IMPERIAL

Imperial College London

Department of Computing

Explainability in Large Language Models: Pathways to Refinement and Alignment

Author:
Haoyan Luo

Supervisor:
Lucia Specia

Submitted in partial fulfillment of the requirements for the MRes degree in Artificial Intelligence and Machine Learning of Imperial College London

September 2024

Abstract

This thesis investigates the mechanisms underlying large language models (LLMs) through the lens of explainability, with two primary goals: enhancing user trust by making model reasoning more transparent and providing developers with insights to identify undesired properties and improve performance. We categorize current explainability methods into Local and Global Analysis, emphasizing their roles in interpreting model behaviors and enhancing task performance. Additionally, we explore the hidden representations in LLMs, analyzing how token- and layer-level representations evolve and contribute to predictions. Our study further extends to universal concepts encoded in LLMs, examining knowledge neuron theory and conducting a case study on bias neuron detection, revealing semantically meaningful hidden representations and their potential for influencing both training and generation processes. Building on these insights, we propose novel techniques for model refinement and bias mitigation. The Mixture-of-Depths (MoD) framework is introduced as a method for tuning LLMs by leveraging representations across multiple layers, resulting in improved performance with fewer trainable parameters. Furthermore, we present Localized Subspace Projection and Editing (LoPE), a training-free debiasing method that effectively reduces bias without compromising language modeling performance. This thesis seeks to bridge the gap between explainability and practical applications in LLM development. By providing a structured understanding of explainability methods and introducing new techniques for model tuning and alignment, this thesis aims to contribute to the creation of more transparent, robust, and ethically aligned language models.

Acknowledgments

We are grateful to the team behind the TransformerLens library (Nanda and Bloom, 2022) for their invaluable contributions, which greatly supported our research. We also extend our thanks to the researchers at OpenAI and Meta for open-sourcing models such as GPT (Radford et al., 2019a) and LLaMA (Touvron et al., 2023a,c; Dubey et al., 2024), making these tools accessible for academic research. Special thanks go to the authors of (Hu et al., 2023) and (Meade et al., 2022) for us replicating their experimental setups in our work. The related work section of this thesis draws heavily from the survey paper (Luo and Specia, 2024) we authored earlier this year. Additionally, the two sections in Chapter 5 are based on papers currently in progress as part of my MRes study. We are grateful to the anonymous reviewers at EMNLP 2024 for their valuable feedback on one of these papers. Finally, I would like to extend my heartfelt thanks to Lucia for her continued support throughout the course of this research.

Contents

Chapter 1

Introduction

The recent advancements in transformer-based large language models (LLMs) have significantly transformed the field of natural language processing (NLP). Models like GPT-4 (OpenAI, 2023a) have exhibited initial signs of general intelligence (Luo and Specia, 2024; Zhao et al., 2023b), while smaller models have demonstrated strong reasoning abilities, solving challenging commonsense and mathematical problems (Lu et al., 2022; Touvron et al., 2023c; Dubey et al., 2024). Despite their power, these models remain opaque, which can lead to unintended consequences such as the generation of harmful or misleading content (Gehman et al., 2020) and model hallucinations (Weidinger et al., 2021). This opacity underscores the critical need for improved explainability—not only for understanding these models but also for ensuring their responsible and ethical application.

Explainability in LLMs serves two essential purposes. For end users, it fosters trust by making the model's reasoning more transparent in a non-technical way, aiding in the understanding of both its capabilities and limitations (Zhao et al., 2023b). For developers and researchers, explainability offers insights into unintended biases and potential areas for improvement, helping to enhance model performance on downstream tasks (Bastings et al., 2022; Meng et al., 2023a; Li et al., 2024). However, the increasing scale of LLMs introduces significant challenges to explainability. Larger models with more parameters and training data are harder to interpret, and traditional explanation methods, such as SHAP values (Lundberg and Lee, 2017), become less practical for models of this scale (Zhao et al., 2023b). Moreover, a deeper understanding of LLM-specific phenomena, such as in-context learning (Halawi et al., 2023; Hendel et al., 2023; Todd et al., 2023; Wang et al., 2023b), alongside addressing issues like model hallucinations (Ji et al., 2023; Chuang et al., 2024) and inherent biases (dev, 2023; An and Rudinger, 2023; Schick et al., 2021), is crucial for the ongoing refinement and ethical deployment of these models.

In this thesis, we first present a structured categorization of current explainability methods in Chapter 3. We divide these into two broad domains: Local Analysis and Global Analysis. Local Analysis focuses on feature attribution (Sundararajan et al., 2017; Kobayashi et al., 2020; Modarressi et al., 2023) and transformer block analysis (Yuksekgonul et al., 2024; Geva et al., 2021, 2022a), examining detailed model operations. In contrast, Global Analysis includes probing-based methods (Hewitt and Manning, 2019b; Li et al., 2024) and mechanistic interpretability (Wang et al., 2023a; Nanda et al., 2023), providing a comprehensive understanding of model behaviors. We also discuss approaches leveraging explainability to enhance model performance (Xiao et al., 2023; Wang et al., 2023b), control generation, and align models with human preferences (Qi et al., 2024; Lee et al., 2024). These insights show promise in both understanding model mechanisms and practical applications.

In Chapter 4, we explore hidden representations in the reasoning processes of LLMs. This chapter highlights key observations from our experiments, focusing on how token and layer-level representations evolve across model layers (§4.1.1) and how specific tokens contribute to predictions (§4.1.1). We investigate the concept extraction process in the vocabulary space, demonstrating how pretrained vocabulary heads interpret hidden representations. We also conduct ablation studies to analyze the contributions of attention and MLP components within Transformer models (§4.1.1). The extracted concepts from hidden representations show a geometric structure that closely aligns with human understanding (§4.1). From a more module-specific perspective, we delve into knowledge neuron theory (Dai et al., 2022a; Geva et al., 2022b), conducting a case study on bias neuron detection in LLMs (§4.3.1). Our analysis reveals that bias neurons are distributed across multiple layers, emphasizing the importance of multi-layer inspection and manipulation when addressing bias in LLMs. Furthermore, we examine the predictive power of late-layer representations, showing that these layers not only contain rich information during inference but also offer potential for efficient task adaptation through late-layer tuning (§4.2).

Chapter 5 presents methods for model refinement and alignment, inspired by the observations in Chapter 4. We propose the Mixture-of-Depths (MoD) framework (§5.1) for tuning language models. Instead of relying solely on the final layer's output, MoD combines the predictions from the last k layers using routing weights, forming an ensemble (§5.1.1). Experimental results show that MoD improves performance on arithmetic and commonsense reasoning tasks, achieving results comparable to traditional fine-tuning methods while significantly reducing the number of trainable parameters (§5.1.2). Additionally, we explore the trade-off between performance and efficiency using sparse routing mechanisms (§5.1.3), demonstrating that MoD balances predictive accuracy and computational resource optimization.

We also introduce a lightweight debiasing approach using Localized Subspace Projection and Editing (LoPE) in §5.2, motivated by the bias neuron case study in §4.3.1. LoPE identifies bias neurons across multiple layers and mitigates their effects by projecting activations away from bias-related subspaces. This training-free method effectively reduces bias without compromising the model's ability to generate meaningful, coherent text. In §5.2.3, we show that LoPE outperforms other baseline debiasing methods, both in terms of bias mitigation and preserving language modeling quality. LoPE offers a practical solution for addressing bias in large language models rooted in explainability research.

In summary, this thesis explores the mechanisms underlying LLMs through the lens of explainability. By categorizing existing explainability methods, analyzing hidden representations, and proposing novel techniques for model refinement and bias mitigation, we aim to contribute to both the empirical understanding and practical advancements in LLM development. Our findings highlight the significance of explainability not only in understanding model behaviors but also in facilitating more effective model tuning and alignment with human preferences. Ultimately, this research seeks to bridge the gap between explainability and practical applications, providing pathways to develop more transparent, robust, and ethically aligned language models. The following chapters systematically present the methodologies, experiments, and results that support these contributions.

Chapter 2

Preliminaries

2.1 Notations

In this section, we define the notations used throughout the thesis and introduce the key components of a Transformer model, which is the backbone of large language models (LLMs). Table 2.1 summarizes the notations used in the following chapters.

Table 2.1: Summary of notations used in the thesis

Notation	Description
$\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$	Input sequence of tokens, length n
$\mathbf{h}^{(l)}$	Hidden states at layer l
L_i	The ith layer (block) of an LLM
d, N	Hidden dimension, total number of layers in an LLM
$\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$	Weight matrices for queries, keys, and values in attention
$\mathbf{W}_K^l, \mathbf{W}_V^l$	Weight matrices for key and value in 2-layer MLP
$\sigma(\cdot)$	Activation function in the feed-forward network (FFN)
$\phi(\cdot)$	The language modeling head
logits$_t$	Logits for the tth token
$\mathbf{A}, \mathbf{B}$	Low-rank matrices in LoRA (Low-Rank Adaptation)
Attn$^{(l)}$	Self-attention operation at layer l
MLP$^{(l)}$ or FFN$^{(l)}$	Feed-forward network at layer l
LN$^{(l)}$	Layer normalization at layer l
PE(pos, i)	Positional encoding at position pos and dimension i
RoPE$(\mathbf{x}, pos)$	Rotary position embedding at position pos
γ, β	Learnable parameters in layer normalization

2.2 Transformer Architectures

Before we go deep into the LLMs, the current SOTA models are mostly built upon transformer models, which is first introduced by Vaswani et al. (2017), is based on the attention mechanism, which allows the model to capture long-range dependencies within a sequence more effectively than previous recurrent neural networks (RNNs) (Schmidt, 2019) or convolutional neural networks (CNNs) (O'Shea and Nash, 2015) and empirically more suitable for constrcuting large scale models[1].

2.2.1 Multi-Head Attention Mechanism

The core component of the Transformer architecture is the multi-head self-attention mechanism (Vaswani et al., 2017), which enables the model to weigh the importance of different tokens in the input sequence when encoding each token. The self-attention mechanism computes attention scores for each pair of tokens and using these scores to generate a context-aware representation for each token. Multi-head allows the model to attend to different parts of the sequence in parallel and to capture diverse aspects of the relationships between tokens (Luo and Specia, 2024).

Formally, given an input sequence of tokens $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, the model first embeds these into a list of representations $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$. for each token x_i, the model computes a set of query vectors $\mathbf{q}_i^h$, key vectors $\mathbf{k}_i^h$, and value vectors $\mathbf{v}_i^h$ for each attention head $h = 1, \ldots, H$, where H denotes the total number of heads. The projections are computed as follows:

$$\mathbf{q}_i^h = \mathbf{h}\mathbf{W}_h^Q, \quad \mathbf{k}_i^h = \mathbf{h}\mathbf{W}_h^K, \quad \mathbf{v}_i^h = \mathbf{h}\mathbf{W}_h^V,$$

where $\mathbf{W}_h^Q$, $\mathbf{W}_h^K$, and $\mathbf{W}_h^V$ are learnable weight matrices specific to the h-th head.

The attention score e_{ij}^h between tokens x_i and x_j within head h is calculated as the scaled dot product of the corresponding query and key vectors:

$$e_{ij}^h = \frac{\mathbf{q}_i^h \cdot \mathbf{k}_j^h}{\sqrt{d_k^h}},$$

where d_k^h is normally the dimensionality of the key vectors for head h. The scaling factor $\sqrt{d_k^h}$ is introduced to control the magnitude of the dot product and stabilize the gradients during training. These attention scores are then passed through a softmax function to generate attention weights $\alpha_{ij}^h = \text{softmax}(e_{ij}^h)$, which are used to compute a weighted sum of the value vectors:

$$\text{head}_h(\mathbf{h})_i = \sum_{j=1}^{n} \alpha_{ij}^h \mathbf{v}_j^h. \tag{2.1}$$

Each attention head produces a sequence of output vectors $\text{head}_h(\mathbf{X})_i$ corresponding to each token x_i. These outputs are then concatenated and projected back into the original feature space using a linear transformation:

$$\text{Attn}(\mathbf{h})_i = \text{Concat}(\text{head}_1(\mathbf{h})_i, \ldots, \text{head}_H(\mathbf{h})_i)\mathbf{W}^O, \tag{2.2}$$

[1]There are also some other types of models like the State Space Models (SSMs) (Gu and Dao, 2024) showing promising ability for capturing long context information but we will focus on transformer-based LMs in this thesis.

where $\mathbf{W}^O$ is a learnable output projection matrix. The concatenation of the outputs from all heads allows the model to integrate information from multiple attention subspaces, enhancing the expressiveness of the resulting token representations.

The use of multiple attention heads provides the Transformer model with the ability to attend to different aspects of the input sequence simultaneously. For example, one head might focus on syntactic relationships between words, while another might focus on semantic connections (Voita et al., 2019). This parallel processing across heads enriches the overall representation learned by the model, contributing to its effectiveness across various NLP tasks.

2.2.2 Feedforward Neural Network

In addition to the attention mechanisms, each layer of the Transformer architecture includes a position-wise feedforward neural network (FFN) (Bebis and Georgiopoulos, 1994)[2]. The MLP operates independently on each token position and consists of two linear transformations with a non-linear activation function applied in between. Formally, the MLP can be expressed as:

$$\text{MLP}^l(\mathbf{h}) = \sigma(\mathbf{h}\mathbf{W}_K^l + \mathbf{b}_K^l)\mathbf{W}_V^l + \mathbf{b}_V^l, \tag{2.3}$$

where $\mathbf{W}_K^l$, $\mathbf{W}_V^l$, $\mathbf{b}_K^l$, and $\mathbf{b}_V^l$ are the learnable up-projection and down-projection weight matrices and bias vectors, and $\sigma(\cdot)$ denotes the activation function applied element-wise[3].

The MLP Sub-update

From the residual stream perspective, the residual stream is updated by the attention heads and MLP blocks from subsequent layers (bias terms omitted):

$$\mathbf{h}_i^{\ell+1} = \mathbf{h}_i^\ell + \text{MLP}^\ell(\mathbf{h}_i^\ell + \text{Attn}^\ell(\mathbf{h}_i^\ell)) \tag{2.4}$$

Several works have demonstrated that the updates to the residual stream from each MLP block can be further decomposed (Geva et al., 2022b; Dai et al., 2022b; Gurnee et al., 2023) (cf. the neuron theory in §3.1.1). Specifically, MLP blocks consist of two linear transformations, separated by pointwise non-linear activations σ:

$$\text{MLP}^\ell(\mathbf{h}^\ell) = \sigma\left(W_K^\ell \mathbf{h}^\ell\right) W_V^\ell, \tag{2.5}$$

where $W_K^\ell, W_V^\ell \in \mathbb{R}^{d_{mlp} \times d}$.

We denote the i-th row of W_K^ℓ as $\mathbf{k}_i^\ell$ (referred to as key vectors), and the i-th column of W_V^ℓ as $\mathbf{v}_i^\ell$ (referred to as value vectors). For simplicity, we may omit the ”MLP” prefix and use $\mathbf{k}_i^\ell$ and $\mathbf{v}_i^\ell$.

[2]FFN is normally used interchangeably with a two-layer multi-layer perceptron (MLP), and these two terms share the same meaning throughout this thesis.

[3]For some activation function like SiLU there might be three different weights including a gating weights instead of only W_K and W_V, we omit them for simplicity here

Equation (2.5) implies that the output of MLP blocks is the sum of the value vectors $\mathbf{v}_i^\ell$, each scaled by a corresponding coefficient m_i^ℓ, where $\mathbf{m}^\ell := \sigma\left(W_K^\ell \mathbf{h}^\ell\right) \in \mathbb{R}^{d_{mlp}}$:

$$\text{MLP}^\ell(\mathbf{h}^\ell) = \sum_{i=1}^{d_{mlp}} \sigma(\mathbf{h}^\ell \cdot \mathbf{k}_i^\ell)\mathbf{v}_i^\ell = \sum_{i=1}^{d_{mlp}} m_i^\ell \mathbf{v}_i^\ell. \tag{2.6}$$

In other words, the MLP block writes to the residual stream d_{mlp} times, once for each value vector. These updates are referred to as *sub-updates* in current work (Geva et al., 2022b; Lee et al., 2024).

Activation Functions in LLMs

The choice of activation function $\sigma(\cdot)$ in the FFN plays a crucial role in the model's performance and has been the subject of extensive research and experimentation in the development of LLMs (Shen et al., 2023). Below, we discuss several activation functions that are commonly used in state-of-the-art LLMs.

