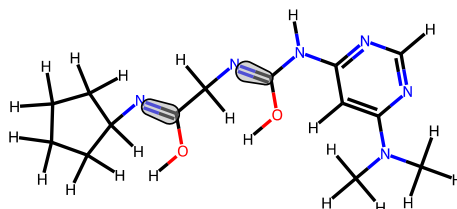
Neuron #263 (2800 activations)

6. CN(C)c1cc(N/C(O)=N/C/C(O)=N/C2CCCCC2)ncn1
9. CN(C)C(=O)[C@H]1CC[C@@H](/C(O)=N/CN2CCOCC2)O1
10. C[C@H](C(=O)N1CCC[C@H](/C(O)=N/CC2CC2)C1)c1ncccc1F
13. CN(C)C(=O)N1CCC(/C(O)=N/[C@H]2[C@@H]3COC[C@@H]32)CC1
14. NS(=O)(=O)[C@@H]1CCN(/C(O)=N/C[C@H]2CCSC2)C1

Summary: Detects double bonds before nitrogen in sequences like (/ C (O) = N.



Sequence 9



Sequence 6

Figure 5.2: Manually annotated example with highlighted active SMILES elements.

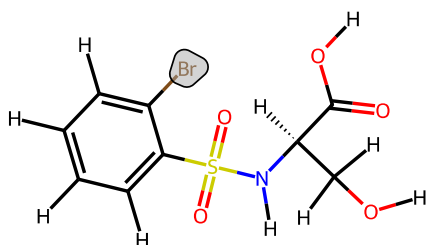
We think that monitoring this class of rarely active neurons can be helpful for explaining model behavior because they are specialized in detecting and generating particular molecular patterns, such as specific functional groups, ring structures, or chiral centers.

Low-density neurons may contribute to the validity of generated molecules by uniquely completing (or eliciting) the generation of certain molecular substructures. Consistent activation of these neurons may prevent the formation of chemically invalid structures, such as incorrect bonding configurations or improbable atom arrangements. Identifying

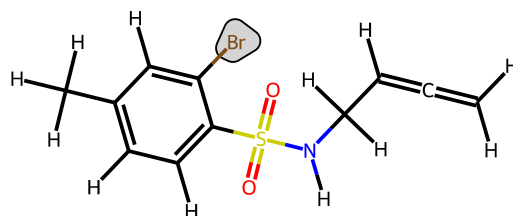
Neuron #767 (4227 activations)

1. O=C(O)[C@@H](CO)NS(=O)(=O)c1ccccc1Br
2. NC(=O)CS(=O)(=O)NC(=O)c1ccccc1Br
5. C[C@@H](C(=O)O)N(C)C(=O)c1cc(=O)n(C)cc1Br
6. C[C@H](C#N)C(=O)Nc1ccc(O)cc1Br
8. C=C=CCNS(=O)(=O)c1ccc(C)cc1Br

Summary: Identifies bromine (Br) atoms, especially at the end of SMILES strings.



Sequence 1



Sequence 8

Figure 5.3: Manually annotated example with highlighted active SMILES elements.

a large number of highly specific features that also play a significant role in generating high-quality molecules can also help in detecting overfitting and memorization.

The specialized nature of low-density neurons may also improve the controllability of the model. Increasing or decreasing the magnitudes of such neurons can help achieve fine-grained control, although given that these features activate as part of rare patterns only, such interventions may decrease uniqueness and validity.

5.3 Automated Annotation

Using the method and natural language prompt described in Section 3.2, we annotate approximately 10% of neurons in an autoencoder with 2048 latents using a general purpose language model. We find that GPT-4o produces a large number of valuable annotations, although it fails to differentiate between closely related and unrelated sets of SMILES strings. Section A.1 contains the full collection of GPT-4o annotated sparse autoencoder features.

Neuron #254 (10,182,799 activations)

```
1. CC(=N/N/C(N)=N/[N+](=O)[O-])/C(C)=N\N/C(N)=N/[N+](=O)[O-]
2. NS(=O)(=O)N1CN(S(N)(=O)=O)CN(S(N)(=O)=O)C1
3. CS(C)(=O)=NS(=O)(=O)N1CCN(CC2(C#N)CC2)CC1
4. NC(=S)N/N=C/Cn1ccc(=O)n(C/C=N\NC(N)=S)c1=O
5. CCS(=O)(CC)=NS(=O)(=O)N1CCN(CC(N)=O)C(=O)C1
```

Interpretability Score: 5

Summary: Neuron detects sulfonamide
S(=O)(=O)N motif presence in sequences.

Reasoning:

1. The common feature is the presence of sulfonamide (S(=O)(=O)N) motifs.
2. These motifs are frequent in antibiotics and sulfonamide-based drugs.
3. The patterns are structured around sulfonamide and nitrogen linkage.

Figure 5.4: Example GPT-4o annotation and the highest neuron activations of the corresponding feature.

The language model links annotated neurons to a diverse set of chemical attributes and functional groups, including:

- **Amide and Sulfonamide Linkages:** Presence of amide ($-\text{CONH}-$) and sulfonamide ($-\text{SO}_2\text{NH}-$) groups.
- **Trifluoromethyl and Nitro Groups:** Functional groups such as trifluoromethyl ($-\text{CF}_3$) and nitro ($-\text{NO}_2$) are common activation triggers.
- **Carbonyl Functional Groups**
- **Alkyne and Ester Groups**

Out of the 200 neurons annotated, the language model categorizes 22 neurons as exact identifiers of specific sequential patterns, each receiving an interpretability score of 5. Additionally, 13 latent variables are assigned a score of 3, indicating that they are interpretable, while only a single neuron is given a score of 2, signifying that it is somewhat interpretable. The remaining 164 features within the sparse autoencoder (SAE) are evaluated as highly interpretable, each assigned a score of 4. Notably, none of the neurons are classified as uninterpretable.

The assignment of high interpretability scores (4 and 5) to 186 out of 200 neurons suggests a bias in the grading system towards plausible but incorrect explanations and it may indicate that the model is overly optimistic about its ability to interpret latent features. This overrepresentation can lead to a false sense of confidence in interpretability of SAE features.

Neuron #1820 (10,837,841 activations)

1. O=[N+](O-)[O]C(CO[N+](=O)[O-])(CO[N+](=O)[O-])CO[N+](=O)[O-]
2. FC(F)C(F)(F)COCCc1nc(-c2ccc3cc[nH]c3n2)no1
3. FC(F)(F)C(F)(F)COCCN1CCC(N2CCCCC2)CC1
4. FC(F)Oc1cccc1C=NNCC(F)(F)C(F)(F)F
5. FC(F)C(F)(F)COCC1noc(Cc2nc3ncccn3n2)n1

Interpretability Score: 4

Summary: Sequences mainly contain fluorinated chains and nitro groups.

