# Explaining Latents in Turing-LLM-1.0-254M with Pre-Defined Function Types

Daniel Davies
Independent Researcher

Ashwarya Maratha
Independent Researcher

**With**
Goodfire and Apart Research

## Abstract

Interpreting the latent representations within large language models (LLMs) remains a significant challenge in advancing AI transparency and control. This study introduces a novel framework for explaining latents in the Turing-LLM-1.0-254M model based on a predefined set of function types, allowing for a more human-readable "source code" of the model's internal mechanisms. By categorising latents using multiple function types, we move towards mechanistic interpretability that does not rely on potentially unreliable explanations generated by other language models. Evaluation strategies include generating unseen sequences using variants of Meta-Llama-3-8B-Instruct provided by GoodFire AI to test the activation patterns of latents, thereby validating the accuracy and reliability of the explanations. Our methods successfully explained up to 95% of a random subset of latents in layers, with results suggesting meaningful explanations were discovered.

*Keywords: AI Observability, Mechanistic Interpretability, Model Reprogramming*

# 1. Introduction

This research introduces a framework for explaining latents through pre-defined functions, advancing our ability to examine these complex systems with greater accuracy. Our endeavour examines and explains the internals of Turing-LLM-1.0-254M using sparse autoencoder features. The objective of this project is to develop logical methods for obtaining interpretable latent explanations.

As models gain intelligence, it is appropriate to assume that their internals will become more complex and potentially more difficult to interpret. Thus, to explain latents using current methods, such as one introduced by Bills, et al. (2023), we'd most likely need to use models of similar intelligence to generate explanations. However, this also requires the model to already be aligned with human values and wish to help us explain latents. This problem presents a challenge that defeats one of the purposes of mechanistic interpretability: to align non-benevolent superintelligences. Furthermore, explaining latents using language models can introduce issues due to their unreliable outputs. We propose methods to generate explanations that are more reliable and mathematical in nature, hoping that this framework will allow future research to develop a myriad of discovered latent functions.

We developed three main functions: *SpecificToken()* which identifies and tracks how the model processes individual tokens, *ConnectingTokens()* which maps pairs of tokens, and *DetectingDatasetTopic()* which uncovers themes of inputs on which latents activate upon. To achieve this, we utilised Sparse Autoencoders (SAEs) that expand the model's neuron activations into 40,960 features per layer.

# 2. Overview

## 2.1 Function Types

| Function Type | Description |
|---|---|
| *SpecificToken()* | A latent that focuses on specific tokens. |
| *ConnectingTokens()* | A latent that focuses on two or more tokens that the model connects internally. |
| *DetectingDatasetTopic()* | A latent that focuses on an interpretable topic from the model's organised training dataset. |

The training dataset of Turing-LLM-1.0-254M consists of entirely structured synthetic texts, with text file paths revealing the topics of texts. For example, one may find a text relating to Relativity in a directory such as: "subjects/items/Physics/items/Relativity/". This allows for a unique advantage in understanding this model's latents as latent top sequences can be related to interpretable topics. Suppose a latent focuses entirely on Relativity, it can be assumed that most of the top sequences will be derived from a similar directory as the previous example. Thus, one would be able to describe certain latents using these topics.

### 2.2 Evaluation of Novel Explanations

To verify our explanations, we employed an approach using the Meta-Llama-3-8B-Instruct model (Dubey, A. et al., 2024) through the Goodfire SDK (Goodfire, 2024). This model generates test sentences that help validate our findings by producing examples for each pattern we discover. For instance, when our system identifies a frequent connection between tokens like "Albert" and " Ein", the Llama model generates sentences using these tokens which we inject in a prompt to confirm if the corresponding latent in Turing-LLM activates on this unseen input. We evaluate 82 latents for each of the 12 layers due to time constraints.

For every test sequence processed, we track how strongly the model reacts. We compare these activation patterns against average activations to evaluate our methods. This gives us a numerical measure of how confident we can be in our explanations.

