

# KAN-LLaMA: An Interpretable Large Language Model With KAN-based Sparse Autoencoders

CSE 5525 Final Project Proposal

Alex Gulko  
Computer Science and Engineering  
The Ohio State University  
Columbus, OH  
gulko.5@osu.edu

Yusen Peng  
Computer Science and Engineering  
The Ohio State University  
Columbus, OH  
peng.1007@buckeyemail.osu.edu

## I. INTRODUCTION

Interpretability has been a dominant challenge in a broad range of machine learning applications. In addition to enhanced feature visualization for users and developers, developing trustworthy interpretation is crucial to high-stakes real world fields including medicine and scientific computing [1]. For example, in time series analysis, XCM, an explainable convolutional neural network, was proposed to generate attribution maps, which are heat maps that indicate features that contribute positively to target activation [2]. Similarly, Large Language Models (LLMs) have been enriched with interpretability considerations in a variety of applications [1].

Two primary directions in terms of LLM interpretability have been sufficiently investigated recently: explaining a dataset using LLMs; explaining an existing LLM itself [1]. In terms of effectively explaining an entire dataset, LIDA, an LLM-based agent, was proposed to generate language visualizations and corresponding infographics [3]. On the other hand, in order to explain an LLM itself, a variety of strategies have been employed. For instance, a novel attribution mechanism, Integrated Gradients, has been applied to extract relevant rules within both image and text models [4]. Another example is to perform probing upon individual attention heads of LLMs to interpret Large Language Models as a whole to achieve global interpretation [5]. In general, the model implementation of interpretability in LLMs can be divided into two parts: inherently interpretable models; models with post-hoc interpretability techniques [1]. Multiple attempts in combining interpretable models with expressive neural networks have been accomplished recently [6]. One classic example is the LLM-based symbolic programs, also known as LSPs, which combined the pre-trained LLM with symbolic conditional branching, revealing the fact that natural language prompts can serve as a major source of interpretable neural network modules [6].

A large number of libraries and packages have become available to integrate interpretability into existing large lan-

guage models. Ecco, a library that focuses on interactive visualization of transformer-based large language models, supports many interpretable tasks, including viewing probabilistic predictions for the next token, ranking across different layers, and reporting input attributions to the output token [7]. Inseq, a library dedicated to attribution visualization, provides multiple types of attribution methods: gradient-based, internal-based, and perturbation-based [8]. BERT-Attention-Analysis, an attention-based visualization toolkit that extracts attention maps from BERT, explores the surface-level patterns in attention layers first and probes individual attention heads respectively [9]. Automated interpretability, consisting of both neuron explainer and neuron viewer, focuses on low-level neuron behaviors by evaluating the explanations by contrasting the simulated and real neuron activations [10].

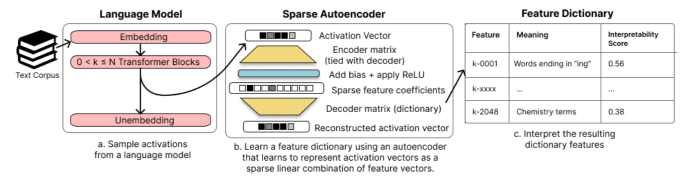


Fig. 1. Sparse-AutoEncoder interpretability pipeline (adapted from [11]).

One of the technical challenges in LLM explainability is polysemanticity, the situation where the same neuron usually gets activated even in the context of very distinct semantics, which undermines how precise and deterministic each neuron can represent certain features [11]. The core issue of superposition, i.e., the number of features exceeds the number of neurons, can be mitigated by effectively learning far more sparse features using sparse autoencoders (SAEs) [11]. Figure 1 demonstrates how a sparse autoencoder can be employed to extract meaningful interpretations from the activation outputs of different layers in a given language model, including attention heads, MLP layers, and residual streams [11]. Accordingly, a sparsity penalty term is combined with the common reconstruction loss in the loss function to reconstruct a sparse linear combination of features [11].

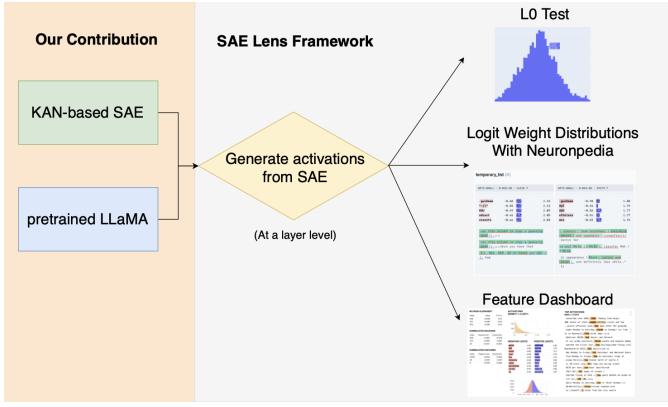


Fig. 2. Our proposed algorithm pipeline for KAN-LLaMA.