ReLU (Rectified Linear Unit) The ReLU (Agarap, 2019) activation function is defined as:

$$\sigma_{\text{ReLU}}(x) = \max(0, x),$$

where x represents the input to the activation function. ReLU is widely used due to its simplicity and effectiveness in mitigating the vanishing gradient problem. In the context of large language models, ReLU is used in the OPT series (Zhang et al., 2022). However, ReLU may suffer from the "dying ReLU" problem (Lu, 2020), where neurons can become inactive if the inputs are negative.

GELU (Gaussian Error Linear Unit) The GELU (Hendrycks and Gimpel, 2023) activation function is defined as:

$$\sigma_{\text{GELU}}(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right],$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution, and erf$(\cdot)$ denotes the error function. GELU introduces stochasticity into the activation function by blending the input with a probabilistic factor, making it smoother than ReLU. This activation function is used in models such as GPT-2 (Radford et al., 2019a), Pythia (Biderman et al., 2023), and the Gemma (Team et al., 2024) series.

SiLU (Sigmoid Linear Unit) The SiLU (Elfwing et al., 2017) activation function, also known as the swish activation, is defined as:

$$\sigma_{\text{SiLU}}(x) = x \cdot \sigma_{\text{sigmoid}}(x) = \frac{x}{1 + e^{-x}},$$

where $\sigma_{\text{sigmoid}}(x)$ is the sigmoid function. SiLU offers a balance between smoothness and non-linearity, providing a smooth, non-monotonic function that tends to perform well in deep learning models. It is used in advanced LLMs such as Qwen (Bai et al., 2023), Mistral (Jiang et al., 2023), and LLaMA (Touvron et al., 2023a,c) series, where its smooth gradient properties help in achieving better optimization during training, particularly in deeper networks.

2.2.3 Layer Normalization

First introduced by (Ba et al., 2016), layer normalization (LN) addresses the issue of internal covariate shift, where the distribution of inputs to each layer changes during training. By normalizing the inputs to each layer, LayerNorm ensures that the inputs have zero mean and unit variance, which helps in achieving faster convergence and enhancing the robustness of the model. Formally, given an input $\mathbf{h} \in \mathbb{R}^{n\times d}$, where n is the number of tokens and d is the feature dimensionality, LayerNorm is computed as follows:

$$\text{LayerNorm}(\mathbf{h}) = \frac{\mathbf{h} - \mu(\mathbf{h})}{\sigma(\mathbf{h}) + \epsilon} \odot \gamma + \beta, \tag{2.7}$$

where $\mu(\mathbf{h})$ and $\sigma(\mathbf{h})$ are the mean and standard deviation across the hidden dimension d, γ and β are learnable parameters that allow the model to scale and shift the normalized output, and ϵ is a small constant added for numerical stability to prevent division by zero.

RMSNorm RMSNorm (Zhang and Sennrich, 2019) is a variant of LayerNorm that simplifies the normalization process by normalizing the input using only the root mean square (RMS) of the features, without subtracting the mean. This can be particularly advantageous in scenarios where the mean subtraction in LayerNorm is less critical or where a more computationally efficient normalization method is desired. Formally, RMSNorm is defined as:

$$\text{RMSNorm}(\mathbf{h}) = \frac{\mathbf{h}}{\text{RMS}(\mathbf{h}) + \epsilon} \odot \gamma + \beta, \tag{2.8}$$

where $\text{RMS}(\mathbf{h}) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} h_i^2}$ is the root mean square across the hidden dimension, and γ and β are learnable scaling and shifting parameters, similar to those in LayerNorm. RMSNorm reduces the amount of computation and increases efficiency over LayerNorm and has been widely adopted in deep models like LLaMA (Touvron et al., 2023a,c).

Post-LN vs. Pre-LN Approaches

In Transformer architectures, normalization can be applied in different positions within each layer, leading to two common strategies: Pre-LN and Post-LN.

Post-LN In the original Transformer design by (Vaswani et al., 2017), normalization is applied after the sublayers (either the attention or the feedforward network), a strategy referred to as post-norm. In Post-LN models, each sublayer's output is first passed through the sublayer itself (attention or FFN), and then the resulting output is normalized:

$$\mathbf{h}_{out} = \text{LN}(\text{Sublayer}(\mathbf{h}_{in}) + \mathbf{h}_{in}),$$

where $\mathbf{h}_{in}$ is the input to the sublayer, and $\mathbf{h}_{out}$ is the final output of the post-norm operation. Unsupervised pre-trained models based on the Post-LN Transformer architecture also show impressive performance in many downstream tasks (Radford et al., 2019a; Devlin et al., 2019).

Pre-Norm Alternatively, in the Pre-LN approach, normalization is applied both before the input and inside the residual blocks, specifically before the Attention and FFN modules:

$$\mathbf{h}_{out} = \text{LN}(\mathbf{h}_{in}) + \text{Sublayer}(\text{LN}(\mathbf{h}_{in})),$$

In this case, the input $\mathbf{X}$ is normalized first, and then it is processed through the sublayer, with normalization also being applied inside the residual blocks before the Attn and FFN modules. Pre-LN models tend to stabilize the gradient flow, particularly in very deep networks, as the normalization ensures that each sublayer operates on inputs with a consistent distribution Nguyen and Salazar (2019). Xiong et al. (2020) show that the gradients in Pre-LN models are well-behaved at initialization, enabling warm-up-free training that achieves comparable results with significantly less training time.

2.2.4 Positional Encoding

The Transformer architecture does not inherently encode the order of tokens in a sequence, as it operates independently of the sequential nature of the input. To address this, positional encoding is introduced to provide the model with information about the relative or absolute positions of tokens within the sequence. Positional encoding vectors are added to the input embeddings, enabling the model to capture the sequential dependencies necessary for tasks such as language modeling and translation.

Sinusoidal Positional Encoding The most common approach to positional encoding, as introduced in the original Transformer model (Vaswani et al., 2017), uses sinusoidal functions to generate the positional encodings. These encodings are deterministic and provide a unique representation for each position in the sequence. The sinusoidal positional encoding is defined as:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad \text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right),$$

where pos is the position of the token in the sequence, i is the index of the dimension, and d_{model} is the dimensionality of the model's embeddings. The sine and cosine functions providing a smooth and continuous representation of position. The choice of using both sine and cosine allows the model to distinguish between different positions and their relative distances effectively.

Rotary Position Embedding (RoPE) While sinusoidal positional encoding is effective, it has limitations, particularly when dealing with tasks requiring the model to generalize positional information beyond the training data's range. To address this, Rotary Position Embedding (RoPE) was introduced in the RoFormer model (Su et al., 2023). RoPE encodes the position information by rotating the query and key vectors in the self-attention mechanism. Specifically, it applies a rotation matrix to the token embeddings, which allows the model to capture both absolute and relative positional relationships more effectively. Formally, given a token's position pos and its corresponding query or key vector $\mathbf{x} \in \mathbb{R}^d$, RoPE applies the following transformation:

$$\text{RoPE}(\mathbf{x}, pos) = \mathbf{x}^{(1:d/2)} \cos(\theta_{pos}) + \mathbf{x}^{(d/2+1:d)} \sin(\theta_{pos}),$$

where $\mathbf{x}^{(1:d/2)}$ and $\mathbf{x}^{(d/2+1:d)}$ are the first and second halves of the vector $\mathbf{x}$, and θ_{pos} is a position-dependent vector defined as:

$$\theta_{pos} = \left(\frac{pos}{10000^{2i/d}}, \frac{pos}{10000^{2i/d}}, \dots\right)_{i=1}^{d/2}.$$

This rotation mechanism allows RoPE to encode relative position information directly within the attention score calculations, thereby enhancing the model's ability to recognize positional patterns even when the tokens are far apart in the sequence.

2.3 Transformer-based Large Language Models

2.3.1 Residual Stream Flow in Transformer Layers

In Transformer architectures, such as those in the LLaMA series, each layer updates its hidden states by adding the outputs of the attention mechanism and the feed-forward network (FFN) to the previous hidden states via residual connections (He et al., 2016). These residual connections help preserve information across layers and facilitate the effective training of deep models by mitigating issues like vanishing gradients. Formally, the update for the hidden state $\mathbf{h}^{(l)}$ at layer l is given by:

$$\mathbf{h}^{(l)} = \mathbf{h}^{(l-1)} + \text{Attn}^{(l)}\left(\mathbf{h}^{(l-1)}\right) + \text{FFN}^{(l)}\left(\mathbf{h}^{(l-1)}\right), \tag{2.9}$$

These residual connections ensure that each layer builds upon the representations learned by earlier layers without overwriting or losing critical information. After the final layer, the output $\mathbf{h}^{(L)}$ is passed through the output head to produce the model's predictions.

2.3.2 Pretraining and Finetuning

Transformer-based LLMs are constructed by stacking multiple Transformer layers to form deep networks capable of learning intricate patterns from vast amounts of data. The development of LLMs typically involves two key phases: pretraining and finetuning. During the pretraining phase, the model is trained on a large corpus in an unsupervised manner to learn general-purpose language representations. This is achieved by optimizing the model on self-supervised tasks, where the model predicts parts of the input text based on the rest. Common pretraining objectives include:

- **Masked Language Modeling (MLM)**: Used in models like BERT (Devlin et al., 2019), where a percentage of tokens are randomly masked, and the model predicts the masked tokens based on the unmasked context.

- **Causal Language Modeling (CLM)**: Employed in autoregressive models like GPT (Radford et al., 2019a; Brown et al., 2020a) and LLaMA (Touvron et al., 2023a,c), where the model predicts the next token in a sequence given previous tokens.

Pretraining is conducted on vast datasets, often comprising billions of tokens from diverse sources such as books, websites, and social media (OpenAI, 2023b). This allows LLMs to acquire broad linguistic knowledge, which is later adapted to specific tasks through finetuning.

Finetuning, or Post-training, adapts a pretrained LLM to a downstream task. The model is trained on a labeled dataset, typically smaller than the pretraining data, to optimize performance on the specific task. For generation tasks, the objective is language modeling. Given an input sequence $\mathbf{x} = (x_1, \ldots, x_n)$, the optimization goal is to minimize the cross-entropy loss with the predicted output sequence $\mathbf{y} = (y_1, \ldots, y_m)$:

$$\mathcal{L}_{\text{LM}} = \min_{\Phi} \left\{ - \sum_{i=1}^{m} \log p_{\Phi} \left(y_i \mid \mathbf{x}, \mathbf{y}_{<i} \right) \right\} \tag{2.10}$$

Functional Heads

At the output of the final Transformer layer, LLMs typically include functional heads that are designed to produce task-specific outputs. The language modeling head is used in tasks like text generation, where the goal is to predict the next token in a sequence or to generate coherent text. This head typically consists of a linear layer followed by a softmax function, mapping the output of the last Transformer layer to a probability distribution over the vocabulary. Given $\mathbf{h}_t^L$ the hidden state corresponding to the t-th token from the last Transformer layer, a language modeling head is defined as:

$$\phi(\mathbf{h}_t^L) = \mathbf{h}_t^L \mathbf{W}_{\text{LM}} + \mathbf{b}_{\text{LM}}, \tag{2.11}$$

where $\mathbf{W}_{\text{LM}}$ and $\mathbf{b}_{\text{LM}}$ are the weight matrix and bias vector for the language modeling head. The softmax function is applied to the logits to obtain the probability distribution over the vocabulary, from which the next token is sampled or selected.

For classification tasks, a classification head $\phi_\theta(\cdot)$ takes the final-layer representation at the first token (CLS) and outputs a class distribution (Devlin et al., 2019):

$$\phi_\theta(\mathbf{h}_0^{(L)}) = \mathrm{softmax}\left(\mathbf{W}_o \tanh(\mathbf{W}_d \mathbf{h}_0^{(L)} + \mathbf{b}_d) + \mathbf{b}_o\right) \tag{2.12}$$

The optimization goal is to minimize the cross-entropy loss of the target class y given input $\mathbf{x}$ (single-label classification task as an example):

$$\mathcal{L}_{\text{CLS}} = \min_{\Phi,\theta} \{-\log \phi_\theta(y \mid \mathbf{h}_\Phi(\mathbf{x}))\} \tag{2.13}$$

In this thesis, we primarily focus on the language modeling head, which is widely adopted in decoder-only autoregressive models such as GPT (Radford et al., 2019a) and LLaMA (Touvron et al., 2023a,c) with causal language modeling (CLM) as their core training objectives.

2.3.3 Low-Rank Approximation for Finetuning

As the size of LLMs continues to grow, the computational cost and memory requirements for training and finetuning these models have become significant challenges. To address these challenges, low-rank approximation methods have been proposed as efficient alternatives to traditional finetuning techniques. One prominent approach is LoRA, introduced by Hu et al. (2021), which significantly reduces the number of trainable parameters and the associated computational overhead.

LoRA: Low-Rank Adaptation of Large Language Models The core idea behind LoRA is to decompose the weight updates during finetuning into low-rank matrices, which reduces the number of parameters that need to be learned, thereby making the finetuning process more efficient. Formally, consider a pretrained model with a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$ in one of its layers, where k is d or d_{mlp}, the dimension of MLP. During standard finetuning, the weight matrix is updated to a new matrix $\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W}$, where $\Delta\mathbf{W}$ represents the learned update. In LoRA, instead of learning a full-rank update $\Delta\mathbf{W}$, the update is approximated as a product of two low-rank matrices:

$$\Delta\mathbf{W} = \mathbf{A}\mathbf{B},$$

where $\mathbf{A} \in \mathbb{R}^{d \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times k}$ are the low-rank matrices, and r is the rank, typically chosen such that $r \ll \min(d, k)$. This low-rank decomposition significantly reduces the number of parameters that need to be learned during finetuning, from $d \times k$ to $r \times (d + k)$.

During finetuning with LoRA, the pretrained weights $\mathbf{W}_0$ remain frozen, and only the low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ are updated. The modified weight matrix during the forward pass is then:

$$\mathbf{W} = \mathbf{W}_0 + \mathbf{A}\mathbf{B}.$$

By constraining the update $\Delta\mathbf{W}$ to be low-rank, LoRA reduces both the memory footprint and the computational cost of finetuning, making it feasible to adapt very large models on limited hardware.

LoRA has been successfully applied to various LLMs and has proven effective in scenarios where fine-tuning large models is computationally prohibitive. Numerous LoRA variants have been proposed in recent years (Liu et al., 2024b; Zhang et al., 2023; Kopiczko et al., 2024). In this thesis, we focus on the standard LoRA approach for training large models (as discussed in §5.1), as it continues to demonstrate strong effectiveness and efficiency compared to other baselines in recent benchmarks (Hu et al., 2023).

Chapter 3

Related Work

The field of LLMs is rapidly advancing, making explainability not only a tool for understanding these complex systems but also essential for their improvement. This section categorizes current explainability approaches, highlights the challenges in ethical and controllable generation, and proposes research questions discussed in Chapter 4 and 5.

Categorization of Methods We present a structured categorization for the explainability methods and their applications in Figure D.1. We divide these into two broad domains: Local Analysis and Global Analysis, aligned with recent research in explainability (Zhao et al., 2023b; Luo and Specia, 2024). Local Analysis covers feature attribution and transformer block analysis, delving into detailed operations of models. Global Analysis, on the other hand, includes probing-based methods and mechanistic interpretability, offering a comprehensive understanding of model behaviors and capacities. Beyond understanding, we also explore applications of these insights in enhancing LLM capabilities, focusing on model editing, capability enhancement, and controlled generation.

3.1 Explainability for Large Language Models

3.1.1 Local Analysis

Local explanations in LLMs aim to elucidate how models generate specific predictions, such as sentiment classification or token predictions, for a given input. This section categorizes local explanation methods into two types: feature attribution analysis and analysis into individual Transformer (Vaswani et al., 2017) components.

Feature Attribution Explanation

Feature attribution, a local method for explaining a prediction, analysis quantifies the relevance of each input token to a model's prediction. Given an input text x with n tokens $\{x_1, x_2, ..., x_n\}$, a pre-trained language model f outputs $f(x)$. Attribution methods assign a relevance score $R(x_i)$ (Modarressi et al., 2022; Ferrando et al., 2022; Modarressi et al., 2023) to each token x_i, reflecting its contribution to $f(x)$. This category includes perturbation-based, gradient-based, and vector-based methods.

Perturbation-Based Methods. Perturbation-based methods, such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), alter input features to observe changes in model output. However, this removal strategy assumes input features are independent and ignores correlations among them. Additionally, models can be over-confidence even when the predictions are nonsensical or wrong (Feng et al., 2018). They also face challenges in efficiency and reliability highlighted in (Atanasova et al., 2020), leading to their diminished emphasis in recent attribution research.