## 3. Code

The code for this project is split into three files, each with a distinct function. Links to all the code and weights are in Appendix 6.1. The following table contains information on each file.

| File Name | Description |
|---|---|
| *get_data.py* | Gets explanations for each latent for the following function types: *SpecificToken()*, *ConnectingTokens()*, and *DetectingDatasetTopic()*. |
| *get_eval_inputs.py* | Gets unseen token sequences for $n$ latents for each layer. Each sequence is generated according to explanations from *get_data.py*. |
| *eval.py* | Gets evaluation results for each unseen sequence for $n$ latents for each layer. These results include success status and distance of the latent activation from the average activation of the latent. |

The Goodfire AI Python SDK (Goodfire, 2024) was utilised with Meta-Llama-3-8B-Instruct (Dubey, A. et al., 2024) to generate test sequences. For the *SpecificToken()* data, two sets of test sequences were produced for each latent, with half generated using variants with dataset-topic-related latents amplified.

The code, weights, SAEs, and interpretability research relating to Turing-LLM-1.0-254M will be introduced by Daniel Davies in a paper in the near future. Currently, information on this model, related research, and its latent space explorer tool is available on https://turingexplorer.com (Davies., 2024) (Karpathy, A., 2024) (Abdin, M. et al., 2024).

TopK SAEs (Gao, L. et al., 2024) were used for each layer of the model to obtain a more monosemantic set of features. The SAEs each have a hidden dimension size of 40,960 latents.

Many latents appeared to have many dataset topics in their top sequences. These topics were filtered using sentence embeddings of topics and hierarchical clustering to set the most cohesive subset of topics. These subsets of topics appeared to more closely align with patterns in top sequences than the original sets of topics.

## 4. Discussion and Conclusion

The following is a table of examples of data relating to latent 2 in layer 1 of Turing-LLM:

| Function Type | Layer 1 Latent 2 Data |
|---|---|
| *SpecificToken()* | `['ure', 'ural', ',', 'agricult', 'and', 'to']` |
| *ConnectingTokens()* | `[['ure', 'agricult'], ['ure', 'ural'], ['ural', 'agricult']]` |
| *DetectingDatasetTopic()* | `['Culinary Arts', 'Industrial Revolution', 'Agricultural Development', 'Soil Science', 'Agricultural practices', 'Agricultural Revolution']` |

As one can observe, the data collected appear related to agriculture, suggesting that this latent may detect this topic. Consistency of such explanations provides confidence qualitatively.