Reasoning:

1. Many sequences contain trifluoromethyl (CF₃) and perfluorinated (fluorine-rich) groups.
2. Several sequences have nitro groups, especially at aromatic rings.
3. There are frequent oxygen atoms connected through ethers or oxides.
4. Heterocycles, especially those with nitrogen, are common in these SMILES.
5. Presence of aromatic and non-aromatic carbon chains linked with fluorinated groups.

Figure 5.5: Example GPT-4o annotation and the highest neuron activations of the corresponding feature.

Chapter 6

Discussion

6.1 Applications

We argue that integrating sparse autoencoders with molecular language models offers significant potential for understanding and enhancing various stages of drug development pipelines. Given that chemical transformers are trained to model more structured sequences compared to general-purpose language models, SAEs have the potential to explain model behavior with greater clarity. Monitoring interpretable features allows researchers to identify causal relationships within the model’s operations effectively. SAEs are cheap to train and can be used as drop-in additions in existing models and pipelines with minimal effort. Steering algorithms, as outlined in Section 3.3, provide a zero-cost method for controlling sample attributes and can be applied during inference without the need to fine-tune the language model or train additional classifiers. Theoretically, steering interpretable SAE activations instead of transformer representations may enhance the understanding of model behavior and enable the model to pick up on critical attributes, rather than being perturbed by noisy and uninformative representations from the guiding examples.

6.1.1 Scientific Understanding

SAEs decompose complex molecular representations into potentially interpretable latent features. By mapping specific latent activations to molecular substructures or functional groups, researchers can explore structure-activity relationships more effectively. A detailed understanding of chemical space enables the rational modification of molecular scaffolds to enhance desired biological activities while mitigating adverse effects. For instance, identifying latent features corresponding to key pharmacophores can streamline the optimization of lead compounds.

Feature interpretability can be applied to analyze and explain existing libraries of therapeutic drugs without the need to generate new molecules or use chemical transformers for *de novo* synthesis. Using large language models for automated feature annotation, as described in Section 3.2, provides an effective approach to generating a large number of high-quality feature explanations. This approach can be leveraged to identify patterns and relationships within molecular datasets that may be overlooked by human annotators. Automated analysis offers new insights into molecular properties and their interactions, supporting knowledge discovery and the formulation of hypotheses. Understanding the chemical properties of existing molecular data can guide experimental validations and inform subsequent rounds of drug design.

6.1.2 Conditional Generation

SAE steering allows for generating molecules with predefined characteristics by manipulating specific latent features. Conditional generation through interpretable features is valuable for designing compounds with targeted properties, such as high binding affinity to specific protein targets or optimized pharmacokinetic profiles, including ADME parameters. SAE activation interventions can produce diverse libraries of compounds that satisfy multiple criteria simultaneously, while also benefiting from explainability.

Targeted Drug Design: SAEs can help in designing compounds with high binding affinity to specific protein targets by controlling latent features associated with binding motifs. This precision may allow for the optimization of drug-target interactions, increasing efficacy while reducing off-target effects.

Pharmacokinetic Optimization: by steering latent features related to pharmacokinetic properties, SAEs can generate molecules with optimized ADME profiles. This includes improving solubility, stability, and bioavailability, which are critical factors in the drug development process.

High-Throughput Virtual Screening: Activation interventions can help create extensive and diverse molecular libraries that meet predefined pharmacological and physicochemical criteria. These libraries can potentially be used in high-throughput virtual screening (Omar et al., 2021) to identify promising candidates for further experimental validation, thus accelerating lead identification.

Scaffold Hopping: Scaffold hopping (Sun et al., 2012) is essential for avoiding intellectual property conflicts and minimizing off-target effects associated with structurally similar compounds. Guided models may be able to generate novel scaffolds that retain essential biological activities while introducing structural diversity.

Algorithms for *in silico* drug development frequently incorporate multi-objective optimization frameworks to evaluate and prioritize compounds based on multiple combined criteria, often including properties such as potency, selectivity, and favorable metabolic profiles. SAE activation steering (and activation steering in general) can be easily adapted to numerous combinatorial and gradient-based optimization methods. Activation steering can be integrated with commonly used molecule optimization algorithms, such as search-based methods (e.g., Sun et al. (2022)) and reinforcement learning approaches, including those demonstrated in Loeffler et al. (2024), or with AlphaZero-style Monte Carlo Tree Search (MCTS) (Silver et al., 2017), to iteratively improve the quality of generated samples.

6.2 Related Work

Our research builds on a large corpus of language modeling and interpretability research. The primary influences of this work are the Gemma 2 family of models (Team et al., 2024), early research on dictionary learning with sparse autoencoders (Bricken et al., 2023; Templeton et al., 2024), and contrastive activation steering, particularly activation addition (Turner et al., 2024). We take inspiration from Nanda et al. (2024) and Templeton et al. (2024) in using SAE features to steer models.

Here, we list some of the related works that are adjacent to our research. This list is meant as a starting point for exploring the applications of generative artificial intelligence to *de novo* drug development and should not be viewed as an exhaustive resource. Readers are encouraged to use it as a springboard for further research.

Machine learning methods are becoming widespread in most subfields of therapeutics. [Tang et al. \(2024\)](#) provides a comprehensive overview of current directions and key concepts in generative modeling for *de novo* drug development, including a review of popular datasets, network architectures, and modeling tasks.

Generative Models in Therapeutics: Generative modeling is frequently applied to design new chemical structures for therapeutic applications. These models use various architectures to generate molecules with specific properties. Common architectures include variational autoencoders, as discussed in [Gómez-Bombarelli et al. \(2018\)](#); [Hu et al. \(2023\)](#), generative adversarial networks presented in [Zhao et al. \(2021\)](#), and recurrent neural networks, as described in [Popova et al. \(2019\)](#); [Zhang et al. \(2021\)](#). Junction Tree Variational Autoencoders ([Jin et al., 2018](#)) have achieved significant performance in generating molecular structures and are one of the most popular choices for generative chemistry. Graph neural networks are commonly applied to molecular generation tasks (e.g., [Wang et al. \(2022\)](#); [Bongini et al. \(2020\)](#)) for their explainability. Recently, certain types of diffusion models gained popularity for their ability to model chemical structures in 3 dimensions. Equivariant diffusion ([Hoogeboom et al., 2022](#)) is employed to produce three-dimensional molecular structures and [Xu et al. \(2022\)](#) utilize geometric diffusion for creating molecular conformers.