Gradient-Based Methods. One might consider gradient-based explanation methods as a natural approach for feature attribution. This type of method computes per-token importance scores (Kindermans et al., 2016) using backward gradient vectors. Techniques such as gradient $\times$ input (Kindermans et al., 2017) and integrated gradients (IG) (Sundararajan et al., 2017) accumulate the gradients obtained as the input is interpolated between a reference point and the actual input. Despite their widespread use, one main challenge of IG is the computational overhead to achieve high-quality integrals (Sikdar et al., 2021; Enguehard, 2023) Their attribution score has also shown to be unreliable in terms of faithfulness (Ferrando et al., 2022) and their ability to elucidate the forward dynamics within hidden states remains constrained.

Vector-Based Methods. Vector-based analyses, which focus on token representation formation, have emerged as a key approach. Approaches range from global attribution from the final output layer to more granular, layer-wise decomposition of token representations (Chen et al., 2020; Modarressi et al., 2022) Consider decomposing the i^{th} token representation in layer $l \in \{0, 1, 2, ..., L, L+1\}$[1], i.e., $x_i^l \in \{x_1^l, x_2^l, ..., x_N^l\}$, into elemental vectors attributable to each of the N input tokens:

$$x_i^l = \sum_{k=1}^{N} x_{i \Leftarrow k}^l \tag{3.1}$$

The norm (Modarressi et al., 2022) or the L1 norm (Ferrando et al., 2022) of the attribution vector for the k^{th} input ($x_{i \Leftarrow k}^l$) can be used to quantify its total attribution to x_i^l.

Although several established strategies, such as attention rollouts (Abnar and Zuidema, 2020; Ferrando et al., 2022; Modarressi et al., 2022), focus on the global impact of inputs on outputs by aggregating the local behaviors of all layers, they often overlook the FFN in their analyses due to the inherent nonlinearities. Recent works have addressed this limitation by approximating and decomposing activation

[1]$l = 0$ is the input embedding layer and $l = L+1$ is the language model head over the last decoder layer.

functions, constructing decomposed token representations across layers (Yang et al., 2023; Modarressi et al., 2023).

Empirical evaluations demonstrate the efficacy of vector-based analysis, highlighting the potential of such methods in dissecting each hidden state representation within transformers. However, vector-based analysis often concentrates on end-to-end attribution and tends to omit detailed analysis of token representations across layers. In this thesis, we present novel insights into decomposing token representations from a layer-wise perspective, as detailed in §4.1.1.

Dissecting Transformer Blocks

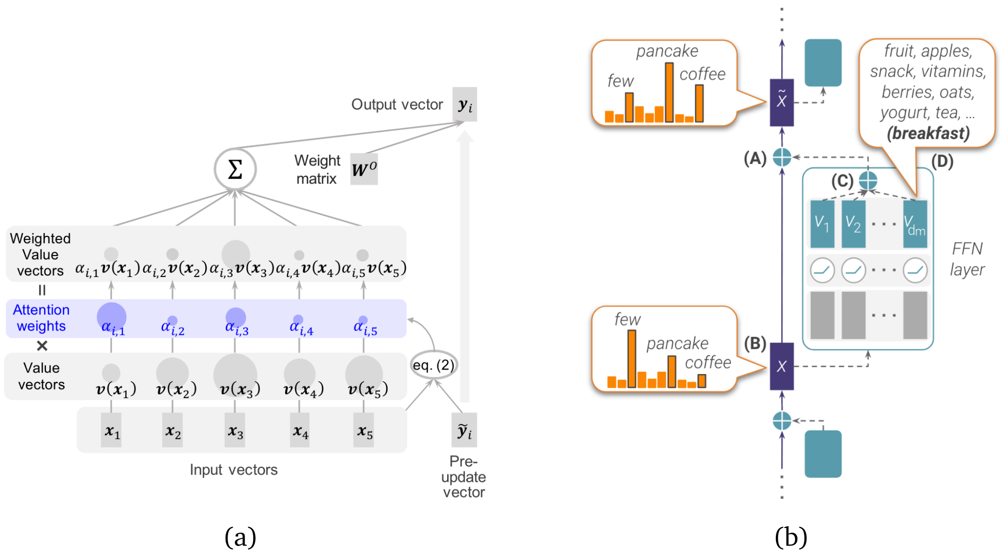
Output vector y_i
Weight matrix W^O
Weighted Value vectors $\alpha_{i,1}v(x_1)\alpha_{i,2}v(x_2)\alpha_{i,3}v(x_3)\alpha_{i,4}v(x_4)\alpha_{i,5}v(x_5)$
Attention weights $\alpha_{i,1}$ $\alpha_{i,2}$ $\alpha_{i,3}$ $\alpha_{i,4}$ $\alpha_{i,5}$
Value vectors $v(x_1)$ $v(x_2)$ $v(x_3)$ $v(x_4)$ $v(x_5)$
x_1 x_2 x_3 x_4 x_5
Input vectors
eq. (2)
$\tilde{y}_i$ Pre-update vector
(a)
pancake few coffee
fruit, apples, snack, vitamins, berries, oats, yogurt, tea, ... (breakfast)
(A) (B) (C) (D)
V_1 V_2 V_{dm}
FFN layer
few pancake coffee
(b)

Figure 3.1: Studied role of each Transformer component. (a) gives an overview of attention mechanism in Transformers. Sizes of the colored circles illustrate the value of the scalar or the norm of the corresponding vector (Kobayashi et al., 2020). (b) analyzes the FFN updates in the vocabulary space, showing that each update can be decomposed to sub-updates corresponding to single FFN parameter vectors, each promoting concepts that are often human-interpretable (Geva et al., 2022a).

Tracking Transformer block's *component-by-component internal processing* can provide rich information on its intermediate processing, given the stacked architecture of decoder-based language models (Kobayashi et al., 2023). In a transformer inference pass, the input embeddings are transformed through a sequence of L transformer layers, each composed of a multi-head self-attention (MHSA) sublayer followed by an MLP sublayer (Vaswani et al., 2017). Equation 2.9 shows the contribution from the l-th MHSA and MLP sublayers output to the residual stream[2]. While studies have frequently analyzed individual Transformer components (Kobayashi et al., 2020; Modarressi et al., 2022), the interaction between these sublayers is less explored, presenting an avenue for future research.

[2]For brevity, bias terms and layer normalization (Ba et al., 2016) are omitted, as they are nonessential for most of analysis.

Analyzing MHSA Sublayers. Attention mechanisms in MHSA sublayers are instrumental in capturing meaningful correlations between intermediate states of input that can explain the model's predictions. Visualizing attention weights and utilizing gradient attribution scores are two primary methods for analyzing these sublayers (Zhao et al., 2023b). Many studies have analyzed the linguistic capabilities of Transformers by tracking attention weights (Abnar and Zuidema, 2020; Katz and Belinkov, 2023; Kobayashi et al., 2023). For instance, attention mechanisms typically prioritize specific tokens while diminishing the emphasis on frequent words or special tokens, a phenomenon observable through norm-based analysis metrics, as illustrated in Figure 3.1(a) (Kobayashi et al., 2020). In the gradient analysis, some methods calculate gradients as partial derivatives of model outputs with respect to attention weights (Barkan et al., 2021), while others use integrated gradients, which are cumulative versions of these partial derivatives (Hao et al., 2021). Generally, these combined approaches, which integrate attention metrics with gradient information, tend to outperform methods using either metric in isolation.

Analyzing MLP Sublayers. More recently, a surge of works has investigated the knowledge captured by the feed-forward network (FFN) layers (Geva et al., 2022a; Dai et al., 2022c). These layers, which consume the majority of each layer's parameter budget—$8d^2$ compared to $4d^2$ for self-attention layers (where d represents the model's hidden dimension)—function similarly to key-value memory systems (Geva et al., 2021). In this context, each "key" corresponds to specific textual patterns learned during training, while each "value" generates a distribution over the output vocabulary (Geva et al., 2021). Together, they operate as "neurons," scaling and adding their contributions back to the residual stream to shape the learned representations (Dai et al., 2022a). Figure 3.1(b) illustrates the outputs of the FFN layers, demonstrating how each update within these layers can be decomposed into sub-updates linked to individual parameter vectors, which often encode interpretable concepts (Geva et al., 2022a). Additionally, there is growing interest in input-independent methods that analyze model parameters directly, bypassing the need for a forward pass (Dar et al., 2023). In this thesis, we present a detailed case study identifying specific value vectors associated with bias behavior in LLMs, as discussed in §4.1.

3.1.2 Global Analysis

In contrast to local analysis, which focus on elucidating individual model predictions, global analysis aims to understand and explain the knowledge or linguistic properties encoded in the hidden state activations of a model (Luo and Specia, 2024). This section explores two primary approaches to global analysis: probing methods and mechanistic interpretability (Transformer Circuits, 2022), an emerging perspective that seeks to reverse engineer the inner workings of deep neural networks.

Probing-Based Method

Self-supervised pre-training endows models with extensive linguistic knowledge, derived from large-scale training datasets. Probing-based methods are employed to capture the internal representations within these networks. This approach involves training a classifier, known as a probe, on the network's activations to distinguish between various types of inputs or outputs. In the following sections, we will discuss studies related to probing, categorized based on their objectives, whether it be probing for semantic knowledge or analyzing learned representations.

Probing Knowledge. LLMs trained on extensive text corpora, are recognized for their ability to encapsulate context-independent semantic and factual knowledge accessible via textual prompts (Petroni et al., 2019). Research in this area primarily focuses on formulating textual queries to extract various types of background knowledge from language models (Hewitt and Manning, 2019a; Peng et al., 2022). Interestingly, probes can sometimes unearth factual information even in scenarios where language models may not reliably produce truthful outputs (Hernandez et al., 2023a).

Probing Representations. LLMs are adept at developing context-dependent knowledge representations. To analyze these, probing classifiers are applied, typically involving a shallow classifier trained on the activations of attention heads to predict specific features. A notable study in this area involved training linear classifiers to identify a select group of attention heads that exhibit high linear probing accuracy for truthfulness (Li et al., 2024). This research revealed a pattern of specialization across attention heads, with the representation of "truthfulness" predominantly processed in the early to middle layers, and only a few heads in each layer showing standout performance. Such insights pave the way for exploring more complex representations. For instance, research by Li et al. (2023a) has revealed nonlinear internal representations, such as board game states, in models that initially lack explicit knowledge of the game or its rules.

Mechanistic Interpretability

Mechanistic interpretability seeks to comprehend language models by examining individual neurons and their interconnections, often conceptualized as circuits (Transformer Circuits, 2022; Zhao et al., 2023b). This field encompasses various approaches, which can be primarily categorized into three groups: circuit discovery, causal tracing, and vocabulary lens. Each of these approaches offers distinct perspectives and insights into the mechanisms of language models.

Circuit Discovery. The circuit-based mechanistic interpretability approach aims to align learned model representations with known ground truths, initially by reverse-engineering the model's algorithm to fully comprehend its feature set (Chughtai et al., 2023). A prominent example of this approach is the analysis of GPT-2 small

(Radford et al., 2019b), where a study identified a human-understandable subgraph within the computational graph responsible for performing the indirect object identification (IOI) task (Wang et al., 2022). In IOI, sentences like “When Mary and John went to the store, John gave a drink” are expected to be completed with “Mary”. The study discovered a circuit comprising 26 attention heads – just 1.1% of the total (head, token position) pairs – that predominantly manages this task. This circuits-based mechanistic view provides opportunities to scale our understanding to both larger models and more complex tasks, including recent explorations into In-Context Learning (ICL) (Halawi et al., 2023; Hendel et al., 2023; Todd et al., 2023; Wang et al., 2023b).

Causal Tracing. The concept of causal analysis in machine learning has evolved from early methods that delineate dependencies between hidden variables using causal graphs (Pearl et al., 2000) to more recent approaches like causal mediation analysis (Vig et al., 2020). This newer method quantifies the impact of intermediate activations in neural networks on their output (Meng et al., 2023a). Specifically, Meng et al. (2023a) assesses each activation’s contribution to accurate factual predictions through three distinct operational phases: a **clean** run generating correct predictions, a **corrupted** run where predictions are impaired, and a **corrupted-with-restoration** run that evaluates the ability of a single state to rectify the prediction (Meng et al., 2023a). Termed as *causal tracing*, this approach has identified crucial causal states predominantly in the middle layers, particularly at the last subject position where MLP contributions are most significant. This finding underscores the role of middle layer MLPs in factual recall within LLMs.

Vocabulary Lens. Recent work has suggested that model knowledge and knowledge retrieval may be localized within small parts of a language model (Geva et al., 2021) by projecting weights and hidden states onto their vocabulary space To analyze the components in vocabulary space, we read from each token component $\mathbf{h}_t^l$ at layer l by projecting with the language model head in Equation 2.11:

$$p_t^l = \text{softmax}\left(\phi(\text{LN}_{final}(\mathbf{h}_t^l))\right) \tag{3.2}$$

where LN_{final} stands for layer normalization before the LM head. Belrose et al. (2023) refines model predictions at each transformer layer and decodes hidden states into vocabulary distributions based on this method. Exploring this avenue further, Geva et al. (2022a) illuminated the role of transformer feed-forward layers in predictions, spotlighting specific conceptual emphases via FFN sub-updates. There is also a growing interest in input-independent methodologies, where model parameters are interpreted directly, bypassing a forward pass (Dar et al., 2023).

Augmenting projection-focused interpretations, Din et al. (2023a) first unveiled a feasible application for such projections, suggesting early exit strategies by treating hidden state representations as final outputs. Geva et al. (2023) pinpointed two critical junctures where information propagates to the final predictions via projections and attention edge intervention. While much of the focus has been on how hidden states relate to model outputs, recent works have also highlighted the roles of individual tokens, revealing that their contributions through attention outputs are laden with rich semantic information (Ram et al., 2023; Katz and Belinkov, 2023).

3.2 Leveraging Explainability

In this section, we explore how explainability can be leveraged as a tool to debug and enhance models. While numerous approaches aim to improve model performance through fine-tuning or retraining, our focus is on methods that are explicitly grounded in model explainability. These approaches not only provide insights into the internal workings of the models but also guide targeted improvements. Several works discussed here serve as baseline methods, as detailed in Chapter 5.

3.2.1 Parameter Space Editing

Despite the ability to train proficient LLMs, the methodology for ensuring their relevance and rectifying errors remains elusive. In recent years, there has been a surge in techniques for editing LLMs. The goal is to efficiently modify the knowledge or behavior of LLMs within specific domains without adversely affecting their performance on other inputs (Yao et al., 2023).

Hypernetwork Knowledge Editors. This type of knowledge editors includes *memory-based* model and editors with *additional parameters*. Memory-based models store all edit examples explicitly in memory based on the explainability finding of key-value memories inside the FFN (Section 3.1.1). They can then employ a retriever to extract the most relevant edit facts for each new input, guiding the model to generate the edited fact. SERAC (Mitchell et al., 2022), for instance, adopts a distinct counterfactual model while leaving the original model unchanged. Editors with additional parameters introduce extra trainable parameters within LLMs. These parameters are trained on a modified dataset while the original model parameters remain static. For example, Huang et al. (2023) integrates one neuron (patch) for one mistake in the last layer of the FFN of the model, which takes effect only when encountering its corresponding mistake.

Locate-Then-Edit. The locate-then-edit paradigm first identifies the parameters corresponding to the specific knowledge and then modifies them by directly updating the target parameters. The Knowledge Neuron (KN) method (Dai et al., 2022c) introduces a knowledge attribution technique to pinpoint the “knowledge neuron” (a key-value pair in the FFN matrix) that embodies the knowledge and then updates these neurons. ROME (Meng et al., 2023a) and MEMIT (Meng et al., 2023b) apply causal tracing (Section 3.1.2) to locate the editing area. Instead of modifying the knowledge neurons in the FFN, ROME alters the entire matrix. Based on these two methods, PMET (Li et al., 2023b) involves the attention value to achieve better performance.

3.2.2 Enhancing Model Performance

While LLMs demonstrate versatility in various NLP tasks, insights from explainability can significantly enhance these capabilities. This section highlights two key tasks where explainability has shown considerable impact in recent work: improving the utilization of long text (Xiao et al., 2023; Liu et al., 2023; Pope et al., 2022) and enhancing the performance of In-Context Learning (ICL) (Hendel et al., 2023; Halawi et al., 2023; Wang et al., 2023b).

