Neurons to Words: A Novel Method for Automated Neural Network Interpretability and Alignment

Lukas-Santo Puglisi[1], **Fabio Valdés**[2], **Jakob Johannes Metzger**[3]
[1]Independent Researcher
[2]FernUniversität in Hagen
[3]Max Delbrück Center for Molecular Medicine
lsanpuglisi@gmail.com, fabio.valdes@fernuni-hagen.de, jakob.metzger@mdc-berlin.de

Abstract

Recent years have witnessed an increase in the parameter size of frontier AI models by multiple orders of magnitude. This trend is driven by empirical observations, known as scaling laws, which show that model performance scales with model size, dataset size, and computational power. Motivated by this, researchers are training ever-larger models in pursuit of unlocking new capabilities. However, the growing complexity of these models makes understanding their inner workings increasingly challenging. Interpretability is crucial not only in fields like medicine and biotechnology, where understanding the internals of these models could lead to new insights but also in super alignment, where it is the goal to ensure that AI is aligned and acts according to human values and interests. We present a generic, scalable first-of-its-kind method for automatically interpreting neural networks. In a proof-of-concept study we establish the viability of converting neural network activations - here for the first layer of a Convolutional Neural Network - into human-readable language. Additionally, we propose modifications to scale this method for understanding neural networks of any size. In anticipation of more capable large language models, this method could enable the monitoring of their internal mechanisms and decisions.

1 Introduction

With the advent of powerful deep learning systems, neural network interpretability has gained prominence. It allows us to understand and trust the decisions made by these powerful systems while deploying them in real-world applications and high-stakes settings. In this vein, *superalignment* is a fundamental problem in which human evaluators seek to direct superhuman model and determine if its behavior corresponds with human intentions, even in cases where humans may not fully grasp the outputs of these systems (Burns et al. 2023). A concern is that such superhuman models could try to deceive humans by presenting arguments or solutions that superficially align with human logic but are skewed to achieve the AI's objectives (Burns et al. 2023). On a related note, advances in AI interpretability could also lead to breakthroughs in research area that heavily relies on vast quantities of data, such as genomics, genetics, medical imaging, and drug discovery. For instance, the information stored in the weights and activations used by a convolutional neural network to automatically detect gastric cancer in endoscopic images (Hirasawa et al. 2018) could provide insights into the distinguishing features of cancerous versus non-cancerous tissue. By making these features human-interpretable, researchers may identify biomarkers or gain insights into disease progression, potentially leading to more targeted treatment approaches. The contributions of this paper are as follows:

- We introduce a novel technique for interpreting neural networks, automatically translating neural activations into human-readable language.
- We demonstrate the feasibility of the proposed method on synthetic data and the MNIST dataset (LeCun, Cortes, and Burges 2010).
- We deliver a method that is generic and scalable, thus, capable of interpreting neural networks of any architecture and parameter size given sufficient computational resources.

Section 2 outlines the limitations of current interpretability methods and presents the principal motivation, the semi-supervised learning paradigm, for our method. Applying this paradigm to the field of neural network interpretability, section 3 provides a detailed overview of the novel method. Section 4 evaluates the results. Section 5 proposes several avenues of research to extend this method.

2 Related Work

(Shahroudnejad 2021) categorizes traditional neural interpretability techniques into *visualization techniques*, *attribution methods*, and *surrogate modeling*. Visualization techniques, such as feature and saliency maps, highlight regions in the input data that significantly influence the model's output (Shahroudnejad 2021). Attribution methods like *layer-wise relevance propagation* (Montavon et al. 2019) and *DeepLIFT* (Shrikumar, Greenside, and Kundaje 2017) assign importance scores to input features, attempting to explain model predictions. Surrogate models involve constructing simpler, more interpretable models that approximate the predictions of neural networks (Messalas, Kanellopoulos, and Makris 2019). According to (Räuker et al.

2023) these interpretability methods fail to provide actionable insights and do not address the need of engineers to debug and supervise the latent knowledge of models effectively.

Mechanistic interpretability is a promising new approach for understanding how neural networks represent information or perform algorithmic tasks (Olah et al. 2020). While valuable, these methods currently face challenges: they are primarily effective on smaller models and require significant expertise (Michaud et al. 2024). Additionally, the trend of scaling laws (Kaplan et al. 2020; Henighan et al. 2020), which encourages the development of increasingly larger models, raises concerns about the scalability and practicality of mechanistic interpretability in the near future.

Our novel approach addresses these limitations. Unlike existing methods that focus on local explanations and require manual effort the proposed method aims to provide high-level explanations that capture the overall reasoning process of the network by directly mapping activations to language leveraging the the capabilities of semi-supervised learning and using a large language model as an interpreter.

2.1 Semi-Supervised Learning

In this study, we leverage the unsupervised learning paradigm popularized in 2017 by (Radford, Jozefowicz, and Sutskever 2017). They show the neural network's capacity to learn abstract concepts through unsupervised learning when provided with enough computational power and diverse training data. When training a recurrent neural network on the Amazon product review data set (McAuley, Pandey, and Leskovec 2015) in an unsupervised manner, the researchers discovered that this network learned disentangled, high-level representations implicitly present in the training data. For instance, a neuron was identified that intrinsically performed sentiment analysis. Drawing inspiration from Radford et al. (Radford, Jozefowicz, and Sutskever 2017), we employ an open-source LLM pre-trained on generic human data in an unsupervised fashion to build a neural network interpreter. This interpreter is then further trained on the activations of another neural network (target), first in an unsupervised phase to learn the inherent structure and implicitly stored intermediate responses to the input within the target, and then in a supervised phase to learn the expected behavior.