Generative models are used in a wide range of conditional generation tasks. Among others, popular conditional architectures include latent diffusion ([Chang and Ye, 2024](#)), graph diffusion transformers ([Liu et al., 2024](#)), and Spanning Tree-Based Graph Generation ([Ahn et al., 2022](#)).

Molecular Language Models: Experiments in [Flam-Shepherd et al. \(2022\)](#) demonstrate that RNN-based language models can learn complex molecular distributions even outperforming models based on graphical representations. The success of the transformer architecture inspired a large body of scientific work in molecular generation. Transformers are used both as standalone molecular generators and in conjunction with other neural architectures, such as graph neural networks, e.g. in [Wang et al. \(2023\)](#). Decoder-style molecular language models such as [Ross et al. \(2024\)](#) have exhibited high an ability to generate high quality samples. [Chaves et al. \(2024\)](#) train an autoregressive language model on multimodal therapeutics data with 540B parameters, including text and SMILES strings. Encoder-style chemical language models, such as [Liu et al. \(2023\)](#) are commonly used for embedding molecules into a high-dimensional latent space.

6.3 Limitations

In our study, SAEs were trained on only 10% of the language model’s training data and subsequently evaluated on a just 5% of the available data due to the high demand in disk space. This restricted data use potentially limits the SAE’s exposure to diverse molecular structures, resulting in a constrained range of interpretability. The majority of this work is confined to evaluating a one-layer transformer model, leaving the application of SAEs on multi-layer models with larger capacities largely unexplored.

Our proposed method for automating feature annotation requires human supervision to ensure the accuracy of GPT-generated annotations. Although language models are promising tools for producing high-fidelity feature descriptions, human involvement remains necessary for validating context-sensitive evaluations of molecular structures.

In Section 4.2, we only consider a single contrastive pair to evaluate conditional generation. This limits insights into the generalizability of activation steering across more diverse molecular pairs.

Larger transformers, despite their capacity to generate high-quality structures, quickly overfit on 10 million molecules from ZINC. This suggests that training should involve much more molecules, and also that hyperparameters should be further optimized and additional regularization should be introduced to prevent excessive memorization.

6.4 Future Work

Future work should address these limitations by scaling up to more comprehensive molecular datasets, potentially including hundreds of millions of molecules. The use of advanced tokenizers, such as byte-pair encodings (Sennrich et al., 2016), should also be considered for training larger networks. This investigation should include an analysis of how more complex tokenization affects interpretability and SAE activations working on different chemical attributes.

Expanding sparse autoencoder architectures for larger transformer models and applying them to multi-layer transformers could reveal more complex feature hierarchies and could open a door to applications in large-scale drug development pipelines.

The automation of GPT-based annotation should be refined by introducing a more sophisticated grading system that can critically evaluate relationships between SMILES sequences. Adjusting criteria to grade interpretability by molecular function rather than activation frequency alone may yield more insightful chemical explanations and better align model outputs with human intuition.

For conditional generation, more systematic analyses using diverse contrastive pairs can offer a clearer understanding of how specific properties—like hydrophilicity or target binding affinity—affect molecular generation. Additionally, implementing inference-time optimization methods, such as MCTS and gradient-based optimization, may enhance the steering capabilities of autoencoders and allow for finer control of molecular outputs based on pharmacological objectives.

Exploring improved SAE architectures, such as JumpReLU (Rajamanoharan, Lieberum, Sonnerat, Conmy, Varma, Kramár and Nanda, 2024), TopK (Gao et al., 2024), and Gated SAEs (Rajamanoharan, Conmy, Smith, Lieberum, Varma, Kramár, Shah and Nanda, 2024), may further refine feature extraction and sparsity control, particularly in high-capacity models.

Acknowledgements

We acknowledge the use of language models for gluing together experiments, such as creating plots and writing utility functions. We clarify that no AI tools were used in implementing our models or developing and analyzing the scientific contributions of this work.

This research was funded by the National Research, Development, and Innovation Fund of Hungary under Grant TKP2021-EGA-02, the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory.

Bibliography

- Ahn, S., Chen, B., Wang, T. and Song, L. (2022), Spanning tree-based graph generation for molecules, *in* ‘International Conference on Learning Representations’.
URL: https://openreview.net/forum?id=w60btE_8T2m
- Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F. and Sanghai, S. (2023), ‘Gqa: Training generalized multi-query transformer models from multi-head checkpoints’.
URL: <https://arxiv.org/abs/2305.13245>
- Arora, S., Li, Y., Liang, Y., Ma, T. and Risteski, A. (2018), ‘Linear Algebraic Structure of Word Senses, with Applications to Polysemy’, *Transactions of the Association for Computational Linguistics* **6**, 483–495.
URL: https://doi.org/10.1162/tacl_a_00034
- Ba, J. L., Kiros, J. R. and Hinton, G. E. (2016), ‘Layer normalization’.
URL: <https://arxiv.org/abs/1607.06450>
- Bello, I., Pham, H., Le, Q. V., Norouzi, M. and Bengio, S. (2017), ‘Neural combinatorial optimization with reinforcement learning’.
URL: <https://arxiv.org/abs/1611.09940>
- Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S. and Hopkins, A. L. (2012), ‘Quantifying the chemical beauty of drugs’, *Nature Chemistry* **4**(2), 90–98.
URL: <https://doi.org/10.1038/nchem.1243>
- Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J. and Saunders, W. (2023), ‘Language models can explain neurons in language models’, <https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html>.
- Bongini, P., Bianchini, M. and Scarselli, F. (2020), ‘Molecular graph generation with graph neural networks’, *arXiv preprint arXiv:2012.07397*.
- Bricken, T., Marcus, J., Mishra-Sharma, S., Tong, M., Perez, E., Sharma, M., Rivoire, K. and Henighan, T. (2024), ‘Using dictionary learning features as classifiers’. Accessed: 3/11/2024.
URL: <https://transformer-circuits.pub/2024/features-as-classifiers/index.html>
- Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T. and Olah, C. (2023), ‘Towards monosemanticity: Decomposing language models with dictionary learning’, *Transformer Circuits Thread*. <https://transformer-circuits.pub/2023/monosemantic-features/index.html>.