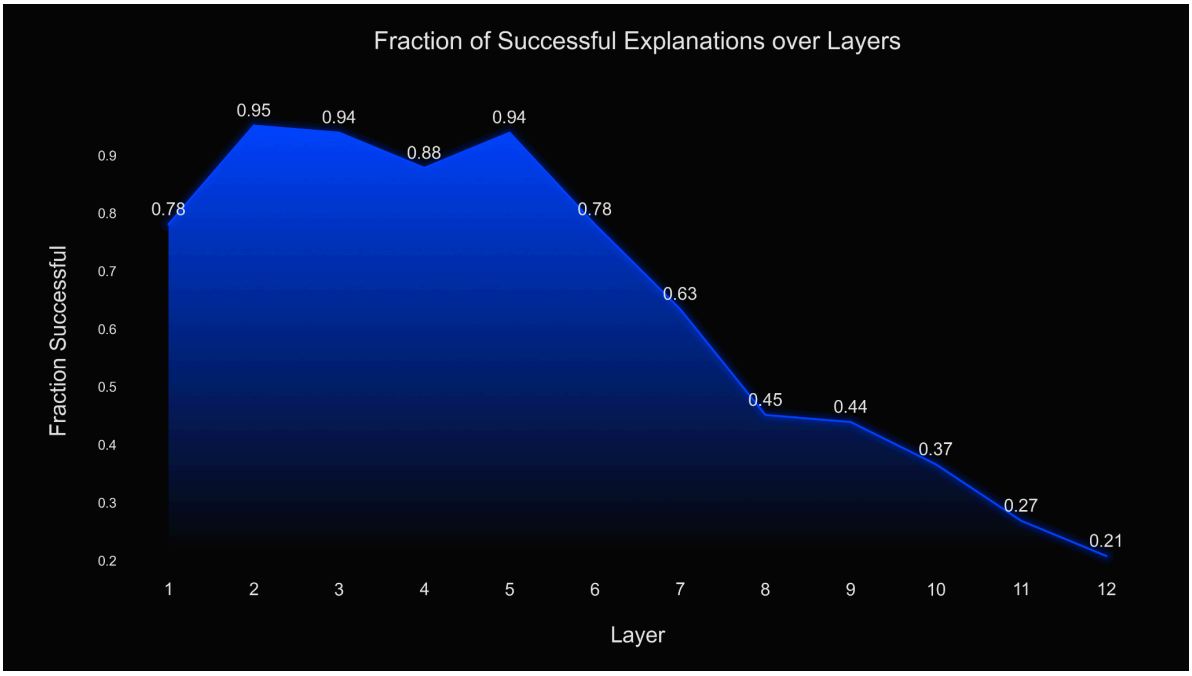


Figure 1 - Fractions of Successful Explanations over Layers

Evaluation results, displayed in Figure 1, suggest evidence that a meaningful fraction of latents have been explained successfully using the function types defined. These results also imply an improvement over various results presented by Bills, et al. (2023), all while not relying on language models for explanations. In an effort to minimise the main content page count, further results, discussions, and suggestions for future work are present in Appendix 6.3. Results relating to usage of the Goodfire AI SDK are also present in the Appendix.

It is our sincere hope that this work may contribute, however modestly, to advancing research that enables us to explore, understand, and learn from digital superintelligence.

## 5. References

Abdin, M. et al., 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. Microsoft.

Bills, S. et al., 2023. Language models can explain neurons in language models. OpenAI.

Davies., 2024. Turing-LLM-1.0-254M. https://turingexplorer.com

Dubey, A. et al., 2024. The Llama 3 Herd of Models. Meta.

Gao, L. et al., 2024. Scaling and evaluating sparse autoencoders. OpenAI.

Goodfire, 2024. Goodfire API. Goodfire AI.

HuggingFace, 2024. all-MiniLM-L6-v2.

Karpathy, A., 2024. Let's reproduce GPT-2 (124M).

# 6. Appendix

## 6.1 Links to Code and Weights

To run *eval.py*, one only needs to download this project's research. However, to run the entire project, all files in the links below need to be downloaded. Several .txt files are placed in the GitHub repository to guide the placement of downloaded files.

To download the code for this research:

https://github.com/DanielJamesDavies/Explaining-Latents-with-Function-Types

To download Turing-LLM-1.0-254M:

https://www.kaggle.com/models/danieljamesdavies/turing-llm-1.0-254m

To download Turing-LLM-1.0-254M Sparse Autoencoders:

https://www.kaggle.com/datasets/danieljamesdavies/turing-llm-sparse-autoencoders

To download Turing-LLM Synthetic Dataset:

https://www.kaggle.com/datasets/danieljamesdavies/turing-llm-synthetic-dataset

To download Turing-LLM Latent Top Sequences:

https://www.kaggle.com/datasets/danieljamesdavies/turing-llm-latent-top-sequences

## 6.2 Further Implementation Details

The model, "sentence-transformers/all-MiniLM-L6-v2" (HuggingFace, 2024), was used to generate the sentence embeddings for dataset topics.