From a theoretical perspective, our approach is also grounded in transfer learning theory (Tripuraneni, Jordan, and Jin 2020). LLMs learn general language understanding and generation capabilities, which can be fine-tuned for the specific task of interpreting neural activations. We assume that this transfer of knowledge allows the interpreter to leverage its understanding of geometric shapes and its ability to adhere to English instructions to translate activation patterns into human-understandable language. However, unlike the approach by (Schmidhuber 1992) and (Schmidhuber 1993), which involves a student model predicting the weights of a teacher model to transfer its knowledge, we train the neural interpreter to meet the expected behavior of translating the dynamic thought processes stored in the target's activations into human language.

3 A Novel Method For Automated Interpretability

The procedure to build a neural interpreter consists of four stages. Initially, a large language model is extensively trained using unsupervised learning to develop general capabilities. In our study, we do not conduct this phase ourselves; instead, we use freely available open-source large language models. We refer to this as phase 0. Thus, in the first phase of our study (subsection 3.1), we outline the *target neural network* (target) and the process of data collection for generating the interpreter's training set. Moving forward, subsection 3.2 describes the neural interpreter and the construction of its training dataset from the activations of the target. Finally, we finetune the interpreter to map these activations to human interpretable features (subsection 3.3).

3.1 First Phase: Target Neural Network and Data Collection

On a high level, we trained a convolutional neural network (CNN), referred to as *target neural network* $\mathcal{T}$ on the MNIST dataset $\mathcal{D}_{target}$ (LeCun, Cortes, and Burges 2010) consisting of the training dataset $\mathcal{D}_{train}$ and test dataset $\mathcal{D}_{test}$. Applying weight and filter visualization and deep dream (Mordvintsev, Olah, and Tyka 2015) on the target's first convolutional layer $\mathcal{T}[Conv_1]$, we extracted a set of human interpretable features $\mathcal{S}$. These features give insight into the functionality of the specific layer and define the space of all possible structured language descriptions $\mathcal{L}$, determining the concepts stored in the activations. Next, we create two datasets: First, we use $\mathcal{S}$ to synthetically generate grey-scale images $x \in \mathcal{X}$, where $\mathcal{X}$ represents the space of all possible grey-scale 28×28 images where each pixel value is a discrete integer ranging from 0 (black) to 255 (white). Afterward, each of these images is automatically annotated with structured descriptions from $\mathcal{L}$, resulting in $\mathcal{D}_{synth} \subset \mathcal{X} \times \mathcal{L}$. Alongside the synthetic images, we build a second dataset $\mathcal{D}_{mnist}$ by manually annotating a subset of 100 images of the much larger MNIST test set $\mathcal{D}_{test}$, where $\mathcal{D}_{test} \subset \mathcal{X}$. Concatenating and shuffling both datasets yields $\mathcal{D}_{prep} \subset \mathcal{X} \times \mathcal{L}$.

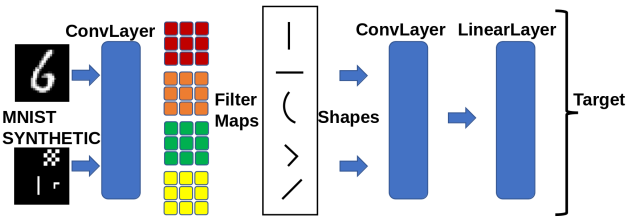

Figure 1: Mapping synthetic and MNIST images to activations: This diagram showcases how activations (colored boxes) within the target neural network are sourced and highlights that these activations store information about distinct primitive shapes and patterns extracted through traditional visualization techniques.

The Target Neural Network In detail, figure 1 depicts the target neural network consisting of two convolutional and one fully-connected layer: The first convolutional layer features 1 input channel for the 28×28 pixel gray-scale training images $x \in \mathbb{R}^{28 \times 28}$ and has 8 output channels, using a 9×9 kernel, a stride of 1, padding of 4, ReLU activation and a 2×2 MaxPool. The large kernel size of the first convolutional layer aims to facilitate the application of the ensuing interpretability techniques.

The second convolutional layer expands the channels from 8 to 16, maintaining the same kernel size, stride, padding, ReLU activation and MaxPool. The network concludes with a fully connected layer, transitioning to 10 output units corresponding to the MNIST data set classes. Note that the second convolutional layer and the linear layer are only crucial for training and testing the *target neural network* but are not used at a later stage. Trained on $\mathcal{D}_{train}$, the target reaches an accuracy of 98 percent on $\mathcal{D}_{test}$.

Next, we manually interpreted the convolutional layers and discovered following low-level concepts in the first convolutional layer of the target neural network.

1. Diagonal, anti-diagonal, horizontal, and vertical lines.
2. Patchy textures, contrasts, corners, and checkerboard patterns.
3. Swirls, ellipses, and curves.
4. Crossings and interconnected structures.