- Chang, Z. and Ye, Y. (2024), ‘Ldmol: Text-conditioned molecule generation with latent diffusion models’, *arXiv preprint arXiv:2405.17829*.
- Chaves, J. M. Z., Wang, E., Tu, T., Vaishnav, E. D., Lee, B., Mahdavi, S. S., Semturs, C., Fleet, D., Natarajan, V. and Azizi, S. (2024), ‘Tx-llm: A large language model for therapeutics’.
URL: <https://arxiv.org/abs/2406.06316>
- Chen, Y., Li, G. and Jin, Z. (2017), Learning sparse overcomplete word vectors without intermediate dense representations, in G. Li, Y. Ge, Z. Zhang, Z. Jin and M. Blumenstein, eds, ‘Knowledge Science, Engineering and Management’, Springer International Publishing, Cham, pp. 3–15.
- Cunningham, H., Ewart, A., Riggs, L., Huben, R. and Sharkey, L. (2023), ‘Sparse autoencoders find highly interpretable features in language models’.
URL: <https://arxiv.org/abs/2309.08600>
- Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M. and Olah, C. (2022), ‘Toy models of superposition’, *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy_model/index.html.
- Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S. and Olah, C. (2021), ‘A mathematical framework for transformer circuits’, *Transformer Circuits Thread*. <https://transformer-circuits.pub/2021/framework/index.html>.
- Ertl, P. and Schuffenhauer, A. (2009), ‘Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions’, *Journal of Cheminformatics* **1**(1), 8.
URL: <https://doi.org/10.1186/1758-2946-1-8>
- Flam-Shepherd, D., Zhu, K. and Aspuru-Guzik, A. (2022), ‘Language models can learn complex molecular distributions’, *Nature Communications* **13**.
- Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J. and Wu, J. (2024), ‘Scaling and evaluating sparse autoencoders’.
URL: <https://arxiv.org/abs/2406.04093>
- Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P. and Aspuru-Guzik, A. (2018), ‘Automatic chemical design using a data-driven continuous representation of molecules’, *ACS Central Science* **4**(2), 268–276.
- Hoogeboom, E., Garcia Satorras, V., Vignac, C. and Welling, M. (2022), Equivariant diffusion for molecule generation in 3d, in ‘International Conference on Machine Learning’, PMLR, pp. 8867–8887.
- Hu, C., Li, S., Yang, C., Chen, J., Xiong, Y., Fan, G., Liu, H. and Hong, L. (2023), ‘Scaffoldgvaes: Scaffold generation and hopping of drug molecules via a variational autoencoder based on multi-view graph neural networks’, *Journal of Cheminformatics* **15**(1), 91.

- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W. (2021), ‘Lora: Low-rank adaptation of large language models’.
URL: <https://arxiv.org/abs/2106.09685>
- Jin, W., Barzilay, R. and Jaakkola, T. (2018), Junction tree variational autoencoder for molecular graph generation, in J. Dy and A. Krause, eds, ‘Proceedings of the 35th International Conference on Machine Learning’, Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 2323–2332.
URL: <https://proceedings.mlr.press/v80/jin18a.html>
- Kingma, D. P. and Ba, J. (2017), ‘Adam: A method for stochastic optimization’.
URL: <https://arxiv.org/abs/1412.6980>
- Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K., Bishop, C., Hall, E., Carbune, V., Rastogi, A. and Prakash, S. (2024), ‘Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback’.
URL: <https://arxiv.org/abs/2309.00267>
- Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A., Shah, R. and Nanda, N. (2024), ‘Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2’.
URL: <https://arxiv.org/abs/2408.05147>
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I. and Cobbe, K. (2023), ‘Let’s verify step by step’.
URL: <https://arxiv.org/abs/2305.20050>
- Liu, G., Xu, J., Luo, T. and Jiang, M. (2024), ‘Graph diffusion transformers for multi-conditional molecular generation’.
URL: <https://arxiv.org/abs/2401.13858>
- Liu, Y., Zhang, R., Li, T., Jiang, J., Ma, J. and Wang, P. (2023), ‘Molrope-bert: An enhanced molecular representation with rotary position embedding for molecular property prediction’, *Journal of Molecular Graphics and Modelling* **118**, 108344.
URL: <https://www.sciencedirect.com/science/article/pii/S1093326322002236>
- Loeffler, H. H., He, J., Tibo, A., Janet, J. P., Voronov, A., Mervin, L. H. and Engkvist, O. (2024), ‘Reinvent 4: Modern ai-driven generative molecule design’, *Journal of Cheminformatics* **16**(1), 20.
URL: <https://doi.org/10.1186/s13321-024-00812-5>
- Meng, K., Bau, D., Andonian, A. and Belinkov, Y. (2023), ‘Locating and editing factual associations in gpt’.
URL: <https://arxiv.org/abs/2202.05262>
- Nanda, N., Conmy, A., Smith, L., Rajamanoharan, S., Lieberum, T., Kramár, J. and Varma, V. (2024), ‘Progress update #1 from the gdm mech interp team’. Accessed: 2024-11-02.
URL: <https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/progress-update-1-from-the-gdm-mech-interp-team-full-update>
- Niazi, S. K. and Mariam, Z. (2024), ‘Computer-aided drug design and drug discovery: A prospective analysis’, *Pharmaceuticals* **17**(1).
URL: <https://www.mdpi.com/1424-8247/17/1/22>

- O’Boyle, N. M. (2012), ‘Towards a universal smiles representation - a standard method to generate canonical smiles based on the inchi’, *Journal of Cheminformatics* **4**(1), 22.
URL: <https://doi.org/10.1186/1758-2946-4-22>
- Omar, O. H., del Cueto, M., Nemataram, T. and Troisi, A. (2021), ‘High-throughput virtual screening for organic electronics: a comparative study of alternative strategies’, *J. Mater. Chem. C* **9**, 13557–13583.
URL: <http://dx.doi.org/10.1039/D1TC03256A>
- OpenAI (2024a), ‘Hello gpt 4o’, <https://openai.com/index/hello-gpt-4o/>. Accessed: 2024-11-01.
- OpenAI (2024b), ‘Openai o1 system card’, <https://openai.com/index/openai-o1-system-card/>. Accessed: 2024-11-01.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J. and Lowe, R. (2022), ‘Training language models to follow instructions with human feedback’.
URL: <https://arxiv.org/abs/2203.02155>
- Park, K., Choe, Y. J. and Veitch, V. (2024), ‘The linear representation hypothesis and the geometry of large language models’.
URL: <https://arxiv.org/abs/2311.03658>
- Piantadosi, S. T., Muller, D. C., Rule, J. S., Kaushik, K., Gorenstein, M., Leib, E. R. and Sanford, E. (2024), ‘Why concepts are (probably) vectors’, *Trends in Cognitive Sciences* **28**(9), 844–856.
URL: <https://www.sciencedirect.com/science/article/pii/S1364661324001712>
- Popova, M., Shvets, M., Oliva, J. and Isayev, O. (2019), ‘Molecularrnn: Generating realistic molecular graphs with optimized properties’, *arXiv preprint arXiv:1905.13372*.
- Radford, A. and Narasimhan, K. (2018), Improving language understanding by generative pre-training.
URL: <https://api.semanticscholar.org/CorpusID:49313245>
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D. and Finn, C. (2024), ‘Direct preference optimization: Your language model is secretly a reward model’.
URL: <https://arxiv.org/abs/2305.18290>
- Rai, D., Zhou, Y., Feng, S., Saparov, A. and Yao, Z. (2024), ‘A practical review of mechanistic interpretability for transformer-based language models’.
URL: <https://arxiv.org/abs/2407.02646>
- Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R. and Nanda, N. (2024), ‘Improving dictionary learning with gated sparse autoencoders’.
URL: <https://arxiv.org/abs/2404.16014>
- Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J. and Nanda, N. (2024), ‘Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders’.
URL: <https://arxiv.org/abs/2407.14435>