## 6.3 Further Results, Discussions, and Suggestions for Future Work

### 6.3.1 Further Results

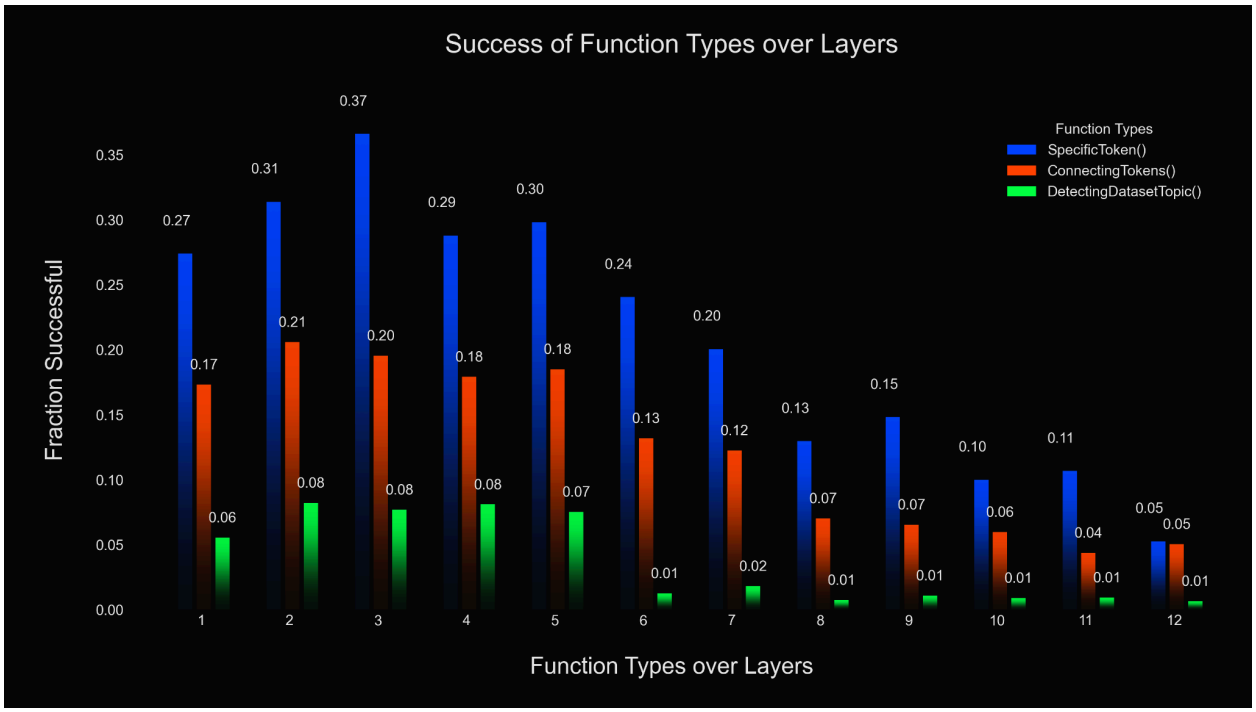


Figure 2 - Success of Function Types over Layers

As visualised in Figure 2, it appears that the *SpecificToken()* function was most successful over layers. Furthermore, all methods managed to achieve a meaningful contribution to explanations. Perhaps many more function types are necessary to fully capture the inner workings of all latents, with each function type contributing a small fraction to a model's intelligence.

Additionally, we found that the evaluation methods were not functioning entirely in this research's favour. Many unseen test sequences were generated without the necessary tokens or themes to be able to evaluate explanations. To achieve more optimal results, one may need to run many more iterations of evaluations per explanation to filter out the noise of undesired test sequences.