## II. RESEARCH MOTIVATION

Sparse autoencoders have been implemented in many existing LLM-interpretability packages: for example, SAELens, an existing LLM-interpretability library, has compiled a total of four sparse autoencoder architectures: Standard SAE, Gated SAE, JumpReLU SAE, and Top-K SAE [12]. However, sparse autoencoders particularly based on Kolmogorov-Arnold Networks (KANs) [13] [14] have not been studied in the context of assisting large language model interpretability. To bridge this gap, we propose to implement sparse autoencoders based on Kolmogorov-Arnold Networks (KANs) [13] to construct a comprehensive interface that effectively captures the interpretability of LLaMA developed by Meta AI [15].

## III. METHODS

### A. KAN-LLaMA: Algorithm Pipeline

We intend to try creating a sparse autoencoder from KANs and using it in SAE Lens [12] to interpret large language model parameters. Our algorithm pipeline is shown in Figure 2. The KAN-based sparse autoencoders are used to generate neuron activations at a particular layer level from the pretrained LLaMA 3.2. The generated activations are passed into SAE Lens [12] to obtain three types of interpretability features, which will be discussed in depth in the section below.

### B. KAN-LLaMA: Interpretability Components

1) *L0 Test: Sparsity Measure of Feature Activation:* L0 Test generates a histogram of the number of activated features per token across a given dataset, which can be used to represent the sparsity of feature activation [12]. Figure 3 illustrates that there are 165 tokens activated between 76 and 77 features.

2) *Logit Weight Distributions With Neuronpedia:* With the overall objective of understanding what each feature from sparse autoencoders actually represents, logit weight distributions can be leveraged to extract statistical measures, such as skewness, kurtosis, and standard deviation, all of which can be leveraged to explain feature behaviors [12]. For example, in Figure 4, after selecting discriminative features with higher skewness values, which indicate strong activation upon

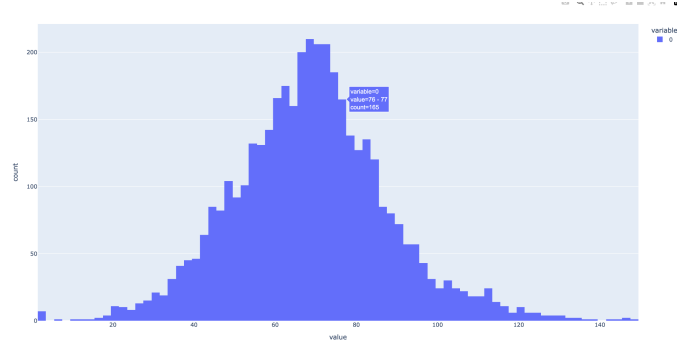


Fig. 3. L0 Test Demo adapted from [12]).

specialized tokens, explicit, comprehensive feature explanation from Neuronpedia is automatically scraped [12].

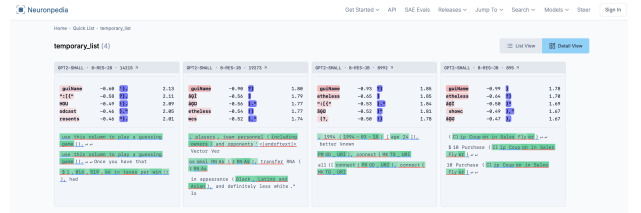


Fig. 4. Logit Visualization Demo adapted from [12]).

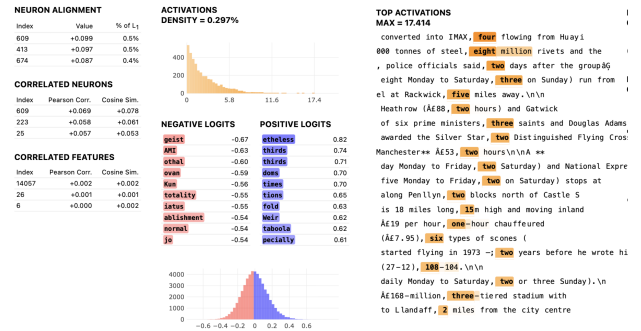


Fig. 5. Feature Dashboard Demo adapted from [12]).