- Rogers, D. and Hahn, M. (2010), ‘Extended-connectivity fingerprints’, *Journal of Chemical Information and Modeling* **50**(5), 742–754.
URL: <https://doi.org/10.1021/ci100050t>
- Ross, J., Belgodere, B., Hoffman, S. C., Chenthamarakshan, V., Mroueh, Y. and Das, P. (2024), ‘Gp-molformer: A foundation model for molecular generation’.
URL: <https://arxiv.org/abs/2405.04912>
- Saphra, N. and Wiegrefe, S. (2024), ‘Mechanistic?’.
URL: <https://arxiv.org/abs/2410.09087>
- Scannell, J. W., Blanckley, A., Boldon, H. and Warrington, B. (2012), ‘Diagnosing the decline in pharmaceutical r&d efficiency’, *Nature Reviews Drug Discovery* **11**(3), 191–200.
URL: <https://doi.org/10.1038/nrd3681>
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017), ‘Proximal policy optimization algorithms’.
URL: <https://arxiv.org/abs/1707.06347>
- Schwaller, P., Gaudin, T., Lanyi, D., Bekas, C. and Laino, T. (2017), “found in translation”: Predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models’.
URL: <https://arxiv.org/abs/1711.04810>
- Sennrich, R., Haddow, B. and Birch, A. (2016), ‘Neural machine translation of rare words with subword units’.
URL: <https://arxiv.org/abs/1508.07909>
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K. and Hassabis, D. (2017), ‘Mastering chess and shogi by self-play with a general reinforcement learning algorithm’.
URL: <https://arxiv.org/abs/1712.01815>
- Simoens, S. and Huys, I. (2021), ‘R&D costs of new medicines: A landscape analysis’, *Frontiers in Medicine* **8**, 760762.
URL: <https://www.frontiersin.org/articles/10.3389/fmed.2021.760762/full>
- Sterling, T. and Irwin, J. (2015), ‘Zinc 15 - ligand discovery for everyone’, *Journal of chemical information and modeling* **55**.
- Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B. and Liu, Y. (2023), ‘Roformer: Enhanced transformer with rotary position embedding’.
URL: <https://arxiv.org/abs/2104.09864>
- Subramani, N., Suresh, N. and Peters, M. E. (2022), ‘Extracting latent steering vectors from pretrained language models’.
URL: <https://arxiv.org/abs/2205.05124>
- Sun, H., Tawa, G. and Wallqvist, A. (2012), ‘Classification of scaffold-hopping approaches’, *Drug discovery today* **17**(7-8), 310–324.
URL: <https://europepmc.org/articles/PMC3328312>
- Sun, M., Xing, J., Meng, H., Wang, H., Chen, B. and Zhou, J. (2022), Molsearch: Search-based multi-objective molecular generation and property optimization, in ‘Proceedings

- of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining', KDD '22, Association for Computing Machinery, New York, NY, USA, p. 4724–4732.
URL: <https://doi.org/10.1145/3534678.3542676>
- Tang, X., Dai, H., Knight, E., Wu, F., Li, Y., Li, T. and Gerstein, M. (2024), 'A survey of generative ai for de novo drug design: New frontiers in molecule and protein generation'.
URL: <https://arxiv.org/abs/2402.08703>
- Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozińska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucińska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J. P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., yeong Ji, J., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L., McNealus, L., Soares, L. B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson, M., Görner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R. A., Agarwal, R., Mullins, R., Saadat, S., Carthy, S. M., Cogan, S., Perrin, S., Arnold, S. M. R., Krause, S., Dai, S., Garg, S., Sheth, S., Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kocisky, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen, Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R. and Andreev, A. (2024), 'Gemma 2: Improving open language models at a practical size'.
URL: <https://arxiv.org/abs/2408.00118>
- Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Summers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C. and Henighan, T. (2024), 'Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet', *Transformer Circuits Thread*.
URL: <https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html>
- Terstappen, G. C. and Reggiani, A. (2001), 'In silico research in drug discovery', *Trends in Pharmacological Sciences* **22**(1), 23–26.
URL: <https://www.sciencedirect.com/science/article/pii/S0165614700015844>

- Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U. and MacDiarmid, M. (2024), ‘Steering language models with activation engineering’.
URL: <https://arxiv.org/abs/2308.10248>
- U.S. Food and Drug Administration (2022), ‘The drug development process’. Accessed: 2024-11-03.
URL: <https://www.fda.gov/patients/learn-about-drug-and-device-approvals/drug-development-process>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2023), ‘Attention is all you need’.
URL: <https://arxiv.org/abs/1706.03762>
- Wang, Y., Li, Z. and Barati Farimani, A. (2022), ‘Graph neural networks for molecules’, *arXiv preprint arXiv:2209.05582*.
- Wang, Y., Pang, C., Wang, Y., Jin, J., Zhang, J., Zeng, X., Su, R., Zou, Q. and Wei, L. (2023), ‘Retrosynthesis prediction with an interpretable deep-learning framework based on molecular assembly tasks’, *Nature Communications* **14**(1), 6155.
URL: <https://doi.org/10.1038/s41467-023-41698-5>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. and Zhou, D. (2023), ‘Chain-of-thought prompting elicits reasoning in large language models’.
URL: <https://arxiv.org/abs/2201.11903>
- Weininger, D. (1988), ‘Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules’, *Journal of Chemical Information and Computer Sciences* **28**(1), 31–36.
URL: <https://doi.org/10.1021/ci00057a005>
- Weininger, D., Weininger, A. and Weininger, J. L. (1989), ‘Smiles. 2. algorithm for generation of unique smiles notation’, *Journal of chemical information and computer sciences* **29**(2), 97–101.
- Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S. and Tang, J. (2022), ‘Geodiff: a geometric diffusion model for molecular conformation generation’, *arXiv preprint arXiv:2203.02923*.
- Zhang, B. and Sennrich, R. (2019), ‘Root mean square layer normalization’.
URL: <https://arxiv.org/abs/1910.07467>
- Zhang, Y., Wang, Y. and Wang, J. (2021), ‘Improving de novo molecule generation by embedding lstm and attention mechanism in cyclegan’, *Frontiers in Genetics* **12**, 709500.
- Zhao, Y., Yang, J., Wang, Y. and Wang, J. (2021), Gans for molecule generation in drug design and discovery, in ‘Generative Adversarial Networks: Advanced Techniques and Applications’, Springer, pp. 235–256.