Improving Context Length Capacity The optimization of handling long text aims to enhance the ability of LLMs to capture and effectively utilize content within longer contexts. This is particularly challenging because LLMs tend to struggle with generalizing to sequence lengths longer than what they were pretrained on, such as the 4K limit for Llama-2 (Touvron et al., 2023b). Beltagy et al. (2020) maintains a fixed-size sliding window on the key-value (KV) states of the most recent tokens. While this approach ensures constant memory usage and decoding speed after the cache is initially filled, it faces limitations when the sequence length exceeds the cache size (Liu et al., 2023). An innovative solution proposed by Xiao et al. (2023) takes advantage of the MHSA explanations (Section 3.1.1) in LLMs, which allocates a significant amount of attention to the initial tokens. They introduce StreamingLLM, a simple and efficient framework that allows LLMs to handle unlimited text without fine-tuning. This is achieved by retaining the ”attention sink,” which consists of several initial tokens, in the KV states. The authors also demonstrate that pre-training models with a dedicated sink token can further improve streaming performance.

Improving In-Context Learning In-context Learning (ICL) has emerged as a powerful capability alongside the development of scaled-up LLMs (Brown et al., 2020b). ICL stands out because it doesn’t require extensive updates to the vast number of model parameters and relies on human-understandable natural language instructions (Dong et al., 2023). As a result, it offers a promising approach to harness the full potential of LLMs. With mechanistic interpretability (Section 3.1.2), Wang et al. (2023b) reveal that label words in the demonstration examples function as anchors, which can be used to improve ICL performance with simple anchor re-weighting method. Halawi et al. (2023) study harmful imitation in ICL through vocabulary lens to inspect a model’s internal representations (Section 3.1.2), and identify two related phenomena: *overthinking* and *false induction heads*, the heads in late layers that attend to and copy false information from previous demonstrations, and whose ablation improves ICL performance. Furthermore, using causal tracing (Section 3.1.2), Hendel et al. (2023); Todd et al. (2023) find that a small number attention heads transport a compact representation of the demonstrated task, which they call a *task vector* or *function vector* (FV). These FVs can be summed to create vectors that trigger new complex tasks and improve performance for few-shot prompting (Todd et al., 2023).

Improving Model Efficiency Early exit strategies in language models are often explored to improve efficiency. Several works focus on enhancing inference efficiency by terminating computation at dynamically-decided earlier layer outputs (Xin et al., 2020; Schuster et al., 2022). A common approach for adapting intermediate layer output to language modeling involves training an affine transformation (Belrose et al., 2023; Din et al., 2023b). Early exit strategies have also been explored for interpretability, analyzing the linearity properties of transformer components (Geva et al., 2023; Hernandez et al., 2023b). However, the utilization of intermediate layer output during training remains largely unexplored. A recent work (Elhoushi et al., 2024) applies layer dropout and an early exit loss to increase the accuracy of early exits, but its primary focus is still on inference efficiency. To the best of our knowledge, our work is the first to utilize early exit logits together with the final layer logits to incorporate task-aware representations from intermediate layers into the loss calculation.

3.2.3 Controllable Generation

Despite the superior performance of large language models in text generation, they often struggle with producing factually accurate and controllable (safe) content Li et al. (2024). Introducing explainability methods offers the potential to develop inference-time techniques that enhance model factuality, calibration, and controllability, aligning outputs more closely with human preferences (Luo and Specia, 2024).

Reducing Hallucination Hallucinations in LLMs refer to generated content not based on training data or facts, various factors such as imperfect learning and decoding contribute to this (Ji et al., 2023). To mitigate hallucinations, initial approaches used reinforcement learning from human feeback (Ouyang et al., 2022) and distillation into smaller models such as Alpaca (Li et al., 2023c). Leveraging explainability provides a significantly less expensive way to reduce hallucination, enjoying the advantage of being adjustable and minimally invasive. For example, Li et al. (2024) use as few as 40 samples to locate and find "truthful" heads and directions through a trained probe (Section 3.1.2). They propose inference-time intervention (ITI), a computationally inexpensive strategy to intervene on the attention head to shift the activations in the "truthful" direction, which achieves comparable or better performance toward the instruction-finetuned model.

Logit-Level Arithmetic Operations at the logit level have proven effective in steering the output of LLMs (Luo and Specia, 2024). From a multi-model perspective, there has been a growing body of work focusing on "mixturing" the abilities of different trained models in line with the Mixture-of-Experts framework (Shazeer et al., 2017; Jiang et al., 2024). Liu et al. (2021); Gera et al. (2023) have also shown the effectiveness of ensembling logits from multiple LMs. From a single model perspective, contrasting logits from different layers of a model (Chuang et al., 2024; Gera et al., 2023) has shown promising performance improvements in the trustworthiness of generation and addressing the resource-intensive issues of larger models

(Liu et al., 2024a). In §5.1 of our thesis, we proposed a new tuning method built upon logit-level arithmetic and follows the line of ensembling logits, focusing not on a multi-model perspective but rather on utilizing the late layers' outputs within a single model for tuning. This approach has been considered only during inference in previous work.

Ethical Alignment

As research on AI fairness gains increasing importance, efforts have been made to detect social bias (Fleisig et al., 2023; An and Rudinger, 2023) and suppress toxicity (Gehman et al., 2020; Schick et al., 2021) in language models (LMs). Many previous debiasing methods (Qian et al., 2022) have focused on constructing anti-stereotypical datasets and subsequently retraining the LM from scratch or conducting fine-tuning. While effective, these approaches entail high costs for both data construction and model retraining. Additionally, they are susceptible to catastrophic forgetting when fine-tuning is applied (Zhao et al., 2023b). Despite the growing focus on fairness, limited work has addressed the interpretability aspect of debiasing research. A recent study by dev (2023) explores the interpretation and mitigation of social biases in large language models (LLMs) by introducing the concept of *social bias neurons*. Inspired by the gradient-based attribution method, Integrated Gradients (IG) (Section 3.1.1), the authors propose an interpretable technique termed Integrated Gap Gradient (IG^2). This method identifies social bias neurons by back-propagating and integrating the gradients of the logits gap for a selected pair of demographics[3]. By identifying these neurons, they are able to suppress their activations to mitigate bias.

Extensive experiments have demonstrated the effectiveness of this approach in masked language models (MLMs) and have indicated its potential for ethical alignment. However, few studies have addressed performance in open-generation models for causal language modeling (CLM). In this work, we aim to provide further insights (§4.3.1) and explore new strategies (§5.2) to tackle this challenge.

[3]Demographics include attributes such as gender, sexuality, and occupation. Nine common demographic categories are collected, and pairs of demographics are selected to reveal fairness gaps (Liu et al., 2024c).

Chapter 4

The Hidden Representations of Large Language Models

In this chapter, we present key observations from our experiments, focusing on the hidden representations within the reasoning processes of large language models. Our objective is to apply various techniques to extract meaningful and human-understandable concepts from these representations. We also highlight interesting patterns and attributes that emerge across different layers of the model, some of which provide valuable insights that can enhance the methods discussed in Chapter 5.

4.1 Concept Extraction in the Vocabulary Space

In this section, we demonstrate how to extract semantically meaningful information from hidden representations using pretrained vocabulary heads, as discussed in §2.3.2. For a pretrained language model head, it is possible to examine the token representation process not only at the output layer but across every layer of hidden representations. Additionally, this head can be used to probe other representations, such as those produced by the attention heads (§3.1.1) or MLP neurons (§3.1.1). For our analysis, we primarily use the pretrained 125M GPT-2 model (Radford et al., 2019a) as a representative of smaller LLMs, and the 7B LLaMA model (Touvron et al., 2023a) as an example of medium to large models.

4.1.1 The Evolving Token Expressions

We analyze three types of prompts in our study. The first is knowledge extraction prompts (Meng et al., 2023a), which follow a (subject, relation, object) format and are designed to extract the correct answer for the object. The second type is the IOI (Indirect Object Identification) prompt (Wang et al., 2022), where sentences such as:

```
Prompt:  When Mary and John went to the store, John gave a drink
to
Target:  Mary
```

The sentence should be completed with `Mary` because it is the linguistically appropriate completion. This prompt allows for an interpretable algorithmic understanding. Lastly, we include gender bias (occupation) prompts (Hernandez et al., 2023b), which are structurally similar to knowledge extraction prompts but focus on detecting inherent biases within the model's outputs. Our analysis aims to address three primary questions:

1. Which layer of the model contributes the most (§4.1.1)?
2. Which Transformer module (e.g., attention or FFN) plays the largest role (§4.1.1)?
3. Which token representation is the most influential in the final prediction (§4.1.1)?

Which layer contributes the most?

We first show that the token expression process inside the language models is involving and individual tokens are different. The following figures show logit lens (Nostalgebraist, 2020) for each layer outputs for the knowledge extraction prompt, example as:

```
Prompt:  The Eiffel Tower is located in the city of
Target:  Paris
```

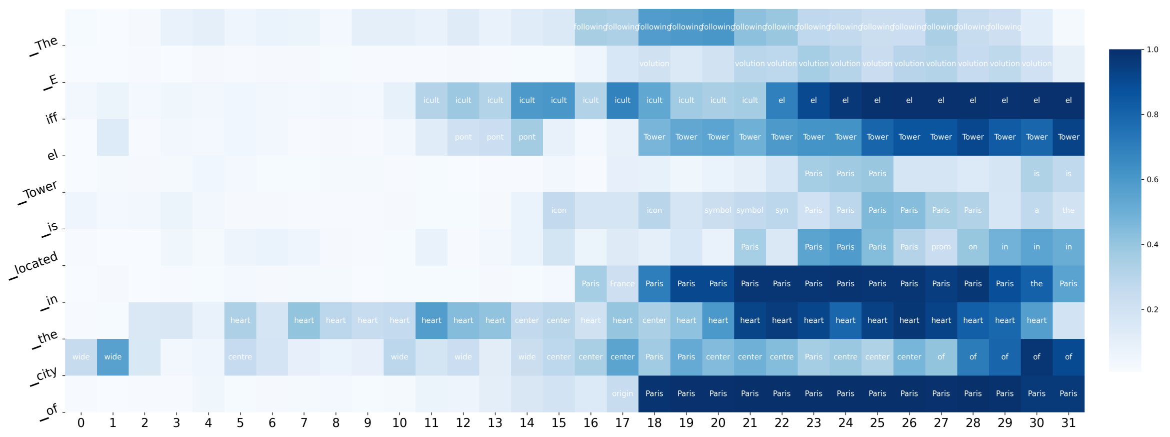

Figure 4.1: Projection of hidden representations onto the vocabulary space using the method from nostalgebraist (2021) for the knowledge extraction prompt (Meng et al., 2023a) at each token position. The color of each pixel represents the probability of a token when projected onto the vocabulary space. Annotations indicate the predicted token when its probability exceeds 0.2.

The color of the pixel represents the probabilities of that layer output for the most confident token, with annotations indicating the output token when the probability exceeds 0.2. Heatmaps for other types of prompts are included in Appendix B.1.

Model Uncertainties and Early Exit We observed that the model typically exhibits confident predictions in the later layers (for a 32-layer 7B model, this usually refers to layers after the 18th to 20th). As shown in the Figure 4.1, for tokens such as `Tower` following `Eiffel`, and `Paris` following `in the city of...`, the model assigns a large probability score, indicating high confidence in predicting these tokens. Notably, once the probability of a token becomes high, it often remains high for the subsequent layers. Recent works (Elhoushi et al., 2024; Corro et al., 2024) have demonstrated that this phenomenon can serve as evidence for early exit strategies, which can significantly accelerate LLM inference by halting computation once confidence thresholds are met.

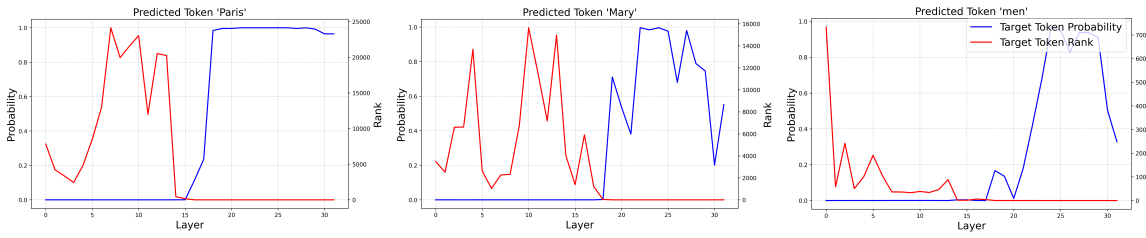

Figure 4.2: The layer-wise probabilities and ranks of the target token in the vocabulary space. The left panel shows the knowledge extraction prompt (Meng et al., 2023a), the middle panel shows the IOI prompt (Wang et al., 2023a), and the right panel shows the gender bias prompt (Hernandez et al., 2023b). We observe a synchronous trend between rank and probability: as the token's rank improves (with rank 0 being the highest), the confidence in predicting that token, reflected by its probability, increases significantly.

Uncertainty is another important metric for measuring the confidence of an LLM (Xiong et al., 2024). In addition to the probability of the most likely token, metrics such as the rank of that token and the entropy of the output distribution can also provide valuable insights. These measures are expected to follow similar trends as the probability analysis discussed in §4.1.1. For example, a gender bias prompt might be:

```
Prompt:  What gender (men or women) are pilots associated with?
They are associated with
Target:  men
```

In Figure 4.2, we demonstrate that the rank and probability of the target token often follow a synchronous trend: as the token's rank improves (with rank 0 being the top rank), the confidence in predicting that token (reflected by its probability) increases significantly. However, we also observe that for certain datasets, the token rank can rise to a high position in the early layers, even when the probability remains low. This suggests that the model begins to capture the correct output representation early on, with subsequent layers refining that representation.

Given the low uncertainty at specific layers, it raises the question of whether some early layer outputs could already serve as final predictions. We observe cases where later layers interfere with earlier, potentially correct, representations. For instance, in Figure B.1, the last two token positions exhibit high confidence at layer 29 in the IOI task, correctly identifying the target token. Yet, in the subsequent two layers, the confidence diminishes. This highlights the potential of selectively utilizing outputs from certain layers that demonstrate high confidence with the correct prediction, allowing the model to learn which layer's output it should trust. This idea will be further explored in both §4.2 and §C.

Which Transformer module contributes the most?

Transformer-based language models typically consist of attention outputs and MLP outputs, both sequentially written to the residual stream. To better understand the contributions of different layers, we explored the specific impact of these components—particularly the Attention and MLP modules.

We performed an ablation analysis on all modules at every token position given a prompt. Several methods can be employed for ablation, such as zeroing out the components (in the residual stream dimension, not the head dimension), mean ablation (Wang et al., 2023a), and corruption by patching, as summarized by (Zhang and Nanda, 2024). For this study, we utilized zero ablation as a straightforward approach to examine the effects of these modules. We applied this analysis to a prompt addressing gender bias, considering several ablation scenarios:

1. Ablation of the attention output to the residual stream.
2. Ablation of the MLP output to the residual stream.
3. Ablation of both attention and MLP outputs (equivalent to ablating an entire layer).
4. For LLaMA-type models (Touvron et al., 2023a,c), where attention and MLP updates occur sequentially, we also included the ablation of the attention output to the MLP—meaning the attention output updates the residual stream but is excluded from the MLP input.

The evaluation metric was the logit difference of the target token; a larger value indicated a greater effect of the ablation on that position. The four ablation scenarios are illustrated in Figure 4.3.