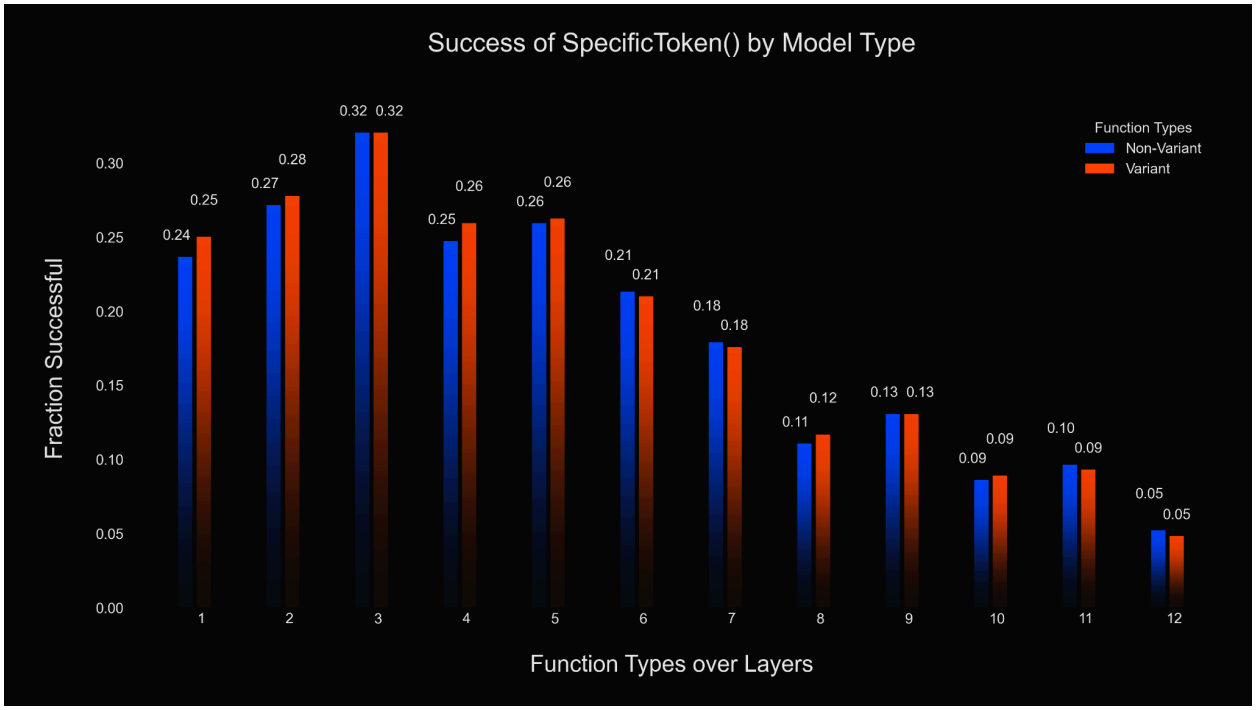


Figure 3 - Success of SpecificToken() by Model Type

Figure 3 visualises results implying that the variant and non-variant configurations of models are relatively equal in success over the function type *SpecificToken()*. However, it appears that in most layers, variant models slightly exceed the non-variant model in success rate. Perhaps this difference would have been increased if this method was applied in a more complex and thoughtful manner, as well as with other function types.

### 6.3.2 Future Work

Due to time constraints, this project was limited in the number of latent evaluations that could have been carried out. Future research could invest more time and compute into fully evaluating every latent in a model using our methods. Furthermore, future work could involve generating multiple unseen test sequences per latent for higher confidence in results.

Two additional function types were devised but not implemented in this research. These are described in the following table.

| Function Type | Description |
|---|---|
| *SpecificWord()* | A latent that focuses on specific words. |
| *DetectingSpecificConcept()* | A latent that focuses on an interpretable specific concept. |

The following is a high-level description of an algorithm for a potential *DetectingSpecificConcept()* function type:

- Split token sequences into subsets of tokens (perhaps by words or phrases)
- Cluster the subsets together using sentence embeddings
- Label these clusters

An example of a group of specific concept cluster labels could be “art” and “interaction” for a latent that focuses on interactive media.

### 6.3.3 Broader Implications of This Research

As described in the introduction section, it may become imperative for the mechanistic interpretability field to rely less on models which we do not fully understand to aid in research. The success of our approach of explaining latents using pre-defined function types shows that reliance on such models may not be necessary. Using the foundations our research presents, future work may involve utilising a large number of function types to “convert” the latents and circuits of models into interpretable “source code”. Such “source code” would allow for full AI transparency, understanding, and control.

### 6.3.4 Acknowledgements

We would like to express our sincere gratitude to the teams at Goodfire AI, Apart Research, and all those who have helped with this research.