Appendix

A.1 List of Annotated Features

Table A.1: Activation annotations generated by GPT-4o on the 200 most frequently activating residual stream SAE neurons.

Neuron ID	Interpretability Score	Total Activations	Summary
#838	5	14,335,266	Neuron fires on nitrogen-rich, heterocyclic aromatic compounds with linking groups.
#682	5	14,117,890	Neuron detects cyclopropane rings and amide groups consistently.
#1889	5	13,510,007	Frequent identification of fluorinated carbons and cyclic amide linkages.
#808	5	12,779,795	Activation caused by chiral centers, aromatic rings and nitrogen groups.
#68	5	12,110,513	Firing based on carbonyl groups adjacent to heterocycles.
#1139	5	11,931,712	Trifluoromethyl groups frequently cause neuron activation in SMILES sequences.
#1554	5	11,511,736	Neuron activates on sulfonamide-containing molecules.
#714	5	10,947,274	Neuron activates on bicyclic nitrogen-containing heterocycles in molecules.
#492	5	10,823,513	Neuron is activated by compounds with chiral amides and cyclic structures.
#487	5	10,718,013	Triggered by alkyne and certain N-cyclic structures.
#905	5	10,621,497	Neuron fires on molecules containing nitro functional groups prominently.
#787	5	10,496,106	Strong activation with chlorine-rich structures, both aromatic and aliphatic.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#254	5	10,182,799	Neuron detects sulfonamide S(=O)(=O)N motif presence in sequences.
#1206	5	9,836,821	Neurons fire on nitro groups and aromatic aromatic rings.
#1777	5	9,496,563	Activation is caused by nitrogen-containing heterocycles with chiral centers.
#437	5	9,218,521	Neurons activate based on sulfonamide groups and aromatic systems.
#1630	5	8,786,477	Neurons identify cyclic and macrocyclic tautomeric lactam/lactone structures.
#385	5	8,405,214	Neuron activation due to nitro groups in structures.
#1569	5	8,167,644	Activation by trifluoromethyl and bulky structure features observed.
#178	5	7,903,809	Firing prioritizes 5-membered rings containing nitrogen and oxygen.
#1860	5	7,811,294	Neuron detects fluorinated groups and amide linkages in heterocycles.
#1986	5	7,104,430	Common structural motifs: sulfonamides, amides, cyclic, stereochemistry.
#946	4	15,683,313	Neurons activated by polycyclic structures with multiple carbonyl and amide groups.
#1734	4	14,754,900	Neurons fire on structures with complex stereochemistry and conjugated systems.
#431	4	14,745,007	Neurons fire on nitrogen-containing heterocycles and amide linkages.
#850	4	14,607,511	Activation linked to cyclic amide and bicyclic structures with carbonyl.
#167	4	14,540,445	Strained cyclic structures, alkynes, and tertiary amines dominate reactions.
#1594	4	14,091,796	Focus on methoxyphenyl and carbonyl group presence in structures.
#392	4	13,959,108	Chiral centers and nitro groups dominate activations.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#872	4	13,854,308	Common substructures: nitro groups, heterocycles with carbonyls/sulfonyls.
#1227	4	13,827,253	Neuron fires on heteroaromatic rings and stereochemistry patterns.
#641	4	13,668,066	Molecules contain sulfonamides, nitro groups, and heteroatomic structures.
#466	4	13,566,433	Neuron identifies structures with cyclic amides and diverse substitutions.
#536	4	13,547,433	Triggers on isopropyl/tert-butyl lactam compounds.
#1548	4	13,440,570	Activation caused by nitrogen heterocycles, tertiary amines, and stereochemistry.
#1471	4	13,362,574	Frequent oxygen and aromatic rings, suggesting ether or alcohol motifs.
#1183	4	13,320,926	Neuron activation on peptide and cyclic structure features.
#1394	4	13,195,324	Sequences often contain trifluoromethyl groups on aromatic rings.
#1614	4	13,164,116	Neuron identifies aromatic heterocycles; pyridine and oxazoles emphasized.
#1153	4	13,072,367	Neurons activate on polycyclic, hydroxylated molecules with stereocenters.
#396	4	13,047,470	Neuron activation indicates presence of nitrogen in cyclic structures.
#904	4	12,860,326	Frequent ester and amide functional groups with steric branching.
#555	4	12,840,367	Neuron activates on patterns related to amide and peptide bonds.
#1050	4	12,794,834	Chiral molecules with complex stereochemistry and diverse functional groups.
#218	4	12,697,268	Neuron fires on aromatic rings and nitro groups.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#1316	4	12,687,619	Activation centers on nitrogen heterocycles and secondary amide structures.
#614	4	12,676,255	Neuron activates on polycyclic structures with amines and amides.
#1760	4	12,582,517	Neuron activates on linear chains, chiral centers, amides, heterocycles.
#340	4	12,540,669	Nitrogen-containing heterocyclic rings with conjugated systems trigger activation.
#1944	4	12,470,330	Frequent trifluoromethyl and chiral centers suggest activation causality.
#1362	4	12,400,786	Neurons activate on structures with aromatic rings and nitro groups.
#71	4	12,318,389	Tertiary carbons connected to carbonyl and nitrogen groups.
#1202	4	12,292,699	Neuron identifies aromatic heterocycles with amide linkages.
#1952	4	12,208,291	Activation triggered by amide bonds and chiral center presence.
#193	4	12,074,223	Activation due to chiral centers and tertiary amines.
#1799	4	12,032,501	Focuses on amines and cyclic structures with nitrogen atoms.
#1020	4	12,031,351	Neuron activates on aromatic nitrogen-containing heterocyclic rings.
#1054	4	11,982,558	Activation often occurs with cyclic structures and chirality features.
#28	4	11,967,388	Activated by amides with conjugated and cyclic heterocyclic structures.
#474	4	11,961,490	Neuron activated by nitro groups and specific heterocyclic patterns.
#732	4	11,945,897	Neuron fires on cyclic amides with complex, stereochemical structures.
#418	4	11,923,361	Neuron fires on trifluoromethyl (C(F)(F)F) and carbonyl presence.
#577	4	11,873,304	Sequences often contain N-substituted amide groups in carbon chains.
#168	4	11,755,377	Sensitive to nitrogen-rich heterocyclic and nitro/nitroso groups.
#33	4	11,737,422	Neurons fire on lactam rings with adjacent chiral centers.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#590	4	11,645,994	Neurons likely recognize chiral centers with specific substituents.