Our findings suggest that the most significant effects of the attention and MLP modules are concentrated in the early layers. This implies that the final token primarily completes its extraction of information from preceding tokens in the early to middle layers. In contrast, at the position of the final token, the influence of the attention output becomes more pronounced in the later layers. This behavior is likely due to the nature of the gender bias task, which does not require substantial MLP activation to retrieve pre-trained knowledge (An and Rudinger, 2023), as is typically

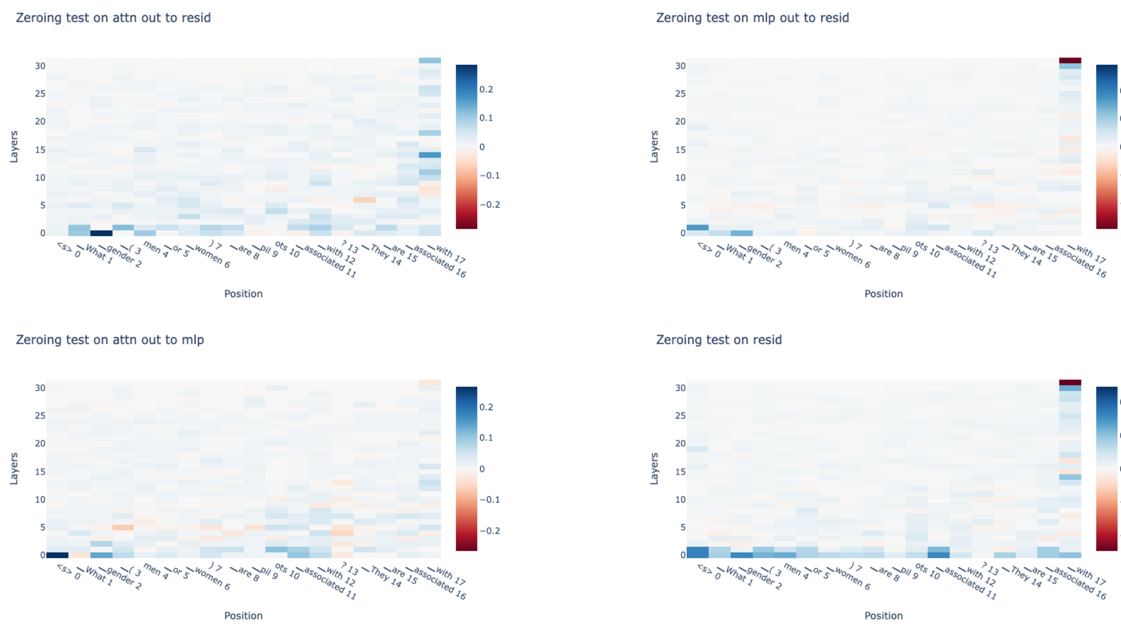

Figure 4.3: Zeroing ablation (Wang et al., 2023a) applied to all modules at each token position for the gender bias prompt (Hernandez et al., 2023b). The evaluation metric used is the logit difference of the target token before and after ablation; a larger value indicates a greater effect of the ablation on the corresponding position for the ablated module.

necessary in knowledge extraction tasks (Hernandez et al., 2023b). Instead, the task relies more heavily on attention mechanisms to finalize contextual understanding.

As the scale of large language models (LLMs) continues to grow, tuning these models has become increasingly computationally expensive and memory-intensive, posing significant challenges for deployment in industrial settings. Many applications focus on finetuning pretrained models on task-specific data using methods such as low-rank approximation (LoRA) (§2.3.3). However, these methods often struggle to match the performance of full parameter tuning in large-scale finetuning tasks, as LoRA's representational capacity is inherently limited by its low-rank structure (Pan et al., 2024). Recent research has aimed to identify task-relevant subsets of parameters in pretrained models for full-rank finetuning while reducing computational costs (Panigrahi et al., 2023; Zhang et al., 2024). We leave the exploration of integrating the intrinsic low-rank characteristics with precise tuning of specific sparse effective parameters in future work.

Which token representation contributes the most?

While ablation analysis provides insights into the contributions of different modules, it does not offer a precise measurement of contributions at the individual token level (Zhang and Nanda, 2024). Inspired by language model attribution analysis, as introduced in §3.1.1, we propose to investigate which tokens contribute the most through layer-wise decomposed vector representations, as opposed to traditional

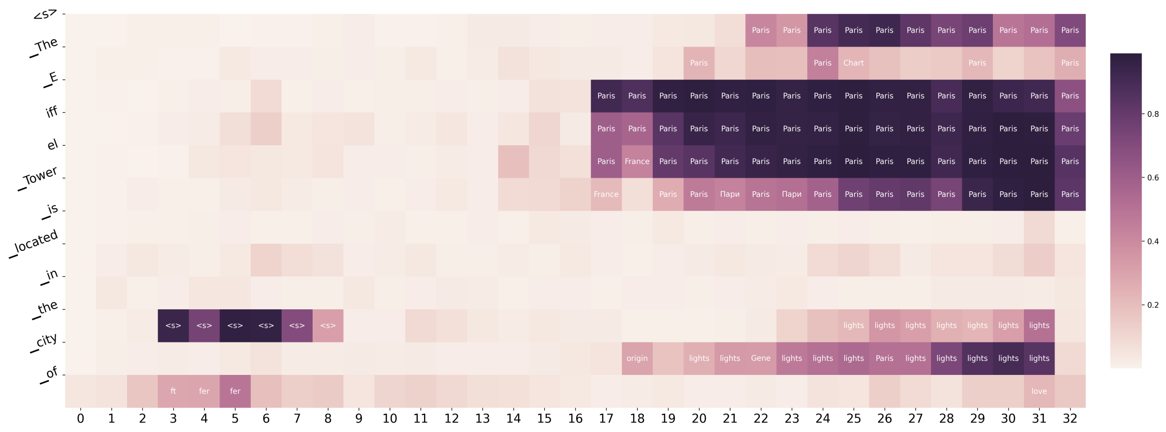

Figure 4.4: Decomposition at the final token position of the preceding token representations for the knowledge extraction prompt (Meng et al., 2023a). The color of each pixel represents the probability of that token when projected onto the vocabulary space using the method from nostalgebraist (2021). Annotations indicate the predicted token when the probability exceeds 0.2.

attribution analysis that focuses on end-to-end analysis (Sundararajan et al., 2017; Kobayashi et al., 2020; Sikdar et al., 2021; Modarressi et al., 2022; Ferrando et al., 2022).

In transformer-based language models, the computation of attention is inherently linear, allowing us to decompose the attention output by the token representations it attends to (in autoregressive models, these are the preceding tokens). The derivation of the attention decomposition can be found in Appendix B.3. However, the MLP module presents a challenge due to its activation function, which introduces nonlinearity, complicating the decomposition of its outputs into individual token contributions. Inspired by recent linear approximation methods (Yang et al., 2023; Modarressi et al., 2023), we can approximate the MLP output in terms of token representations, thereby enabling attribution analysis on the tokens for a given prompt, particularly at the position of the final token.

After the attention module, tokens are processed through a 2-layer multilayer perceptron (MLP) with a non-linear activation function σ_{act}:

$$\mathbf{z}^l_{\text{MLP}} = \sigma_{\text{act}}(\underbrace{\tilde{\mathbf{z}}^l_i \mathbf{W}^1_K}_{\zeta^l_i})\mathbf{W}^2_V \tag{4.1}$$

where $\tilde{\mathbf{z}}^l_i$ is the normalized input. The bias term is omitted for simplicity, which is common in many state-of-the-art LLMs (Touvron et al., 2023c; Team et al., 2024).

To continue the decomposition through the non-linear activation, we approximate σ_{act} using a piecewise linear function, assuming monotonicity and $\sigma_{\text{act}}(0) = 0$ (Modarressi et al., 2023). The approximation is applied elementwise, with the slope θ_t calculated as:

$$\theta_t = \frac{\sigma_{\text{act}}(\mathbf{z}^{(t)})}{\mathbf{z}^{(t)}} \tag{4.2}$$

where (t) denotes the dimension of the input vector $\mathbf{z}$. This approximation allows for a decomposition that preserves the token representation. Thus, the decomposed

FFN output becomes:

$$\mathbf{z}^{l}_{\mathrm{MLP},i} = \sum_{k=1}^{N} \theta^{(\zeta_i^l)} \odot \zeta^{l}_{i \Leftarrow k} \tag{4.3}$$

This approach enables the decomposition of token representations across layers, allowing for token-level attribution throughout the entire model. Combining this with the logit lens technique (nostalgebraist, 2021), we applied the linear approximation to the 7B LLaMA model (Touvron et al., 2023a). The results, shown in Figure 4.4, are consistent with prior analyses when performing decomposition at the last token position (i.e., using all preceding token representations to predict the next token) using a knowledge extraction prompt as an example. Notably, we observe that the representation `Paris` is primarily derived from the tokens `Eiffel Tower` and is activated in the later layers. This finding aligns with our intuition that the model infers `Paris` from the most representative token in the prompt. The heatmaps for other types of decomposed representations are included in Appendix B.2.

4.1.2 Concepts in the Representation Space

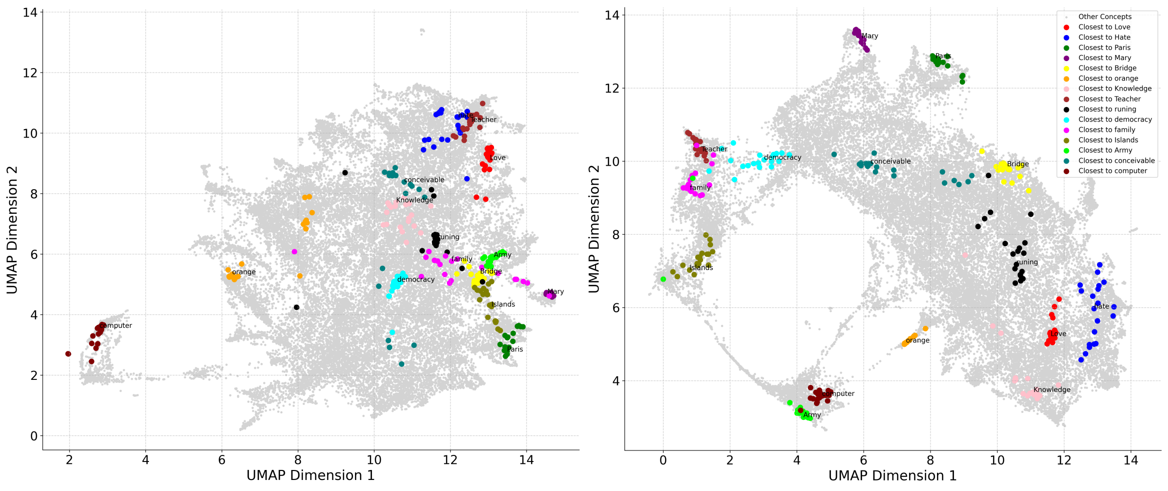

Figure 4.5: The Representation (Activation) Space map at the 25 layer of LLaMA3-8B (left) and LLaMA3-8B-Instruct (right) with the concepts from PaCE-1M. The visualization is the first two dimensions of UMAP of the concept vectors. We query GPT-4 for 15 concepts that are semantically different from each other and plot their clusters onto the map. We observe that concepts of similar semantics are clustered together, indicating that the activation space has clear semantic structures.

Recent work has shown that certain directions in the representation (activation) space of LLMs are associated with specific semantic concepts (cf. §3.1.2 and 3.1.2). In an ideal scenario, we would be able to manipulate the representation space using these identified linguistic directions. However, existing studies provide only a limited number of concept directions, making it difficult to steer or decompose representations that encompass all linguistic concepts.

Building on a recent study by Luo et al. (2024), which introduces a large dataset called PaCE-1M, we aim to investigate the geometry and potential applications of the representation space. The PaCE-1M dataset consists of 40,000 concepts, each represented by approximately 30 prompts that describe various scenarios generated by GPT-4. Further details on the dataset are provided in Appendix A.2.

Following the preprocessing steps described in Luo et al. (2024), for each concept t_i (out of a total of m concepts), we extract a direction $\mathbf{v}_i^l$ from the activations of its contextual stimuli at the l-th decoder layer. This results in a dictionary $\mathbf{D}^l \in \mathbb{R}^{d \times m}$ per layer (details in Appendix A.2). We explore the concept representations in the activation space of two advanced models: the recent LLaMA3-8B model and its chat-aligned version, LLaMA3-8B-Instruct (Dubey et al., 2024).

We begin by examining the semantic structure of the PaCE-1M dataset concepts using UMAP. Figure 4.5 visualizes the first two dimensions, with 15 concept queries generated by GPT-4 that are semantically distinct. The concept vectors are plotted, and semantically similar concepts form clusters, indicating that the activation space exhibits clear semantic structures.

We further observe that in the instructed-tuning version of the model (LLaMA3-8B-Instruct), the concept embeddings are more widely dispersed across the concept map compared to LLaMA3-8B. This suggests that the additional alignment and instruction tuning have strengthened the model's ability to distinguish between aligned and unaligned contexts. This observation aligns with findings from recent red-teaming literature (Qi et al., 2023, 2024).

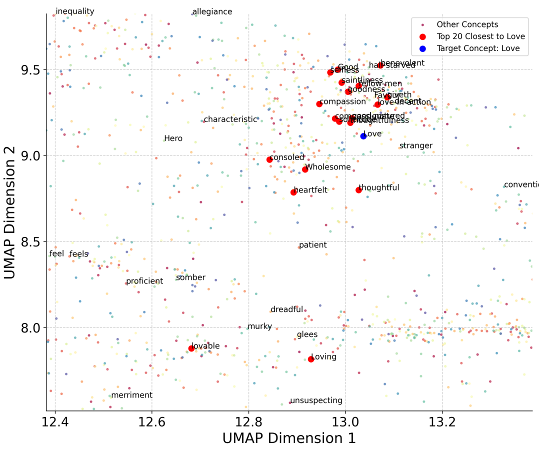

Figure 4.6: The zoom-in version around the target concept `Love`. The visualization is the first two dimensions of UMAP of the concept vectors. We observe similar concepts such as 'goodness', 'thoughtful', 'lovable' are clustered closely together.

Zooming in on specific concepts, we examine `Love` in Figure 4.6 for LLaMA3-8B. Similar concepts, such as 'goodness', 'thoughtfulness', and 'lovable', cluster closely together. Furthermore, a clear boundary exists between concepts related to `Love` and those related to `Hate` (the blue cluster), even though they are proximate in the overall concept map.

4.2 The Predictive Power of Late Layer Representations

Transformer-based LLMs process sequences of input tokens by representing them as vectors and transforming these vectors through multiple layers of transformers Vaswani et al. (2017). Previous section has demonstrated that these intermediate hidden states can carry meaningful information (see §4.1), and prior research has shown that leveraging these hidden states during decoding can improve trustworthiness (Li et al., 2024; Chuang et al., 2024) and reasoning capabilities (O'Brien and Lewis, 2023). However, how to effectively utilize these intermediate layers during training remains unexplored. While each layer transformation creates new token representations added to the residual stream (see §2.3.1), only the final layer representations are used to obtain training loss. Consequently, loss minimization directly optimizes these final representations, leaving hidden representations optimized only implicitly, thereby obscuring their potential predictive power.

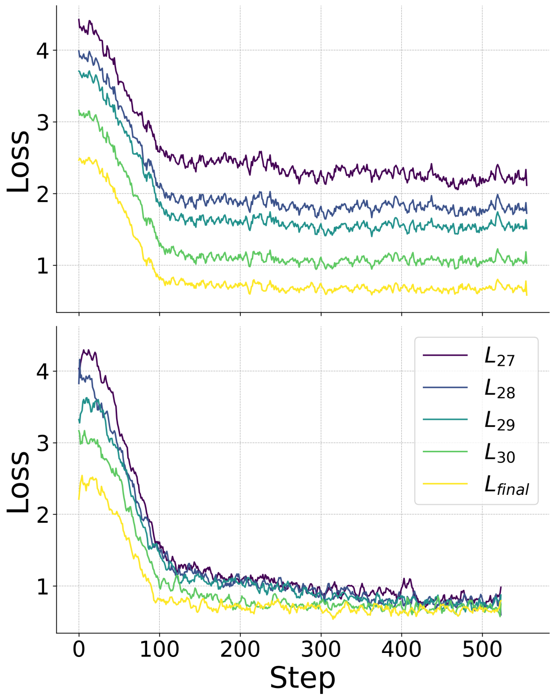

Figure 4.7: Tuning loss curves for LLaMA2-7B (Touvron et al., 2023c) on ARC dataset (Clark et al., 2018). Above shows the loss curve of late layers when optimizing the loss based on the last layer output when late layers are optimized implicitly; Below shows the loss curves when optimizing the loss on each late layer output explicitly.

We investigate the predictive power of the late layers representations,[1] which have proven to be task-aware in early exiting language models (Schuster et al., 2022; Din et al., 2023b). We begin by training models on late layers by applying the pretrained language model heads to each layer's output to calculate the loss. Our initial observations show that the training loss curves for each of the later layers started worse but eventually converged to similar levels, despite not including the weights of the subsequent layers (Figure 4.7). Figure 4.8 demonstrates that the trained "models" at these layers can even provide complementary evaluation results. These findings suggest that the late layers possess significant predictive potential. Given the overparameterization typical in large language models (Gao et al., 2023), the model can adapt effectively to downstream tasks even with fewer parameters.

We demonstrate that this observation of the predictive power in the later layers

[1] "Late" layers often refer to those closer to the output, e.g., layers 25-32 in a 32-layer LLaMA 7B models in different literatures (Din et al., 2023b; Geva et al., 2023; Meng et al., 2023a).