3) *Feature Activation Dashboard:* In addition to poking the most discriminative features through logit weight distribution, an even more comprehensive dashboard per feature, shown in Figure 5, can be generated, highlighting the most correlated neurons and other features, and visualizing top activations within the text paragraph [12]. Additionally, the feature dashboard depicts the activation density and lists a series of tokens contributing to negative and positive logits the most [12].

### C. KAN-LLaMA: LLaMA

LLaMA, a large language model developed by Meta AI, is characterized by leading performance in a large number of language-based tasks: common sense reasoning, closed-book question answering, reading comprehension, mathematical reasoning, and code generation [15]. Unlike other large language


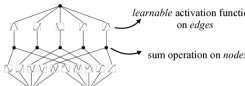
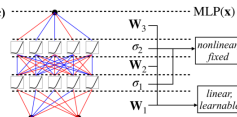
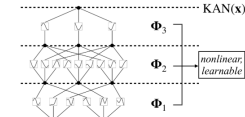
Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(x) \approx \sum_{i=1}^{N(x)} a_i \sigma(w_i \cdot x + b_i)$	$f(x) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^n \phi_{qp}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$MLP(x) = (W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1 \circ W_1)(x)$	$KAN(x) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(x)$
Model (Deep)	(c) 	(d) 

Fig. 6. Multi-Layer Perceptrons (MLPs) vs. Kolmogorov-Arnold Networks (KANs) [13]

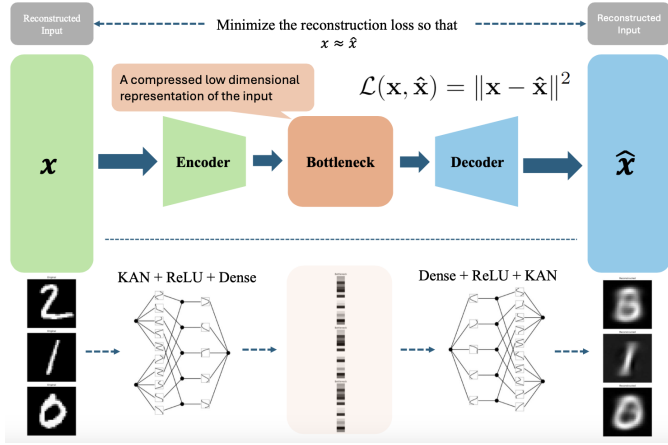


Fig. 7. KAN-based autoencoders architecture [14]

models, LLaMA, a completely open-source, work-compatible model, has been trained only with publicly available data including English CommonCrawl, GitHub, Wikipedia, ArXiv, and Stack Exchange [15].

#### D. KAN-LLAMA: KAN-based Autoencoders

Kolmogorov-Arnold Networks (KANs) are a novel alternative to MLPs, where all activation functions are replaced with flexible trainable spline functions, unique to every neuron [13], as shown in Figure 6. This creates both high parametrization and flexibility of a model, and easy interpretability of the model, since a trained spline function can be loosely mapped to a trigonometric function. There has been some work in creating and using KAN-based autoencoders with the key difference of edge-based activations from traditional autoencoders [14].

### IV. EXPERIMENT EVALUATION

To evaluate KAN-LLaMA, in spite of the focus on model interpretability, both expressiveness and interpretability shall be evaluated respectively.

#### A. Expressiveness Evaluation

1) *Reconstruction and Zero Ablation Test - Sparse Autoencoder Accuracy*: Reconstruction Test is designed to use Sparse Autoencoders to reconstruct and simulate the language model's original feature activations by contrasting its original loss and the reconstruction loss [12]. Similarly, Zero Ablation Test is to wipe out all feature activations by effectively setting them to zero and compute the zero loss accordingly [12]. By comparing the zero loss with its reconstruction loss, the expressiveness of the sparse autoencoders can be captured effectively.

2) *Specific Capability Test - Task Performance Tracing*: Specific Capability Test is designed to evaluate the model's performance of sparse-autoencoder-embedded language models via concrete language tasks, including next-token generation as one of the most common examples, showing that SAE-embedded large language models can still preserve their expressiveness [12].