#979	4	11,563,171	Neurons highly activate on nitro groups in molecular structures.
#1870	4	11,527,636	Neuron activation is triggered by carbohydrates and chiral centers.
#1408	4	11,485,779	Strong activation by tertiary groups with trifluoromethyl substitutions.
#1660	4	11,462,967	Activated by aromatic structures with nitrogen and nitrile groups.
#289	4	11,407,904	Activated by carbonyls, heterocycles, and nitro or sulfonamide groups.
#869	4	11,394,644	Neuron detects conjugated double bonds and cyclic saturation patterns.
#169	4	11,371,621	Molecules contain imine or thioimine C=N/S bonds.
#1415	4	11,370,762	Prefers sequences with nitrogen heterocycles and specific chiral centers.
#61	4	11,282,420	Neuron activates on nitrogen heterocycles and carbonyl-containing structures.
#208	4	11,190,749	Activation due to multiple amide linkages and cyclic structures.
#840	4	11,170,296	Neuron fires on nitro groups and bulky ester linkages.
#1656	4	11,022,230	Neuron identifies complex multi-ring heterocyclic aromatic structures.
#1956	4	10,998,439	Activation occurs on nitro groups and aromatic rings presence.
#892	4	10,990,765	Neuron fires on cyclic amides, especially lactams or related structures.
#1474	4	10,941,280	Activation related to combined amide bonds and nitrile groups presence.
#365	4	10,897,617	Frequent firing on amides and cyclic compounds with chiral centers.
#490	4	10,890,201	Neuron fires on lactam-containing heterocycles with specific stereochemistry.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#1812	4	10,871,093	Cyclic amides and stereochemistry configuration are key activation features.
#1820	4	10,837,841	Sequences mainly contain fluorinated chains and nitro groups.
#64	4	10,802,889	Neuron activates on fused nitrogen-containing heterocycles and nitrile groups.
#1088	4	10,789,035	Neurons identified N-C bonds with nitrogen-containing groups and tautomerism.
#1277	4	10,747,417	Fires on molecules with amides and ring structures.
#1771	4	10,705,700	Activation generally linked to chiral centers and complex bicyclic systems.
#1726	4	10,623,522	Frequent activation on bicyclic amidine structures with N-substitution.
#460	4	10,618,296	Neurons fired on SMILES with tertiary alkyl structures.
#856	4	10,463,378	Neurons fire on acetylenic and aromatic structures with nitrogen atoms.
#1701	4	10,453,497	Neuron activates on tertiary amide with cyclic structure in sequences.
#1907	4	10,446,048	Compounds predominantly feature aromatic rings with acylated nitrogen groups.
#1858	4	10,432,906	Fires on nitrogen-containing rings, amide linkages, and branching.
#1085	4	10,421,268	Neuron activation likely by nitrogen-containing heterocycles in these molecules.
#1274	4	10,413,562	Neuron fires on chiral centers with bulky groups nearby.
#994	4	10,372,360	Recognition of pyrrole-like structures and thiazole-like systems.
#1962	4	10,312,154	Activation linked to nitrogen groups and cyano functionality presence.
#17	4	10,300,828	The neuron is triggered by cyclic structures and amide linkages.
#406	4	10,218,439	Neurons activate on chiral centers and cyclic amide structures.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#861	4	10,175,177	Neuron activates on molecules with heterocycles, chiral centers, carbonyls, fluoro/nitro groups.
#653	4	10,126,602	Neuron activates on complex bicyclic N-heterocycles with carbonyl groups.
#1336	4	10,114,611	Tertiary amines linked to heterocycles seem crucial for activation.
#573	4	10,018,571	Activation caused by nitro and sulfonyl groups, plus sterics.
#554	4	9,999,170	Neuron fires on SMILES with chiral centers and amide groups.
#874	4	9,903,562	Neurons detect double bonds and amide linkages in SMILES.
#2045	4	9,801,920	Neuron responds to aromatic rings with acyl, sulfonamide, cyano groups.
#1526	4	9,672,879	Neuron activated by nitro groups and multicyclic structures in molecules.
#2034	4	9,654,051	Nitrogen heterocycles, especially piperazine, activate neuron significantly.
#267	4	9,635,721	Neuron fires on bicyclic structures with quaternary carbons.
#1738	4	9,432,455	Trifluoromethyl groups and aromatic rings contribute to neuron firing.
#501	4	9,401,523	Neuron activation likely due to stereochemistry and amide bond pattern.
#215	4	9,381,794	Neuron activates on chiral centers and amide/ester functional groups.
#1668	4	9,348,769	Neuron strongly correlates with presence of lactam and amide structures.
#37	4	9,323,934	Sequences predominantly feature cyclic amides and nitrogenous heterocycles.
#375	4	9,321,676	Fires on nitrogen heterocyclic and amide-connected molecular structures.
#58	4	9,297,390	Activated by stereochemistry, nitro groups, and conjugated carbonyl systems.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#120	4	9,284,008	Common feature is trifluoromethyl group presence and various amides.
#1619	4	9,283,839	Activated by nitrogen-containing heteroaromatic systems with complex substitutions.
#1025	4	9,242,310	Activation largely due to trifluoromethyl and cyano groups.
#1788	4	9,102,646	Neuron activation is driven by specific stereochemistry and carbonyl presence.
#164	4	9,097,246	Frequent presence of nitro-aromatic systems with complex substituents.
#1874	4	9,062,914	Sequences emphasize cyclic amine derivatives with amide and nitroso groups.
#1765	4	9,057,990	Neuron globally activates on aromatic amides and nitrogen heterocycles.
#1781	4	9,039,219	Neuron identifies chiral centers and amide-linked cyclic structures.
#701	4	9,027,446	Neuron activation likely linked to complex nitrogen-containing heterocycles.
#1710	4	9,022,241	Neurons fire on tertiary amine-containing heterocycles and nitrogen patterns.
#82	4	8,928,340	Neuron identifies bulky cyclohexane derivatives with amide linkages.
#912	4	8,883,784	Activation primarily by nitrogen heterocycles and conjugated aromatic systems.
#655	4	8,838,333	Neuron activates on chiral centers and bulky group arrangements.
#1708	4	8,750,706	Neurons fire on sulfur-containing compounds with specific substituents.
#676	4	8,748,159	Amides and nitriles frequently cause neuron activation.
#1445	4	8,732,337	Consistent presence of nitro groups and complex aromatic ring structures.