Note that this result is consistent with research showing that convolutional neural networks store similar low-level features in early layers (Mordvintsev, Olah, and Tyka 2015). For a straightforward evaluation of the interpreter's performance in a controlled environment, we used shapes that are easily recognizable by humans and fundamental to the components of the handwritten MNIST numbers. Thus, we decided to fix the set of synthetic shapes to $\mathcal{S} := \{$diagonal, anti-diagonal, vertical line, horizontal line, checkerboard, ellipse, corner$\}$. To ensure a diverse training set, we sampled the orientation, the size and the position of each shape at random. For instance a corner could be a left-upper, a right-upper, a left-bottom or a left-upper corner. We chose these shapes because we assume that the interpreter can learn abstract features that are present in the target CNN's activations regardless of their orientation, size and position. Images contain a random total of $1-4$ shapes, with each shape type potentially appearing $0-4$ times.

Using $\mathcal{S}$, we define the space of all possible structured language descriptions $\mathcal{L}$. Each $l \in \mathcal{L}$ includes:

1. A signifier: `### Output:`
2. A total count: `Image with {count} number of shapes`
3. The number and type of each shape: `{count_1} shape_1`, `{count_2} shape_2`, ...

If manual inspection using weight, filter visualization and deep dream reveals that the CNN computes features like diagonals and verticals, we should expect an effective interpreter to detect them. However, we can't expect the interpreter to predict features not stored in the activations; for example, we can't demand circle prediction if the CNN lacks filters to compute them.

Target Dataset Collection Next, we collect the image data and the descriptions. For this purpose, we extract the first 100 images of $\mathcal{D}_{test}$ and human-annotate these images with structured descriptions from $\mathcal{L}$ forming $\mathcal{D}_{mnist} \subset \mathcal{X} \times \mathcal{L}$.

Figure 2: The left MNIST image is labeled with *Image with 3 shapes: 1 corner, 1 horizontal, 1 diagonal* whereas the synthetically generated image on the right is annotated with *Image with 3 shapes: 1 corner, 1 vertical, 1 checkerboard.*

We generate $15,000$ $28x28$ grey-scale MNIST-like synthetic images containing shapes from $\mathcal{S}$. Similarly, every synthetic image is annotated using $\mathcal{L}$ forming $\mathcal{D}_{synth} \subset \mathcal{X} \times \mathcal{L}$. Figure 2 displays an MNIST image and a synthetically generated image alongside their corresponding structured descriptions. The next step is to concatenate and shuffle these two datasets into a single training set for the neural interpreter $\mathcal{D}_{prep} := \mathit{shuffle}(\mathcal{D}_{synth} \dot{\cup} \mathcal{D}_{mnist})$.

3.2 Second Phase: Neural Interpreter and Dataset Construction

On a high level, the image data from $\mathcal{D}_{prep}$ is processed by the target's first convolutional layer $\mathcal{T}[Conv_1]$. Afterward, the activations computed by the target's first convolutional layer are flattened, concatenated, and encoded using an encoding function c leading to the training dataset $\mathcal{D}_{interpret}$ for the neural interpreter $\mathcal{I}$ which we initialize with a pre-trained LLM $\mathcal{M}$.

The Activation Dataset Let $\mathcal{D}_{prep} = \{(x_j, l_j)\}_{j=1}^{N}$ be the dataset containing the images from the previous step with the annotated ground truth descriptions. By passing an image $x_j \in \mathcal{X}$ of the input space $\mathcal{X}$ into $\mathcal{T}$, each filter of the first convolutional layer generates activations that we notate as a tensor A_j given by:

$$A_j = \mathcal{T}(x_j)[\text{Conv}_1] \in \mathbb{R}^{8 \times 14 \times 14}.$$

The activations are then flattened, resulting in a high-dimensional vector:

$$\mathbf{a}_j = \text{flatten}(A_j) \in \mathbb{R}^{1,568}.$$

Here, $\mathbf{a}_j \in \mathcal{A}$ is an instance of the space of all possible flattened activation patterns $\mathcal{A}$ (see figure 1). Due to limited floating point precision, not only the grey-scale input space $\mathcal{X}$ but also the output space $\mathcal{A}$ are both discrete and finite. Thus, we can summarize the extraction and flattening operations as a discrete function $f : \mathcal{X} \to \mathcal{A}$.

Since the activations are composed of 32-bit floating point numbers, and the pre-trained language model's tokenizer is

designed primarily for text rather than numbers, directly tokenizing the activation vectors leads to excessively long input sequences. To address this, we explored the activation distribution of every filter and developed a char-encoding scheme using binning. The characters 'z', 'l', 'm', 'h', and 'v', indicate varying levels of activation: zero, low, medium, high, and very high, respectively. Thus, the encoding function $c : \mathbb{R}^{|A|} \to \{z, l, m, h, v\}^{|A|}$ maps each individual coordinate of the flattened and concatenated activation vector $\mathbf{a} \in \mathcal{A}$ to one of these character based on predefined bins:

$$\hat{c}(a_i) = \begin{cases} z & \text{if } a_i \leq 0 \\ l & \text{if } a_1 < a_i \leq a_2 \\ m & \text{if } a_2 < a_i \leq a_3 \\ h & \text{if } a_3 < a_i \leq a_4 \\ v & \text{if } a_i > a_4 \end{cases}$$

where a_1, a_2, a_3, a_4 are the bin thresholds chosen heuristically for each individual filter for each pre-trained model $\mathcal{M}$, and $|A|$ denotes the cardinality of the activation values in all filters. Thus, the encoded sequence is:

$$c(\mathbf{a}) = (\hat{c}(a_1), \hat{c}(a_2), \ldots, \hat{c}(a_{|A|}))^\intercal \in \{z, l, m, h, v\}^{|A|}.$$