#### B. Interpretability Evaluation

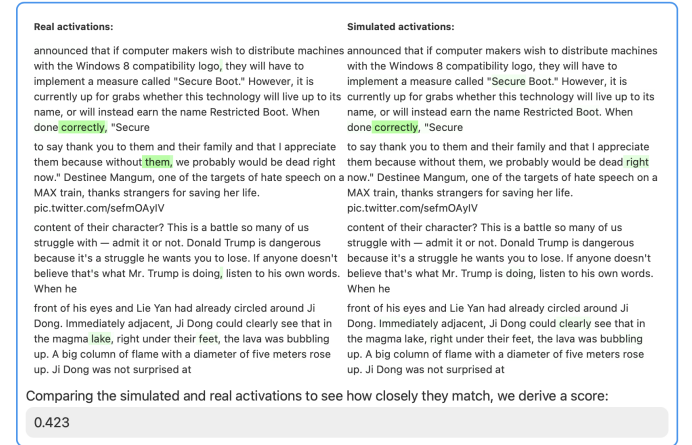


Fig. 8. Interpretability Score Demo adapted from [10]).

In terms of metrics evaluation, interpretability can often be quantified using the autointerpretability score proposed in [10]. The autointerpretability scoring system asks the language model to generate the corresponding readable interpretations based on the real activations, which will be used to simulate the same activations [10]. The degree of correlation between real and simulated activations is defined as the interpretability score of the characteristic [10]. Figure 8 illustrates an example of computed interpretability scores of five distinct characteristics [10].

### REFERENCES

- [1] C. Singh, J. P. Inala, M. Galley, R. Caruana, and J. Gao, "Rethinking interpretability in the era of large language models," *arXiv*, 2024. arXiv: 2402.01761. [Online]. Available: <https://arxiv.org/abs/2402.01761>.

- [2] K. Fauvel, T. Lin, V. Masson, É. Fromont, and A. Termier, “Xcm: An explainable convolutional neural network for multivariate time series classification,” *Mathematics*, 2023. [Online]. Available: <https://inria.hal.science/hal-03469487>.
- [3] V. Dibia, “Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models,” *arXiv*, 2023. arXiv: 2303.02927. [Online]. Available: <https://arxiv.org/abs/2303.02927>.
- [4] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” *arXiv*, 2017. arXiv: 1703.01365. [Online]. Available: <https://arxiv.org/abs/1703.01365>.
- [5] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, “What does bert look at? an analysis of bert’s attention,” *arXiv*, 2019. arXiv: 1906.04341. [Online]. Available: <https://arxiv.org/abs/1906.04341>.
- [6] R. W. et al., “Large language models are interpretable learners,” *arXiv*, 2024. arXiv: 2406.17224. [Online]. Available: <https://arxiv.org/abs/2406.17224>.
- [7] J. Alammar, “Ecco: An open source library for the explainability of transformer language models,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2021.
- [8] G. Sarti, N. Feldhus, L. Sickert, O. van der Wal, M. Nissim, and A. Bisazza, “Inseq: An interpretability toolkit for sequence generation models,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 421–435. DOI: 10.18653/v1/2023.acl-demo.40. [Online]. Available: <https://aclanthology.org/2023.acl-demo.40>.
- [9] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, “What does bert look at? an analysis of bert’s attention,” in *BlackBoxNLP@ACL*, 2019.
- [10] S. Bills, N. Cammarata, D. Mossing, et al., *Language models can explain neurons in language models*, <https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html>, 2023.
- [11] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, *Sparse autoencoders find highly interpretable features in language models*, 2023. arXiv: 2309.08600 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2309.08600>.
- [12] C. T. Joseph Bloom and D. Chanin, *Saelens*, <https://github.com/jbloomAus/SAELens>, 2024.
- [13] Z. Liu, Y. Wang, S. Vaidya, et al., *Kan: Kolmogorov-arnold networks*, 2025. arXiv: 2404.19756 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2404.19756>.
- [14] M. Moradi, S. Panahi, E. Bollt, and Y.-C. Lai, *Kolmogorov-arnold network autoencoders*, 2024. arXiv: 2410.02077 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2410.02077>.
- [15] H. T. et al., “Llama: Open and efficient foundation language models,” *arXiv*, 2023. arXiv: 2302.13971. [Online]. Available: <https://arxiv.org/abs/2302.13971>.

## V. APPENDIX

### A. KAN Implementation

- 1) Official with builtin interpretability features, but slow: GitHub
- 2) Efficient, but less features: GitHub
- 3) More resources: GitHub list

### B. SAE Lens Implementations

- 1) Official SAE implementation: GitHub
- 2) Official SAE Dashboard implementation: GitHub