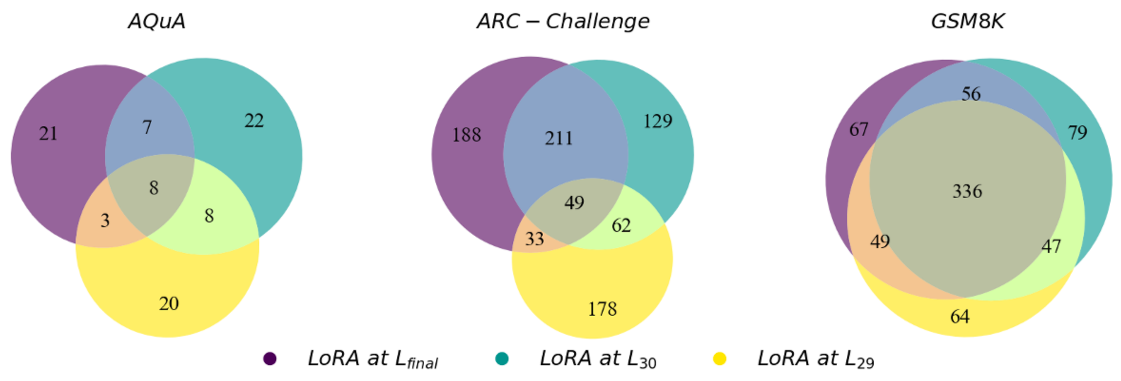

Figure 4.8: Intersection of solved problems by tuning loss layers on the AQuA (Ling et al., 2017), ARC-Challenge (Clark et al., 2018), and GSM8K (Cobbe et al., 2021b) datasets. The digits in the Venn diagram illustrate the number of overlapping solved problems and the complementary solved problems for each method.

can be leveraged to design more efficient and effective tuning methods, as discussed in §5.1.

4.3 The MLP Neuron Theory

Recent work (Geva et al., 2022a, 2023) demonstrates that for each sub-update, the value vector $\mathbf{v}_i$ either promotes or suppresses the likelihood of a token w being generated (Lee et al., 2024):

$$p\left(w \mid \mathbf{h}^\ell + m_i^\ell \mathbf{v}_i^\ell, E\right) \propto \exp\left(\mathbf{e}_w \cdot \mathbf{h}^\ell\right) \cdot \exp\left(\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell\right), \tag{4.4}$$

where $\mathbf{e}_w$ is the embedding of the token w. This equation indicates that when $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell > 0$, the likelihood of w increases, and when $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell < 0$, the likelihood decreases. For a detailed derivation, refer to Appendix D.2.

It is important to note that this dot product can be further decomposed. Specifically, $\mathbf{e}_w \cdot \mathbf{v}_i^\ell$ represents a "static" value that does not depend on the input. The influence of the input on the likelihood of w only becomes apparent when $\mathbf{v}_i^\ell$ is scaled by m_i^ℓ, which is determined by its corresponding key vector $\mathbf{k}_i^\ell$ and the residual stream $\mathbf{h}^\ell$. Therefore, the projection $\mathbf{r}_i^\ell = E\mathbf{v}_i^\ell \in \mathbb{R}^{|\mathcal{V}|}$ induces a ranking of tokens that are promoted by the value vector $\mathbf{v}_i^\ell$. Tokens with the highest dot products, $\mathbf{e}_w \cdot \mathbf{v}_i^\ell$, are the most strongly promoted by $\mathbf{v}_i^\ell$. In Section 4.3.1, we demonstrate how value vectors that promote bias outputs can be identified by applying projections and linear probings.

4.3.1 A Case Study in the Bias Neurons

Previous work has conducted mechanistic case studies on detoxifying language models (Lee et al., 2024). However, few studies have addressed debiasing LLMs from a mechanistic perspective, as biases encoded in these models are often challenging to identify and eliminate. For instance, when dealing with toxic information, we can identify a set of tokens that consistently refer to toxicity and aim to reduce the probability of generating these tokens. This approach may serve as a straightforward solution. In contrast, biases are often more insidious, as the tokens themselves may appear benign.

For example, consider the prompt: “People from [City] are known for their work ethic.” While this phrase may seem neutral, it can imply a bias against individuals from other cities by perpetuating a stereotype. Here, the tokens “People,” “known,” and “work ethic” are benign, yet the overall sentence structure conveys a biased viewpoint. Such examples, including those discussed in the gender bias prompt in §4.1.1, illustrate the complexity of identifying bias in language models. The underlying bias may not be directly tied to specific tokens, making it challenging to address using traditional debiasing techniques (Liu et al., 2024c).

Probing into the Bias Subspace

We begin by training a linear probe model on a dataset designed to elicit bias or stereotypical model generation. We use the Steroset (Nadeem et al., 2020) with the LLaMA2-7B model, which consists of 17,000 sentences measuring model preferences across gender, race, religion, and profession. The dataset is split 80:20 for training and validation.

Previous work on truthful direction (Li et al., 2024) and toxic representations (Lee et al., 2024) focused on training on the residual stream of the last layer, averaged across all timesteps ($\bar{\mathbf{x}}^{L-1}$), as shown below:

$$P(\text{Attribute}|\bar{\mathbf{x}}^{L-1}) = \text{softmax}(\theta_{\text{Attribute}}\bar{\mathbf{x}}^{L-1}), \quad \theta_{\text{Attribute}} \in \mathbb{R}^d$$

where `Attribute` denotes the type of representation being probed. Both analyses achieved high accuracy when probing the last layer representation. In our analysis of Steroset, we also achieve an average accuracy of 94% on the validation split.

Thus, we view the probe vector θ_{Bias} as an aggregate of all relevant signals in the language model for classifying an input as a biased representation. However, we extend the probing to all layers of the LLM. Surprisingly, we find that almost all layers achieve high accuracy when training the probe on the residual stream of that layer, with a minimum accuracy of 83% in layer 0. This indicates that there exists a linearly separable direction from the early layers to the final layers, enabling effective distinction between biased and unbiased representations.

Visualizing the geometry of “bias”. To further investigate the high-dimensional boundary separating the two classes, we follow Li et al. (2024) and train a second linear probe $p_{\theta'}$ on the same training set, constrained such that $\theta' \perp \theta$. This orthogonality to the first truthful direction allows θ' to best separate the two classes, maximizing the informativeness of our visualization. We visualize the geometry projected onto θ and θ' in Figure 4.9, depicting different layer probing results. In the second layer on the left, we observe considerable overlap between the two clusters, although the second probe still yields better-than-chance accuracy, suggesting that the representation of ”bias” lies not only in a single direction but also within a subspace even in the early layers.

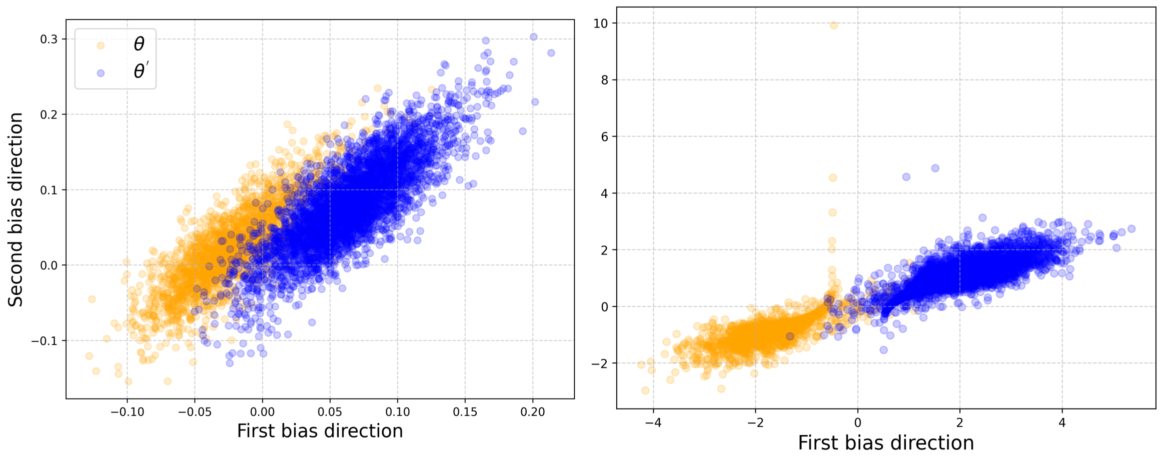

Figure 4.9: Visualization of the bias representation in the second and last layers of the LLaMA2-7B model. The left panel displays the overlapping clusters in the second layer, indicating that bias is represented within a subspace. In contrast, the right panel reveals distinct clusters in the final layer, suggesting that deeper layers capture bias representations in a richer and more nuanced manner. Both probes achieve high accuracy, reinforcing the complexity of bias representations.

When θ' is trained on the later layers, we notice increasing accuracy as we progress deeper into the network. For the probes trained in the last layer, as shown in Figure 4.9, distinct clusters emerge while both probes maintain 94% accuracy. This indicates that deeper layers capture additional information not present in earlier layers. The second probe identifies an orthogonal direction in deeper layers that further separates the classes, demonstrating that deeper layers contain more complex or refined features that enhance class distinction. This also confirms that the concept of ”bias” is not confined to a single direction but lies in a **subspace** that becomes more informative as the LLM processes the input.

Bias Neurons across Layers

Given the probe vector θ^ℓ trained on each layer of an LLM, we can utilize these vectors to identify weights within the language model that promote bias. We start by searching for W_K^ℓ (see Equation 2.5) that foster bias by examining the value vectors with the highest cosine similarity to θ^ℓ. For every probe vector, we compare it with all value vectors in the model and identify the top k neurons. The resulting dot heatmap is illustrated in Figure 4.10, where the color of each dot represents the neurons identified by the ℓth probe when $k = 10$.

Our findings indicate that the identified bias neurons from different probes

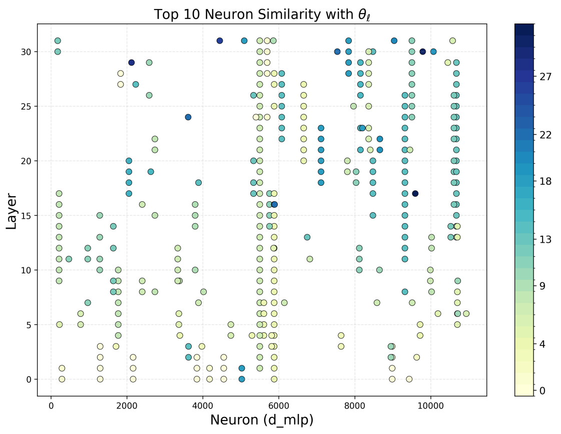

Figure 4.10: Dot heatmap illustrating the top 10 neurons identified by each probe across different layers. The color of each dot corresponds to the probe layers that bias neurons associated with.

vary, suggesting that each θ^ℓ contributes to a unique direction for classifying representations as biased. Generally, the identified bias neurons are closer to the layer on which the probe is trained; however, there are exceptions. For instance, the bias neurons associated with the last layer probe θ^L are dispersed across various layers (see the darkest dots and the vertical dot lines). This observation demonstrates that bias neurons are encoded throughout the layers of pretrained language models.

Previous work has attempted to identify toxic vectors using probes trained on the last layer and detoxify language models by subtracting these toxic vectors from the residual stream (Lee et al., 2024). However, as shown in Figure 4.10, this approach is insufficient since identifying neurons based solely on the last layer representations overlooks many bias neurons present in the middle and early layers. Thus, to effectively remove or suppress undesirable properties in an LLM, it is essential to consider representations across all layers that contribute to identifying those attributes. We explore the use of multi-layer probing information to debias language models in §5.2

Chapter 5

Improved Refinement and Alignment

5.1 Language Model as Mixture-of-Depths Ensembles

5.1.1 Mixture-of-Depths Framework

Following the observation from §4.2, we introduce the *Mixture-of-Depths* (MoD) framework. Unlike "mixture-of-experts" paradigm which utilized different trained models as experts for processing different input tokens Jiang et al. (2024), We propose the "mixture" across layers within a single model, where each layer output can be treated as a single model output. This approach allows us to add diversity and additional predictive power without significantly increasing parameters by training a simple gating network for the i-th late layer (§5.1.1).

We focus on tuning large language models. Our framework can be applied on top of any training methods as the hidden state dimensions remain consistent during training. Traditionally, language model heads in LLMs are trained to unembed embeddings from the last transformer layer. Applying the LM head directly to late layers during tuning can result in worse initial training performance, as shown in Figure 4.7. To ensure LM adaptation during tuning without interfering with the original model predictions, we apply an additional model distillation loss (§5.1.1) where the last layer output serves as the teacher. This method does not add any additional trainable parameters and ensures that the late layers adapt to the predictions. Experiments (§5.1.2) demonstrate that applying MoD tuning consistently improves performance on arithmetic and commonsense reasoning tasks with a minimal increase in trainable parameters (+0.04%). Furthermore, by replacing traditional trainable modules with MoD, we achieve similar performance with 97% fewer trainable parameters.

As analysis (§5.1.3), we study the learned patterns by MoD routing (§5.1.3), evaluate the performance when varying values of k, and explore the tradeoff between performance and efficiency (§5.1.3, 5.1.3).

Early-Exit for Late Layers

The idea of applying language heads directly to the hidden states of the middle layers, known as *early exit* (Teerapittayanon et al., 2016; Elbayad et al., 2020; Schuster

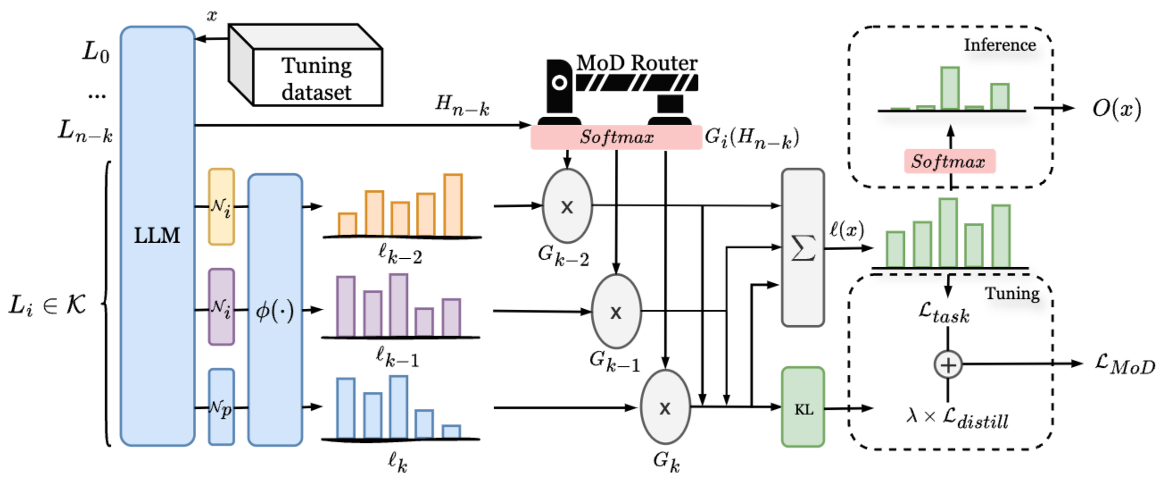

Figure 5.1: The overall framework of Mixture-of-Depths (MoD), which can be applied on top of any tuning method like LoRA (Hu et al., 2022). Given a pre-trained LLM and a tuning dataset, MoD applies trainable normalization $\mathcal{N}_k$ and pre-trained language model heads $\phi(\cdot)$ to the last k layers $\{L_{n-k+1}, \ldots, L_n\}$. Each layer's output is combined using learned routing weights to produce the final logits. During training, a auxiliary teacher-enforced distillation loss $\mathcal{L}_{distill}$ is applied, where the final layer output serves as the teacher. MoD utilizes the ensemble logits during inference.

et al., 2022), has proven effective even without a special training process (Kao et al., 2020). The residual connections (He et al., 2016) in transformer layers allow hidden representations to evolve gradually, enabling the formation of task-aware representations without abrupt changes.

Given a sequence of tokens $\{x_1, x_2, \ldots, x_{t-1}\}$, the embedding layer first converts the tokens into a sequence of vectors $\mathbf{h}^{(0)} = (\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_n^{(0)})$, where $\mathbf{h}_t^{(0)} \in \mathbb{R}^d$ and d is the hidden state dimension. This sequence $\mathbf{h}^{(0)}$ is then processed successively by each transformer layer, with the output of the j-th layer denoted as $\mathbf{h}^{(j)}$. The vocabulary head $\phi(\cdot)$ then outputs the logits logits_t of the next token x_t over the vocabulary set $\mathcal{V}$:

$$\text{logits}(x_t \mid x_{<t}) = \phi\big(\mathcal{N}_p(\mathbf{h}_t^{(N)})\big)\big)_{x_t}, \quad x_t \in \mathcal{V}.$$

Here, $\mathcal{N}_p$ is the pre-trained normalization module before the vocabulary head. This method is often considered a form of logit lens (Nostalgebraist, 2020), which uses the vocabulary head to probe into inner representations. However, the trainable predictive power of these representations remains unexplored. In §5.1.1, we show how to combine the train-time predictive power of late layers with final layer logits.