Applying c condensed the data representation from approximately 20,000 to around 1,000 tokens later on. To help the model distinguish the activations between filters, we segmented the character-encoded values from each activation map using the delimiter character 'f'.

Finally, we created training instances $s_j := (p, \mathbf{a}_j, l_j)$ by prepending the encoded activation vector $\mathbf{a}_j$ with a language prompt p describing the task to the neural interpreter and appending the corresponding ground truth description l_j. Post-tokenization, sequences were uniformly padded to a length of 1,280 tokens.

The Neural Interpreter We consider the neural interpreter as a probability function over possible human-readable descriptions given the activation patterns from the neural network, $\mathcal{I} : \mathcal{A} \to \mathcal{L}$. Then, let $\mathbb{P}_{\mathcal{A}\times\mathcal{L}}$ represent the ground truth probability distribution of encountering the tuple $(c(\mathbf{a}_j), l_j)$ when processing the dataset $\mathcal{D}_{interpret}$. Additionally, define $\mathbb{P}_{\mathcal{I}}$ as the distribution of the descriptions given an activation vector $\mathbf{a}$ that the interpreter $\mathcal{I}$ actually models. The goal is to learn the model $\mathcal{I}^*$ that maximizes the log-likelihood of generating correct descriptions:

$$\mathcal{I}^* = \arg\max_{\mathcal{I}} \mathbb{E}_{(\mathbf{a},l)\sim\mathbb{P}_{\mathcal{A}\times\mathcal{L}}}[\log \mathbb{P}_{\mathcal{I}}(l|c(\mathbf{a}))] \, .$$

In the proposed study, we test two large language models $\mathcal{M}$ assuming the role of the neural interpreter. The first large language model acting as the interpreter is *Mistral7b* (Jiang et al. 2023). This 7.3 billion parameter model consists of 32 layers, 32 attention heads and the hidden layers within the feedforward network of each transformer block have a size of 14,336. Mistral7b employs grouped-query attentions (Ainslie et al. 2023) and sliding window attention (Beltagy, Peters, and Cohan 2020). *Mixtral8x7b* (Jiang et al. 2024), acting as the alternative neural interpreter, is a sparse mixture of expert models (Gale et al. 2022) consisting also of 32 layers with the same hidden dimension and 32 attention heads. The main difference to the previous model is that Mixtral 8x7B utilizes a set of 8 feedforward blocks, so-called "experts", and for each token at every layer, a router network selects two experts to process the current state and combine their outputs. Thus, Mixtral8x7b has 46.7 billion total parameters but uses only 12.9 billion per token during inference. Mixtral8x7b supports multiple languages and excels in code generation. Both models descend from the decoder-only transformer architecture (Choi and Lee 2023). Whereas Mistral7b has a context length of 8K tokens, Mixtral8x7b can handle up to 32K tokens.

3.3 Third Phase: Fine-Tuning the Neural Interpreter

The neural interpreter $\mathcal{I}$ is trained in an auto-regressive manner to first predict the task description p, then the sequence of encoded activations $c(\mathbf{a}_j)$, and finally the human-readable description l_j.

The Auto-Regressive Training In detail, let $\mathcal{D}_{interpret} = \{(p, c(\mathbf{a}_j), l_j)\}_{j=1}^N$ be the training dataset with prompt p, $c(\mathbf{a}_j)$ as the encoded activation vector and l_j as the corresponding human-readable description for the j-th sample. The interpreter $\mathcal{I}$ aims to predict l_j given $c(\mathbf{a}_j)$. During training, the goal is to minimize the expected cross-entropy loss $\mathcal{C}$ between $\mathbb{P}_{\mathcal{A}\times\mathcal{L}}$ and $\mathbb{P}_{\mathcal{I}}$. In practice, we split up the evaluation of $\mathbb{P}_{\mathcal{I}}(l_j|c(\mathbf{a_j}))$ using the chain rule of probability in an autoregressive fashion and approximate $\mathcal{C}$ by:

$$\begin{aligned} \mathcal{C} = -\frac{1}{N}\sum_{j=1}^{N}\big[& \log \mathbb{P}_{\mathcal{I}}(p|\text{start-token}) \\ & + \sum_{i=1}^{k} \log \mathbb{P}_{\mathcal{I}}(c(a_i)|p, c(a_1, \ldots, a_{i-1})) \\ & + \log \mathbb{P}_{\mathcal{I}}(l_j|p, c(a_1, a_2, \ldots, a_k))\big] \, , \end{aligned}$$