#694	4	8,694,816	Neurons trigger on oxo-linked heterocycles; notable aromaticity and amides.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#462	4	8,681,960	Activation due to presence of amide bonds and complex structures.
#1711	4	8,669,973	Focus on chiral centers and amide groups.
#217	4	8,660,106	Neuron activates on conjugated systems with carbonyl and cyclic motifs.
#19	4	8,620,205	Neuron detects nitrogenous heterocycles, especially imidazole and pyrrole, plus amides.
#1856	4	8,604,376	Activation on multi-aromatic systems with nitrogen and ether groups.
#404	4	8,557,431	Neuron fires on branched molecules with carbonyl and chiral centers.
#1253	4	8,557,246	Molecules contain carbonyl groups adjacent to aromatic or similar rings.
#337	4	8,471,867	Activations caused by sulfone and cyclic amine structures, with fluorine.
#1494	4	8,470,998	Activated by isothioureia or conjugated enamine structures.
#401	4	8,396,102	Neurons activate on sequences with amide bonds or lactams.
#214	4	8,331,983	Frequent activation by molecules with tertiary carbon centers or cyclics.
#820	4	8,272,751	Neurons respond to heterocyclic aromatic structures with nitro or nitroso groups.
#500	4	8,135,854	Neurons activate on molecules with tertiary amines and trifluoromethyl.
#230	4	8,109,915	Neurons activate on aromatic rings, prominent substituents, and amides.
#52	4	8,071,987	Fluorinated groups and nitrogenous heterocycles trigger consistent neuron activation.
#1724	4	7,988,751	Fires on nitrogen-containing heteroaromatic systems.
#1289	4	7,926,575	Firing on bicyclic aromatic systems with nitro and carbonyl groups.
#1581	4	7,908,402	Activated by amide bonded to aromatic with chiral heterocycles.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#908	4	7,886,333	Neuron fires on nitrogen heterocycles with complex chirality.
#5	4	7,884,500	Nitrogen-based functional groups and chiral centers prominently feature.
#1638	4	7,842,048	Neuron activation by cyclic amines and stereochemical annotations.
#801	4	7,800,960	Neuron targets nitrogen and oxygen heterocycles or associated substitutions.
#1912	4	7,795,500	Neurons activate on polycyclic structures with hydroxyl and carboxamide groups.
#1410	4	7,755,886	Neuron activates on branched structures with bulky substituents.
#207	4	7,713,157	Activation likely due to carbon-nitrogen double bond presence.
#1842	4	7,703,644	Neuron detects aromatic rings with amide or ester linkages.
#1825	4	7,675,272	Activated by chiral centers and aromatic ring structures.
#1586	4	7,590,379	Neuron fires on cyclic amines with carbonyl linkages and stereocenters.
#1666	4	7,539,835	Neurons activate on lactam-like structures with cyclic heterocycles.
#126	4	7,539,264	Neurons respond to nitro, fluoro, and azide groups' presence.
#899	4	7,532,768	Neuron highly activates on nitrogen-containing ring structures with stereochemistry.
#1156	4	7,528,199	Neuron fires for nitrogenated heterocycles and carbonyl groups in molecules.
#1648	4	7,478,093	Neuron activated by nitro-aromatized heterocyclic structures in SMILES strings.
#99	4	7,389,556	Focuses on amide-linked cyclic nitrogen heterocycles with substituents.
#1347	4	7,350,801	Activates on complex spirocyclic and bicyclic ring systems.
#1776	4	7,309,030	Neuron likely activates on cyclic amides and heterocyclic structures.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#967	4	7,292,152	Activation likely due to aromatic rings and nitrogen-containing functional groups.
#114	4	7,291,325	Neuron fires on chiral centers with cyclic, amide structures.
#1545	4	7,288,318	Activation on conjugated C=C double bonds or carbonyl groups.
#1484	4	7,239,561	Bicyclic/tricyclic nitrogenous structures with amide functionalities occur frequently.
#1652	4	7,216,261	Neuron activation is driven by chiral macrocyclic and amide structures.
#963	4	7,117,337	Activation primarily caused by nitro group presence in structures.
#1072	4	7,106,610	Neuron activated prominently by nitrogenous and nitro-containing aromatic structures.
#533	4	7,018,854	Neuron activation correlates with nitrogen functionalities in chemical structures.
#1268	4	6,984,059	Neuron identifies chirality and complex bicyclic or polycyclic structures.
#737	4	6,943,313	Neurons activate on molecules with chiral centers and nitrogen heterocycles.
#1122	4	6,906,624	Neuron fires on nitrogen-containing heterocycles with conjugation.
#310	4	6,882,691	Neurons fire on C=C bonds and S(=O)(=O) near N=C.
#1329	4	6,835,644	Activates on molecules with fluorinated alkyl chains and amine groups.
#957	4	6,798,752	Cyclic molecules with amide bonds consistently activate this neuron.
#544	4	6,775,548	Neuron recognizes cyclic amide structures with bulky substituents.
#1514	3	14,425,708	Neuron activation on heterocycles and oxygen-related functionalities detected.
#671	3	13,504,682	Neuron activates on complex cyclic structures with heterocycles and chiral centers.

Continued on next page

Table A.1 – continued from previous page

Neuron ID	Interpretability Score	Total Activations	Summary
#1453	3	12,305,247	Neuron responds to tetrahedral, sterically hindered cyclic amine arrangements.
#1489	3	11,206,587	Common features are methoxy and amino groups in SMILES strings.
#177	3	10,716,672	Neuron likely detects trifluoromethyl groups and bulky substituents.
#1720	3	10,465,084	Activated by conjugated systems and geometric isomerism.
#46	3	10,224,183	Neurons activate on molecules with complex stereochemistry and aromaticity.
#379	3	9,455,414	Neuron activates on hydroxylated carbohydrate and fluorinated cyclic structures.
#1546	3	9,314,098	Activation linked to sulfonamide groups and heterocycles containing nitrogen.
#1863	3	7,684,831	Sequences feature chiral centers and nitrogen-containing functional groups.
#717	3	7,555,941	Neuron activates on carboxylic acids and aromatic rings.
#1882	3	6,981,722	Neuron recognizes fluorinated hydrocarbons and cyclic structures in SMILES strings.
#489	3	6,941,828	Neurons likely target aromatic heterocycles with specific substitution patterns.
#1187	2	11,284,726	Neuron activation by amides with diverse heterocycles.