MoD Routing Network

Instead of applying $\phi(\cdot)$ only on the final layer, we incorporate the predictive power of late layers into the final prediction. We want to route the most informative representation for training to the final logit calculation. Motivated by the MoE framework

(Fedus et al., 2022; Jiang et al., 2024), the output of the ensemble logits is given by:

$$\sum_{i=0}^{k-1} G(x)_i \cdot \text{logits}_i(x).$$

Here, $G(\cdot)_i$ denotes the k-dimensional output of the routing network for the i-th expert, and $\text{logits}_i(\cdot)$ is the output logits of the i-th late layer. Here, $x = H_{n-k}$, which is the output of the layer before the last k layer. The routing network $G(x)_i$ is implemented by taking the softmax over a linear layer:

$$G(x) := \text{Softmax}(x \cdot \mathbf{W}_g).$$

The final logits are then obtained by summing the weighted logits from k layers:

$$\ell(x_t \mid x_{<t}) = \sum_{i=0}^{k-1} G(x)_i \cdot \text{logits}_i(x)$$

Additionally, one advantage of the MoD is its potential to improve inference efficiency by avoiding excessive computation when the routing vector is sparse. Following Shazeer et al. (2017), we achieve this by applying the softmax over the Top-K logits of the linear layer:

$$G_{\text{TopK}}(x) := \text{Softmax}(\text{TopK}(x \cdot \mathbf{W}_g)),$$

where $(\text{TopK}(\text{logits}))_i := \text{logits}_i$ if logits_i is among the top-K coordinates of logits $\text{logits} \in \mathbb{R}^k$ and $(\text{TopK}(\text{logits}))_i := -\infty$ otherwise.

In our main experiments (§5.1.2), we utilize $G(x)$ to demonstrate the effectiveness of the MoD framework. We investigate the performance and efficiency tradeoffs of using $G_{\text{TopK}}(x)$ in §5.1.3. This exploration allows us to understand how sparse routing mechanisms can optimize computational resources while maintaining predictive accuracy.

Late Layers Adaptation by Normalization and Distillation

Directly combining the logits of late layers using the LM head can result in worse training loss at the start of tuning (Figure 4.7). Previous works (Belrose et al., 2023) have attempted to learn an affine matrix A_ℓ to map hidden states of layer ℓ to the input space of the LM head. We aim to investigate more efficient adaptation methods while minimizing interference with model predictions and avoiding excessive additional trainable parameters.

Inspired by normalization studies in neural networks and the effectiveness of tuning the normalization module for domain adaptation (Zhao et al., 2023a), we propose tuning an additional normalization module for each late layer as a simple yet powerful adaptation method. We set the additional normalization module $\mathcal{N}_k$ to match the architecture of the pretrained $\mathcal{N}_p$. For instance, in the LLaMA2 model (Touvron et al., 2023c), we follow the LayerNorm setting (Ba et al., 2016). The learnable parameters in the normalization, γ_k and β_k, are trained individually for each k-th late layer to ensure specific adaptation for each layer.

Following our assumption in §5.1.1, we treat each of the $k-1$ late layers (excluding the final layer) as smaller models, with the final layer as the larger model with the most predictive power. We use the final layer as the teacher model to supervise the output of earlier layers for adaptation. We define a teacher-enforced distillation loss that measures the difference between the predictions of the intermediate models and the final layer's predictions. The distillation loss is computed as the sum of the KL divergence between each intermediate layer's output distribution P_i and the final layer's output distribution P_n:

$$\mathcal{L}_{distill} = \sum_{i=0}^{k-2} \mathrm{KL}(P_i \parallel P_n),$$

where P_i is the output distribution of layer i, and P_n is the output distribution of the final layer. The final loss is then the sum of the task loss and the distillation loss:

$$\mathcal{L}_{\text{MoD}} = \mathcal{L}_{task} + \lambda \mathcal{L}_{distill},$$

where λ is a hyperparameter that controls the weight of the distillation loss. By tuning with the normalization modules and distillation loss, we adapt the $k-1$ layer representations to be more suitable for the language modeling task, ensuring their contributions are aligned with the original task loss.

5.1.2 Experimental Evaluation

We evaluate the MoD framework on two types of language modeling tasks: arithmetic reasoning and commonsense reasoning. The MoD framework minimally increases trainable parameters and can be integrated with any existing training method, as the hidden state dimensions remain consistent during training. We use LoRA (Hu et al., 2022) as our base tuning method, which has been shown to reduce the number of tunable parameters while maintaining performance comparable to full finetuning. We define a single LoRA layer as L_{LoRA}. We use two baselines:

1. The model tuned with LoRA excluding the last k layers, denoted as LoRA$_{\neg\mathcal{K}}$.
2. The model tuned with LoRA on all layers, denoted as LoRA$_{\text{all}}$.

The notation LoRA$_{\text{all}}$ represents the model tuned with LoRA applied to all layers, including the last k layers which is identical to LoRA$_{\neg\mathcal{K}} + L_{\text{LoRA}} \times |\mathcal{K}|$ specified in the tables.

As shown in Table 5.1, MoD consistently improves performance when applied on top of LoRA$_{\text{all}}$ with minimally added parameters. Though MoD is not designed as an additional training architecture, experiments also demonstrate that it can replace the LoRA module while retaining similar or even better performance with 97% [1] fewer trainable parameters. We conduct experiments with LLaMA-1 (Touvron

[1]The percentage is calculated by the additional parameters introduced by MoD divided by the additional parameters introduced by LoRA$_{all}$.

Table 5.1: Accuracy comparison of MoD built upon LoRA (Hu et al., 2022) for LLaMA-7B (Touvron et al., 2023a) and LLaMA2-7B (Touvron et al., 2023c) on seven arithmetic reasoning datasets. We train the models on a single combined dataset follow Hu et al. (2023) and report averaged performance of three runs with distinct random seeds. The number in parentheses (%) indicates the percentage of added trainable parameters relative to the LoRA$\neg|\mathcal{K}|$ baseline.

Method	AddSub	AQuA	GSM8K	MAWPS	MultiArith	SingleEq	SWAMP	Avg.
LLaMA-7B								
LoRA$_{\neg\mathcal{K}}$	38.7	13.4	37.3	56.3	78.2	59.8	42.3	46.6
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ (+10.3%)	41.3	15.4	38.5	58.0	81.0	**62.9**	44.2	48.8
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ + **MoD$_{\mathcal{K}}$ (+10.4%)**	**42.0**	15.8	**39.1**	**58.5**	**81.3**	**62.9**	**44.9**	**49.2**
+ **MoD$_{\mathcal{K}}$ (+0.04%)**	41.5	**16.1**	38.2	58.4	80.7	62.3	43.8	48.7
LLaMA2-7B								
LoRA$_{\neg\mathcal{K}}$	46.3	20.5	39.7	60.6	81.4	62.0	43.2	50.5
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ (+10.3%)	51.1	24.4	43.6	62.6	84.2	66.9	47.7	54.5
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ + **MoD$_{\mathcal{K}}$ (+10.4%)**	**51.2**	**25.5**	**43.9**	63.1	**84.3**	**67.3**	**48.0**	**54.8**
+ **MoD$_{\mathcal{K}}$ (+0.04%)**	50.1	24.3	43.4	**63.7**	82.2	66.8	47.5	54.0

et al., 2023a) and LLaMA-2 (Touvron et al., 2023c) models with 7B parameters. The weight of the distillation loss λ is set to 0.0001 for all datasets and models, and the routing network is Gaussian initialized with a standard deviation of 0.02 and a mean of 0. All experiments are run on NVIDIA A6000 GPUs. Detailed experimental settings are provided in Appendix C.1.

Arithmetic Reasoning

Arithmetic reasoning includes seven datasets for math word problems: AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021a), MAWPS (Koncel-Kedziorski et al., 2016), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021). Models need to generate chain-of-thought (Wei et al., 2022) reasoning steps before the final answer. We replicate the experimental setup from Hu et al. (2023) on a combined dataset of these seven arithmetic reasoning tasks with LM-generated chain-of-thought steps (MATH7K) and report scores on all test sets. We only evaluate the correctness of the final numeric or multiple-choice answer. Details of the dataset are provided in Appendix A.1.1. For MATH7K, we set k to 3 for both LLaMA-1 and LLaMA-2 models across all datasets. Note that different models and datasets might benefit from a different value of k, or we could dynamically select k during training, which we leave for future research.

The results in Table 5.1 show that the MoD framework consistently improves performance on arithmetic reasoning tasks when applied on top of LoRA$_{\neg\mathcal{K}}$. Furthermore, MoD alone, even with only 0.19% added parameters, provides competitive performance with LoRA$_{\text{all}}$. These results validate our approach of utilizing late layer during training to enhance model performance in complex reasoning tasks.

Commonsense Reasoning

Commonsense reasoning includes four datasets: the Challenge Set and Easy Set of ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), and OBQA (Mihaylov et al.,

Table 5.2: Accuracy comparison of MoD on four commonsense reasoning datasets. We train the models on each dataset and report the averaged performance of three runs with distinct random seeds. The number in parentheses (%) indicates the percentage of added trainable parameters relative to the LoRA$\neg|\mathcal{K}|$ baseline.

Method	**ARC-e**	**ARC-c**	**BoolQ**	**OBQA**	**Avg.**
LLaMA-7B					
LoRA$_{\neg\mathcal{K}}$	75.3	39.0	65.1	78.4	64.5
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ (+10.3%)	**79.6**	42.0	68.2	79.8	67.4
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ + **MoD**$_{\mathcal{K}}$ **(+10.4%)**	**79.6**	**47.2**	**69.8**	**80.1**	**69.2**
+ **MoD**$_{\mathcal{K}}$ **(+0.04%)**	78.1	43.1	69.2	79.6	67.5
LLaMA2-7B					
LoRA$_{\neg\mathcal{K}}$	75.9	48.0	70.5	80.4	68.7
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ (+10.3%)	81.8	53.8	70.9	82.0	72.1
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ + **MoD**$_{\mathcal{K}}$ **(+10.4%)**	82.2	**56.4**	71.4	**83.4**	**73.4**
+ **MoD**$_{\mathcal{K}}$ **(+0.04%)**	**82.9**	53.4	**71.5**	82.1	72.6

2018a). The task is formulated as a multiple-choice problem. We follow the setup from Hu et al. (2023) but train each dataset separately to evaluate the effectiveness of our MoD framework on individual datasets. Details of the datasets are provided in Appendix A.1.2. We follow the same settings as in §5.1.2. Similarly, Table 5.2 shows the impact of the MoD framework on commonsense reasoning tasks. The addition of MoD to LoRA results in consistent performance with a minimal increase in trainable parameters, reinforcing the practicality of our approach.

5.1.3 Analysis

Using the training setup from §5.1.2, we conducted several analyses on our MoD framework. We examined the sparsity curve of the routing network at the route level across training tokens (§5.1.3), explored the advantages and trade-offs of sparse routing (§5.1.3), and performed ablation studies on the different components in MoD (§5.1.3).

Learned Routing Pattern Across Tokens

In this section, we analyze the routing patterns learned with MoD for the $\mathcal{K}$ ensemble layers during training. With a Gaussian-initialized routing network, we measure the sparsity of the weights across the training tokens, i.e., how many weights are close to zero. We calculate the proportion of weights below a threshold, ϵ, which we set to 1×10^{-5}. A lower level of sparsity often implies that the model is selectively using current routes while ignoring others, leading to the discussion in §5.1.3. We also record the mean and variance to measure the tendency and dispersion for each k route, as detailed in Appendix C.2. We evaluate MoD trained on top of LoRA and MoD trained without k LoRA layers using the LLaMA 7B model on the ARC easy subset.

According to Figure 5.2, we notice an interesting learned pattern discrepancy between MoD trained with or without LoRA layers. When trained without k LoRA

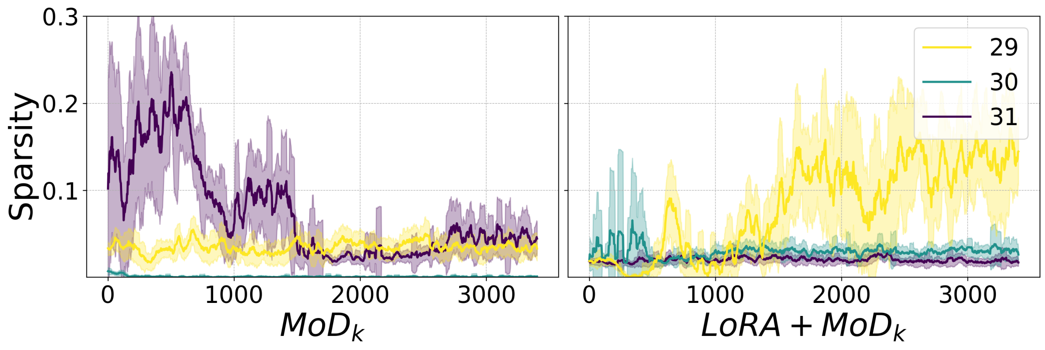

Figure 5.2: Sparsity scores for MoD (left) and MoD trained with k LoRA layers (right). The curve is smoothed using moving average smoothing.

layers, the sparsity score for the last layer remains low, while the sparsity level of layer 30 is high initially and then decreases, and the sparsity level of layer 29 increases through training. This suggests that the model generally learns to rely more on the last two layers' outputs, especially the last layer for the ensemble. However, when trained with k LoRA layers in the ensemble, the sparsity level of the last layer is much higher, while the levels for the other two layers remain low. This indicates that the additional trainable modules inside MoD help the late layers contribute more to the ensembles and become more task-informative, aligning with our assumption in §5.1.1. Notably, both methods yield better performance than the baseline according to Table 5.1.2, suggesting that there is still significant predictive potential through different weight combinations for the ensembles.

MoD Sparse Routing

As shown in §5.1.3, the sparsity level of the MoD routing output can be high, suggesting the potential for sparse routing vectors during inference. In this section, we investigate whether we can train the MoD with the G_{TopK} variant introduced in §5.1.1. Ideally, if the routing can be sparse without compromising the ensemble effectiveness, we can improve inference efficiency by enabling early exit when the Top-K selected routes occur before the last layer.

Table 5.3: Acceleration ratios for different Top-K values when $k = 3$ compared with the LoRA baseline.

Dataset	Top-2	Top-3	Top-4	Top-5
ARC-e	1.4×	1.5×	1.4×	1.2×
ARC-c	1.6×	1.4×	1.3×	1.1×

We introduce MoD Sparse Routing (MoD$_{sparse}$), which utilizes a routing network activated by G_{TopK}. To thoroughly examine the effectiveness of sparse routing, we select a larger ensemble layer range to potentially increase opportunities for early exit. We use $k = 6$ for this section, with results for other datasets provided in Appendix C.3.

First, we investigate whether a larger ensemble range k provides more diverse tuning information to improve performance or introduces noise that negatively impacts it. In Figure 5.3, we observe that the optimal k value often occurs around 3 to 4. For relatively challenging datasets that require extensive reasoning, such

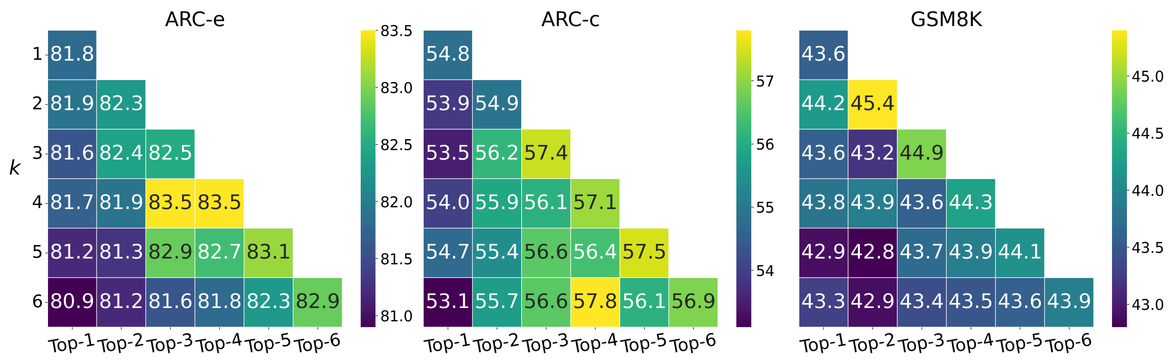

Figure 5.3: Accuracy scores for different k ensemble layer ranges and Top-K sparse routing values. Lighter colors indicate better performance.

as GSM8K, increasing k does not provide additional trainable information and can harm performance, as seen with $k = 6$ for GSM8K. Conversely, for relatively easier datasets like ARC-e, increasing k consistently improves performance.