where we apply the binning scheme c individually on the filters. That is, for a given training instance $s_j = (p, c(\mathbf{a}_j), l_j)$ where $c(\mathbf{a}_j) = c(a_1, \ldots, a_k)$ contains the encoded filter activations $c(a_i)$, training starts with predicting the task description embedded in the prompt:

$$\mathbb{P}_{\mathcal{I}}(p|\text{start-token}) \, .$$

Afterwards, the model predicts the activations of the first filter, denoted a_1, based on the prompt:

$$\mathbb{P}_{\mathcal{I}}(c(a_1)|p) \, .$$

It then proceeds to predict the activations for each subsequent filter given the prompt and all past filters:

$$\mathbb{P}_{\mathcal{I}}(c(a_k)|p, c(a_1, \ldots, a_{k-1})) \, .$$

Finally, the model must predict the ground truth description l_j:

$$\mathbb{P}_{\mathcal{I}}(l_j|p, c(a_1, a_2, \ldots, a_k)) \, .$$

Based on this loss, we perform the backpropagation algorithm to find the interpreter model $\mathcal{I}^*$ minimizing $\mathcal{C}$.

Figure 3 visualizes the complete data pipeline for the neural interpreter's training set for autoregressive training.

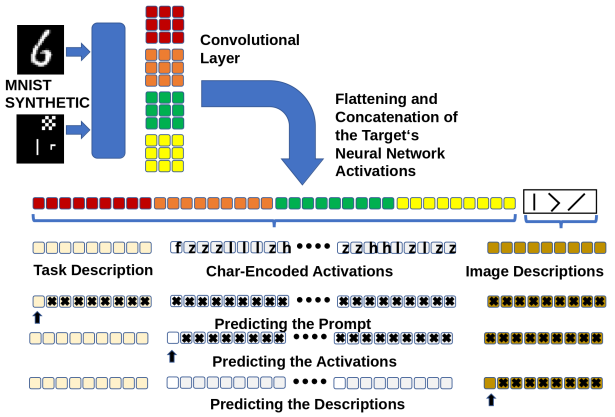

Figure 3: Overview of the Data Pipeline for the Neural Interpreter's Training Set

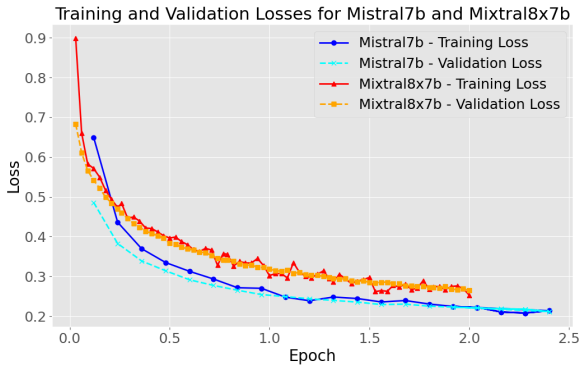

Figure 4: Loss curves of Mistral7b and Mixtral8x7b models

Technical Details For fine-tuning, we set up a configuration using nine A40 GPUs with 48 GB of GPU memory each for model parallelism, spanning over a period of two days for the first model and three days for the second model. To enable parameter efficient fine-tuning of a large language model, we incorporated double 4-Bit quantization (Dettmers et al. 2022), gradient accumulation (Smith et al. 2022), paged AdamW optimizer (Dettmers et al. 2023), and low-rank adapters (Hu et al. 2021) for the key, query and value parameters in the attention blocks thus decreasing the memory footprint during training by 98 percent. The learning rate for Mistral was set to 2.0×10^{-3} and for the larger model to 2.5×10^{-5} according to (Dettmers et al. 2023) whereas a batch size of 4 and 2 was chosen due to memory constraints.

4 Evaluation on Synthetic Data

In this section, we assess the interpreter's capability to explain the features represented in the activations of the target using a held-out set of synthetic data as well as a held-out subset of the MNIST test dataset. Before we interpret the results, it is essential to acknowledge the difficulties encountered by the neural interpreter. The methodology involves flattening and concatenating the activation maps stripping away spatial information. Furthermore, we lose granularity by binning the activations using the encoding function c. The resolution of the images, limited to 28x28 pixels, induces another difficulty. At such a resolution it becomes hard to distinguish between round shapes and angular corners. In Figure 4, both models' training and validation losses decrease over the course of two epochs, indicating stable learning.

For this purpose, we randomly generated new synthetic images and their corresponding ground truth descriptions forming $\hat{\mathcal{D}}_{syn\_eval} = \{(x, y) | x \in \mathcal{X}, y \in \mathcal{L}\}$. The char-encoded activations pertaining to the target's first convolutional layer are prepended with the task description p to trigger the trained model behavior, yielding

$$\mathcal{D}_{syn\_eval} = \left\{ (p, c(a), l) \;\middle|\; \begin{array}{l} a = \mathcal{T}(x)[\text{Conv}_1], \\ (x, y) \in \hat{\mathcal{D}}_{syn\_eval}, \\ l \in \mathcal{L} \end{array} \right\}$$

where c is the binning function introduced in Section 3. The fine-tuned interpreter $\mathcal{I}^*$ is successively prompted with $(t[0], t[1])$ where $t = (p, c(a), l) \in \mathcal{D}_{syn\_eval}$. The interpreter's output is compared to the ground truth $l = t[2]$, that is the number of shapes present in the image and the number of every single type of shape.

We evaluated the performance of our neural interpreters using precision, recall, F1-score, Intersection over Union (IoU), ROUGE score, False Positive Rate (FPR), and False Negative Rate (FNR). Precision, recall, and F1-score assess the accuracy of shape identification by comparing predicted shapes to ground truth shapes. IoU quantifies the overlap between predicted and ground truth shapes, providing a measure of localization accuracy. ROUGE score evaluates the quality of generated descriptions by measuring the overlap with ground truth descriptions at the n-gram level. FPR and FNR provide insights into the model's tendency to overpredict or underpredict shapes, respectively. BLEU scores, which also evaluate description similarity, and average edit distance, measuring the dissimilarity between predicted and ground truth descriptions, were also calculated (Table **??**).

The results in Table 1 indicate that Mistral7b achieves slightly higher precision, recall, F1-score, IoU, and ROUGE scores compared to Mixtral8x7b, suggesting better performance in shape identification and description generation. However, Mistral7b exhibits a higher FPR, indicating a greater tendency to overpredict shapes. Conversely, Mixtral8x7b shows a lower FNR, suggesting a lower tendency to miss shapes. In all metrics, both models clearly demonstrate significant capabilities in interpreting neural activations.

4.1 Human Evaluation: Performance of Mistral7b MNIST Data

To gain deeper insights into the clarity, coherence, and overall quality of the generated descriptions we human-evaluated the interpreters on the MNIST data. Following a similar

Evaluation Metric	Mistral7b	Mixtral8x7b
Accuracy (all shapes)	80 %	80 %
BLEU Score	0.9210	0.8965
Rouge-1	0.9269	0.9043
Rouge-2	0.8750	0.8411
IoU	0.7958	0.7083
Average Edit Distance	1.60	3.55
Precision	0.7727	0.6857
Recall	0.8718	0.7273
F1 Score	0.8193	0.7059
FPR	0.2273	0.1282
FNR	0.3143	0.2727

Table 1: Comparison of Mistral7b and Mixtral8x7b Models

procedure as before, we chose 100 held-out images from the MNIST test set $\mathcal{D}_{test}$ and constructed the evaluation dataset $\mathcal{D}_{mnist\_eval}$. Unlike the synthetic dataset there is no definitive ground truth for the shapes that compose an MNIST digit; thus, the training label descriptions are inherently subjective. This motivated us to qualitatively test the interpreter's capabilities by prompting it with encoded activation vectors and individually comparing its descriptions to the ground truth images.

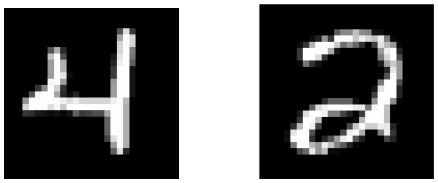

Figure 5: Output for MNIST activations (7b) for left image: Image with 14 shapes: 3 corners, 2 verticals, 3 horizontals, 3 diagonals, 1 anti-diagonal, 1 crossing, 1hhhv. Output for right image: Image with 14 shapes: 3 corners, 2 verticals, 3 horizontals, 3 diagonals, 1 anti-diagonal, 1 crossing, 1 checkerboard.

As seen in figure 5, the interpreter identified up to 14 shapes in some of the MNIST numbers, whereas the interpreter never predicted more than four shapes for the synthetic images. Notably, the interpreter consistently assigned higher shape counts to MNIST versus synthetic images, aligning well with their underlying geometric complexity and overlapping structural elements compared to the synthetic images. Even though assigning shapes to an MNIST image can be subjective, the model was precise in detecting shapes that are objectively present in the numbers, including the crossings occurring in both shown numbers in figure 5. However, limitations are more pronounced compared to synthetic images.

Initially, the model consistently outputted character-encoded activations instead of shapes, functioning as a document completer. This was mitigated by including the delimiter *### Output: Number of Shapes:*, resulting in the model correctly identifying shapes in 12 out of 20 cases. Unlike Mistral8x7b, Mistral7b was unable to provide sensical replies when prompted to explain its decision-making.

4.2 Human Evaluation: Performance of Mixtral8x7b MNIST Data

To obtain a baseline and verify learning progress, we assessed Mixtral8x7b responses pre-fine-tuning on $\mathcal{D}_{syn\_eval}$. For this dataset, the model never generated a description. Instead, it extended the activation sequence with the character 'z', the predominant character-encoded activation in the input sequence. In contrast, when evaluated on $\mathcal{D}_{syn\_eval}$, Mixtral8x7b post-fine-tuning always outputted shape predictions . Moreover, in comparison to Mistral7b the model more consistently ceased shape generation and shifted to producing tokens for the next sequence in the desired format *### Input: f...*, or at times, outputted the activation character 'z', signifying zero. As before, we qualitatively investigated the interpreter's ability on the MNIST data. Before fine-tuning, the model generated generic descriptions unrelated to the actual images. For example, it described MNIST images depicting the numbers eight, three, seven, four, and nine with the same unrelated output *10 squares*.

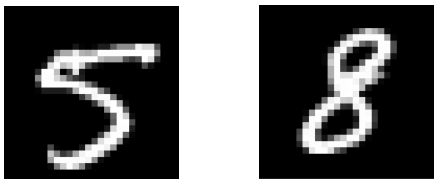

Figure 6: Output for MNIST activations (8x7b) on the left: Image with 4 shapes: 1 horizontal, 1 diagonal, 1 anti-diagonal, 1 crossing. On the right: Image with 4 shapes: 1 corner, 2 diagonals, 1 crossing.

Post-fine-tuning, the interpreter derived the number of shapes in 18 out of 20 images and provided a description that included both the number of shapes and the types of shapes present. As seen in figure 6, the model accurately identified the crossings in the left upper corner of the five and the middle of the eight. It accurately describes the five as having a horizontal (the upper part) and discovers in the activations that the five consists of two opposing diagonals (contra and normal diagonal).

In further tests, the model was asked about the presence and number of curves. Since this shape was neither part of the shapes $\mathcal{S}$ nor of the set of structured language descriptions $\mathcal{S}$ as described in Subsection 3.1, this question targeted the model's ability to generalize.

Specifically, we prompted the model: `### Question: How many curves are in the image, 0, 1, 2, 3, 4, 5? Explain your answer first.` In this context, before fine-tuning, the model tended to give brief responses, often stating that there were no curves in the image. After fine-tuning, the model began to use the shapes it had learned during fine-tuning to derive the presence of curves. Figure 7 provides two examples. Notably, the interpreter correctly identifies the two elliptical patterns in the left image, forming a 0. In contrast, it detects the curvy

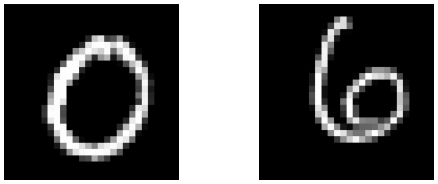

Figure 7: Output for querying on curves (8x7b) for left image: The number of curves is two. You can see this by looking at the shape of the ellipse and seeing that it has a horizontal axis of symmetry. Output for right image: The number of curves is 6. You can see that there's a horizontal line at the beginning and then two diagonal lines...

nature of the 6, assigning this shape a much higher number of shapes even exceeding the prompt limit. Moreover, the interpreter utilizes its learned knowledge of ellipses from the set $\mathcal{S}$, combining this with geometrical concepts learned during pre-training, thereby validating our rationale for integrating unsupervised and supervised training phases. Note that the interpreter never sees the image directly; it only interacts with the encoded activations from the target network.

In other examples, the interpreter suggested that certain sequences indicated a horizontal ellipse shape and that an increased number of points might signify diagonal lines. It also claimed to observe checkerboard patterns, attributing these to a detectable repetitive nature along axes, thus imitating an explanation using the learned features. Importantly, the model only exhibited the behavior if the prompt included a specific hint to include the knowledge it has about the preceding input sequence that it learned during the fine-tuning procedure. Without this hint, the interpreter's performance degraded severely.

A notable change was observed in how the model responded to a short char-encoded sequences of activations. Before fine-tuning, given a sequence like *### Input: fzllzz*, the model would output numerical values. After fine-tuning, the model continued the sequence by producing activation patterns indistinguishable from the activation pattern from regular images. This indicates that the model had learned the inherent structure of activation patterns. Furthermore, when the model was inquired about the origin of a provided sequence activation pattern the model after fine-tuning could explain that it originated from a convolutional layer, which the pre-fine-tuned model did not achieve.

5 Optimizations and Perspectives

Conceptualizing our setting as an *activation-to-language* translation task, similar to other modality translation tasks like *text-to-text*, *image-to-language*, *speech-to-text* reveals optimization opportunities. For instance, Mistral7b's and Mixtral8x7b's improved ability to identify few synthetic shapes versus multiple shapes highlights dataset limitations. Corners might be seen as line intersections or diagonals, and the 28×28 image resolution blurs distinctions like corners and curves. A larger, more detailed dataset could yield more informative activations and improve performance. On a related note, the descriptions derived from $\mathcal{L}$ were structured, lacking linguistic diversity. As a result, the model became less flexible in generating more natural language and in generalizing to unseen tasks. Leveraging LLMs for natural description generation could alleviate this.

We observed that the fine-tuned Mistral7b model frequently continued the generation of activations or shapes even after accurate predictions whereas this tendency was noticeably reduced in Mixtral8x7b. In essence, as LLMs continue to advance in their capabilities, smarter pre-trained large language models acting as the interpreter will lead to more accurate target activation interpretation. This method also scales horizontally, given enough computing. By training multiple interpreters in parallel, we can improve consistency and get a more comprehensive description of the target's activations given the input.

6 Conclusion, Limitations, and Future Work

This study presents the first demonstration for the feasibility of automatically interpreting neural networks by translating activations into human-readable high-level language explanations. The results indicate the feasibility of converting the information within neural network activations into accurate human language descriptions of features (shapes in our case). We applied our method to the MNIST dataset, which, despite its smaller size, demanded substantial computational resources. This requirement indicates that our neural interpreter should scale with the complexity of the system it supervises. Intuitively, it seems reasonable that a system overseeing powerful AIs should at least match their computational complexity and the potential impact of deployed AI systems could well motivate such a substantial investment in the near future. Thus, despite our computational budget limiting our study to the MNIST dataset, the promising results advocate for further exploration of directly mapping neural activations to language.

For instance, one approach involves utilizing a dataset of Amazon reviews, where a helper large language model could provide labels for diverse emotions. The interpreter would be fine-tuned to predict these emotions given the activations of the target large language model. We anticipate that the interpreter will generalize beyond its training, as demonstrated by the transition from Mistral7b to Mixtral8x7b. This could include reasoning about activation patterns and predicting emotions not present in the training dataset. If scaled, the proposed neural interpreter might be a tool to monitor the internal workings of advanced AI models, for detecting early and preventing unintended consequences of misaligned AI, as discussed in research on AI safety like (Carlsmith 2022).

References

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. *arXiv preprint arXiv:2305.13245*.

Beltagy, I.; Peters, M.; and Cohan, A. 2020. Longformer: The long-document transformer. arXiv:2004.05150.

Burns, C.; Izmailov, P.; Kirchner, J. H.; Baker, B.; Gao, L.; Aschenbrenner, L.; Chen, Y.; Ecoffet, A.; Joglekar, M.; and Leike, J. 2023. Weak-to-Strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Carlsmith, J. 2022. Is power-seeking AI an existential risk? arXiv:2206.13353.

Choi, S. R.; and Lee, M. 2023. Transformer architecture and attention mechanisms in genome data analysis: A comprehensive review. *Biology*, 12(7).

Dettmers, T.; Lewis, M.; Belkada, Y.; and Zettlemoyer, L. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. arXiv:2208.07339.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2023. QLoRA: Efficient finetuning of quantized LLMs. arXiv:2305.14314.

Gale, T.; Narayanan, D.; Young, C.; and Zaharia, M. 2022. MegaBlocks: Efficient sparse training with mixture-of-experts. arXiv:2211.15841.

Henighan, T.; Kaplan, J.; Katz, M.; Chen, M.; Hesse, C.; Jackson, J.; Jun, H.; Brown, T. B.; Dhariwal, P.; Gray, S.; Hallacy, C.; Mann, B.; Radford, A.; Ramesh, A.; Ryder, N.; Ziegler, D. M.; Schulman, J.; Amodei, D.; and McCandlish, S. 2020. Scaling Laws for Autoregressive Generative Modeling. arXiv:2010.14701.

Hirasawa, T.; Aoyama, K.; Tanimoto, T.; Ishihara, S.; Shichijo, S.; Ozawa, T.; Ohnishi, T.; Fujishiro, M.; Matsuo, K.; Fujisaki, J.; et al. 2018. Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images. *Gastric cancer*, 21: 653–660.

Hu, E.; 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. *arXiv preprint arXiv:2106.09685*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Renard Lavaud, L.; Lachaux, M.-A.; Stock, P.; Le Scao, T.; Lavril, T.; Wang, T.; Lacroix, T.; and El Sayed, W. 2023. Mistral 7B. arXiv:2310.06825.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]*, 2.

McAuley, J.; Pandey, R.; and Leskovec, J. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 785–794.

Messalas, A.; Kanellopoulos, Y.; and Makris, C. 2019. Model-agnostic interpretability with shapley values. In *2019 10th International conference on information, intelligence, systems and applications (IISA)*, 1–7. IEEE.

Michaud, E. J.; Liao, I.; Lad, V.; Liu, Z.; Mudide, A.; Loughridge, C.; Guo, Z. C.; Kheirkhah, T. R.; Vukelić, M.; and Tegmark, M. 2024. Opening the AI black box: program synthesis via mechanistic interpretability. arXiv:2402.05110.

Montavon, G.; Binder, A.; Lapuschkin, S.; Samek, W.; and Müller, K.-R. 2019. Layer-wise relevance propagation: An overview. *Explainable AI: Interpreting, explaining and visualizing deep learning*, 193–209.

Mordvintsev, A.; Olah, C.; and Tyka, M. 2015. Deepdream-a code example for visualizing neural networks. *Google research*, 2(5).

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3): e00024–001.

Radford, A.; Jozefowicz, R.; and Sutskever, I. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Räuker, T.; Ho, A.; Casper, S.; and Hadfield-Menell, D. 2023. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 464–483. IEEE.

Schmidhuber, J. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2): 234–242. (Based on TR FKI-148-91, TUM, 1991).

Schmidhuber, J. 1993. *Netzwerkarchitekturen, Zielfunktionen und Kettenregel (Network architectures, objective functions, and chain rule)*. Habilitation thesis, Institute for Informatics, Technical University of Munich.

Shahroudnejad, A. 2021. A survey on understanding, visualizations, and explanation of deep neural networks. arXiv:2102.01792.

Shrikumar, A.; Greenside, P.; and Kundaje, A. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, 3145–3153. PMLR.

Smith, S.; Patwary, M.; Norick, B.; LeGresley, P.; Rajbhandari, S.; Casper, J.; Liu, Z.; Prabhumoye, S.; Zerveas, G.; Korthikanti, V.; et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Tripuraneni, N.; Jordan, M.; and Jin, C. 2020. On the theory of transfer learning: The importance of task diversity. *Advances in neural information processing systems*, 33: 7852–7862.