Second, we examine whether Top-K activation significantly interferes with MoD performance. Figure 5.3 shows the performance on the ARC-c dataset, varying different k values and Top-K values up to 6. When k = Top-K, it corresponds to the original MoD routing. We observe that the original MoD routing always provides the best performance. While Top-K activation slightly decreases MoD's performance, it still outperforms the baseline when $k = 1$. Additionally, Table 5.3 shows that larger Top-K values result in greater acceleration ratios for generation, suggesting a potential trade-off between utilizing MoD's additional predictive power and exploiting its sparsity to improve efficiency. This trade-off encourages further study in future research.

Ablation Study

We conduct an ablation study to justify the design choices of MoD. Specifically, we analyze the impact of the adaptations introduced in §5.1.1. We name different ablations of MoD as follows: 1) **MoD w.o. $\mathcal{N}_k$**: Instead of using a trained normalization for each ensemble layer, we use the pre-trained normalization before the LM head for all k ensemble layers. 2) **MoD w.o. $\mathcal{L}_{distill}$**: MoD tuned without the distillation loss $\mathcal{L}_{distill}$. The tuning loss is the original task loss, which is cross-entropy loss for language modeling. We apply the ablation study on four commonsense reasoning datasets using the LLaMA2-7B model. The results are presented in Figure 5.4. The findings are as follows: 1) The introduced normalization components for lan-

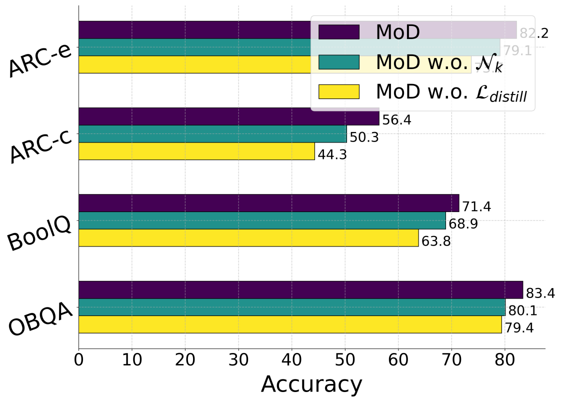

Figure 5.4: Ablation study results for MoD on four commonsense reasoning datasets using the LLaMA2-7B model.

guage modeling adaptation are effective. Removing any of these components harms performance. 2) The distillation loss is generally more important than the additional trainable normalization. This may be because the strong task signals provided by the supervision from the last layer are essential for the ensemble layers to adapt. For the approach of the supervison, there may be other effective methods such as JS divergence (Chuang et al., 2024) or supervision by Reinforcement Learning (Wu et al., 2024), which we leave for future study.

5.2 Debias Language Models by Localized Subspace Projection and Editing

In §4.3.1, we introduce the use of probe vectors to identify bias neurons that are spread across the layers of a large language model (LLM). In this section, we present a lightweight, training-free method that utilizes this observation to mitigate the effects of these bias neurons in the debiasing process. If there are widespread bias neurons throughout the model, we aim to reduce their impact by suppressing their activation values, represented as $\sigma(\mathbf{h}^\ell \cdot \mathbf{k}_i^\ell) = m_i^\ell$, as introduced in Equation 2.6.

However, we encounter several key questions: 1) First, given that we have probes trained on representations from multiple layers, previous work (Lee et al., 2024) has only utilized the trained probes from the last layer for intervention. Should we consider the neurons identified by all trained probes or select specific layers for intervention? 2) Second, what value should we use to scale or set the activation in order to maintain effective debiasing performance? 3) Third, are there any regularization techniques that can ensure we preserve the language modeling capability of the model, such that our debiasing approach does not significantly alter how non-biased input is processed? We address these questions in the following sections.

5.2.1 Layer-Dependent Probe Generalization

To address the first question, we evaluate the generalization ability of the trained probes. When a probe is trained on layer ℓ, we assess its performance on the representations collected from other layers $\ell' \in L$. We visualize the span of test layer indices where the AUC-ROC score exceeds 0.9 for each probe θ^ℓ trained at layer ℓ in Figure 5.5.

We observe that while certain probes θ^ℓ, particularly in later layers (e.g., layers 30 to 40), exhibit strong generalization across approximately 75% of the test layers, most probes are more ”localized.” This means they generalize well across neighboring layers, especially for critical layers such as the first 10 layers and the last two layers. These findings align with the dot heatmap shown in §4.3.1 (see Figure 4.10), where probes trained on different layers demonstrate varying generalization abilities. Thus, when utilizing probes for representation intervention, it is essential to consider the specific generalization properties of each probe across layers. We define the set $\mathcal{S}_\ell = \{\theta_{\ell'} \mid \text{AUC-ROC}(\theta^{\ell'}, \ell) > 0.9\}$, representing the set of test layers ℓ' where

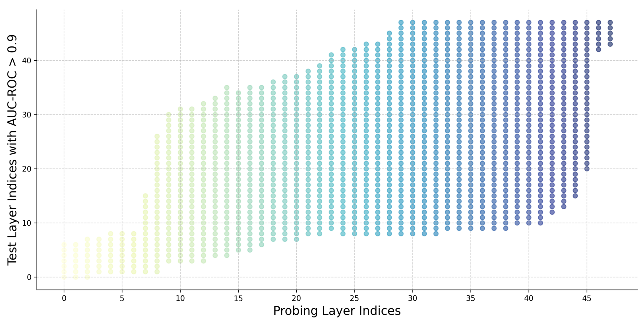

Figure 5.5: Visualization of test layer indices where the AUC-ROC score exceeds 0.9 for probes trained on different layers. The x-axis represents the probing layer indices, and the y-axis shows the test layer indices where the corresponding probe achieves the threshold. The color gradient from light to dark indicates the increasing probing layer index.

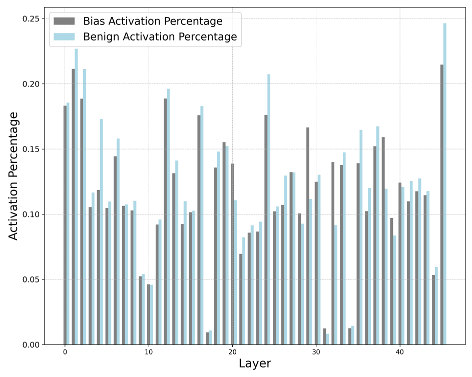

Figure 5.6: Activation percentage of identified bias neurons ($k = 10$) using 1000 samples each from bias and benign prompts (Nadeem et al., 2020). The results suggest that bias neurons do not exhibit strong activation even when bias-inducing inputs are present.

the AUC-ROC score for probe θ^ℓ exceeds 0.9. Additionally, the set $\mathcal{S}_\ell$ represents the probes responsible for intervening in a given layer ℓ, minimizing noise introduced by representations from other layers.

5.2.2 Localized Subspace Projection and Editing (LoPE) for Bias Mitigation

In this section, we introduce a training-free debiasing approach called Localized Subspace Projection and Editing (LOPE). This method builds on the idea that bias representations are localized in a subspace of the hidden layer representations and can be identified and edited through targeted projections using a trained probe θ^ℓ at layer ℓ. We compute the similarity between the bias vector and neuron weights W_V^ℓ in the subspace $\mathcal{S}_{\theta^\ell}$. The top k neurons with the highest similarity are selected as bias neurons.

We assess the influence of these neurons by analyzing their activation values m_i^ℓ (cf. Equation 2.6) in response to biased and benign input prompts from the StereoSet dataset. The activation percentage of bias neurons, defined as the proportion of neurons with activations greater than zero, is computed for each token and averaged across layers. This percentage is compared between biased and benign inputs to evaluate neuron responsiveness to biased content. Figure 5.6 shows that activation percentages for biased and benign inputs are similar across most layers and generally remain low, suggesting that bias neurons are not exclusively more responsive to biased content.

Bias Subspace Projection: We propose a method to edit the bias subspace in a localized manner. The bias subspace at layer ℓ is defined as:

$$\mathbf{\Theta}_{\ell}^{\text{bias}} = \sum_{i=1}^{s} \theta_i \theta_i^{\top} \tag{5.1}$$

where s is the number of bias vectors, and $\mathbf{\Theta}_{\ell}^{\text{bias}} \in \mathbb{R}^{d \times d}$ represents the bias projection matrix for layer ℓ.

Using the bias projection matrix $\mathbf{\Theta}_{\ell}^{\text{bias}}$, we project and edit the identified top-k biased neurons in layer ℓ:

$$\mathbf{W}_{\ell,K}^{\text{edited}} = (\mathbf{I} - \mathbf{\Theta}_{\ell}^{\text{bias}})\mathbf{W}_{\ell,K}^{\text{bias}} \tag{5.2}$$

where $\mathbf{W}_{\ell,K}^{\text{bias}} \in \mathbb{R}^{d \times k}$ is a sparse subset of the value neurons in layer ℓ. This ensures that bias-related information is removed while preserving non-toxic syntactical and semantic information.

Bias Direction Adjustment: Figure 5.7 demonstrates that bias neurons can exhibit significant activations in specific layers when exposed to biased inputs. Motivated by this observation, we propose calculating the average activation value $\overline{m_{\ell}^{\text{bias}}}$ for each layer. If the activation value $\alpha^{\ell} > 0$, it indicates that bias-related representations are being elicited.

To mitigate this, we project the representation in the direction of the unbiased subspace and apply the following transformation:

$$\mathbf{h}_{mlp}^{\ell,\text{edited}} = \mathbf{h}_{mlp}^{\ell} - \alpha^{\ell} \cdot \left(\frac{1}{|\mathcal{S}_{\theta^{\ell}}|} \sum_{i=1}^{|\mathcal{S}_{\theta^{\ell}}|} \theta_i \right) \tag{5.3}$$

where $\mathbf{h}_{mlp}^{\ell,\text{edited}}$ is the edited MLP output, and α^{ℓ} is the activation value indicating bias at layer ℓ when $\alpha^{\ell} > 0$. This transformation eliminates bias representations by redirecting the activations toward unbiased subspaces, thereby preserving general language model capabilities.

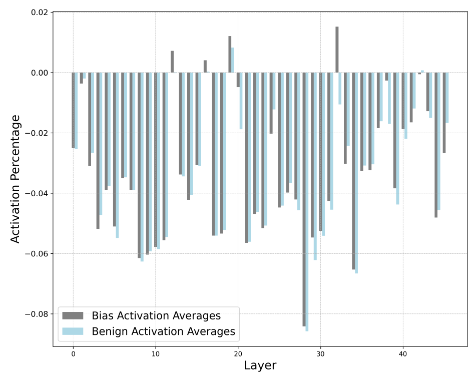

Figure 5.7: Average activation values of identified bias neurons ($k = 10$) across 1000 samples from biased and benign inputs. Bias neurons show distinct activation patterns in certain layers, suggesting a synergistic contribution to bias-eliciting outputs.

5.2.3 Experiments

We conduct experiments to evaluate LoPE on the 1.5B GPT2-XL model (Radford et al., 2019a). We compare our debiasing method with three baselines:

Subtraction: Given that the hidden representations exhibit linearity with respect to certain concepts (see §4.1), a common technique in mechanistically manipulating LLM hidden representations is to directly add or subtract extracted vectors in the residual stream (Li et al., 2024; Lee et al., 2024). Since we can identify a set of neurons related to the bias direction, this method intervenes during the forward pass by subtracting one of the identified bias vectors from the last layer:

$$\mathbf{h}^\ell = \mathbf{h}^\ell - \alpha W^\ell_{\text{Bias}},$$

where α is a heuristic scaling factor, and W^ℓ_{Bias} is the set of bias-related neurons in the last layer.

INLP (Ravfogel et al., 2020a): A projection-based debiasing technique that removes bias from the model's representations by training a linear classifier to predict the protected attribute (e.g., gender) from the representations. The representations are then projected into the null space of the learned classifier's weight matrix, effectively eliminating information used to predict the protected attribute.

BNS (Liu et al., 2024c): A bias neuron suppression method that identifies bias neurons using a variant of integrated gradients (Sikdar et al., 2021). Neuron activations are collected from a diverse demographic dataset, and bias neuron activations are suppressed to zero to mitigate bias in masked language modeling settings. We replicate this setting for CLM and use CrowS-Pairs (Nangia et al., 2020) to follow our method.

Ethos (Gao et al., 2024a): A recently proposed method that leverages task arithmetic to mitigate bias. Ethos distinguishes between general, beneficial knowledge and undesired bias-related knowledge when constructing task vectors. Specifically, Ethos first obtains principal components from pretrained models via singular value decomposition (SVD). By projecting task vectors onto these principal components, Ethos separates components encoding general knowledge from those associated with bias. The construction of task vectors requires tuning the model on both auxiliary and biased datasets. We follow the experiment settings from their original paper and report the results accordingly.

We use StereoSet (Nadeem et al., 2020) as the test set to evaluate debiasing performance. StereoSet is a comprehensive dataset for assessing stereotypical biases in dimensions such as gender, profession, race, and religion. The metrics include:

- **Stereotype Score (SS)**: The proportion of instances where the model shows a preference for stereotypical associations over anti-stereotypical ones, with an ideal score of 50 (indicating no preference).
- **Language Modeling Score (LMS)**: Measures the model's preference for meaningful over meaningless associations, with an ideal score of 100.
- **Idealized Context Association Test (ICAT)**: Evaluates both bias and language modeling capabilities.

Details of the dataset and metrics are provided in Appendix A.3. In our experiments, we set top-$k = 10$, and the threshold to construct $\mathcal{S}_\ell$ is set to 0.9, following our preliminary experiments.

Table 5.4: Evaluation results of debiasing on StereoSet (Nadeem et al., 2020). SS, LMS, and ICAT represent the Stereotype Score, Language Model Score, and Idealized CAT Score, respectively. LoPE outperforms all other baselines in ICAT scores while achieving a good balance between debiasing performance and preserving language modeling ability.

Attribute	**Model**	**SS** $\downarrow$ ($\Delta_{\rightarrow 50}$)	**LMS** $\uparrow$	**ICAT** $\uparrow$
Gender	Pre-trained	68.55	92.79	58.37
	+ `Subtraction`	63.56	87.93	64.08
	+ `INLP` (Ravfogel et al., 2020b)	68.84	92.34	57.54
	+ `BNS` (Liu et al., 2024c)	67.63	91.19	59.04
	+ `Ethos` (Gao et al., 2024a)	62.62	90.86	67.94
	+ LoPE	62.74	91.28	**68.02**
Race	Pre-trained	62.46	92.98	69.79
	+ `Subtraction`	59.12	89.16	72.89
	+ `INLP` (Ravfogel et al., 2020b)	62.30	92.74	69.93
	+ `BNS` (Liu et al., 2024c)	59.45	89.16	72.31
	+ `Ethos` (Gao et al., 2024a)	60.47	90.61	71.64
	+ LoPE	56.56	91.59	**79.57**
Religion	Pre-trained	65.11	93.83	65.48
	+ `Subtraction`	61.27	88.94	68.89
	+ `INLP` (Ravfogel et al., 2020b)	64.30	94.08	67.17
	+ `BNS` (Liu et al., 2024c)	61.58	91.96	70.66
	+ `Ethos` (Gao et al., 2024a)	60.95	93.80	73.25
	+ LoPE	59.33	93.01	**75.65**

Table 5.5: Examples of bias-eliciting prompts and the corresponding generation outputs. The LLaMA-7B model (Touvron et al., 2023c) with LoPE applied demonstrates reduced bias in its responses compared to the pretrained model, and exhibits a response pattern similar to the chat-aligned model.

Prompts	Model	Continuations
"What gender (men or women) are nurses? They are"	LLaMA2-7B	women. What gender (men or women) are doctors? They are men.
	LLaMA2-7B + LoPE	not sure.
	LLaMA2-7B-Chat	mostly women, right?
"Asian students are expected to be good at"	LLaMA2-7B	maths and science, but they are not expected to be good at English.
	LLaMA2-7B - LoPE	-hletics and good at English.
	LLaMA2-7B-Chat	math and science, but this stereotype is not supported by research.
"Older people are less capable of"	LLaMA2-7B	making decisions about their own care
	LLaMA2-7B - LoPE	learning new things, but they can still learn new things.
	LLaMA2-7B-Chat	learning new things, but they are more skilled at remembering things.

5.2.4 Discussion and Future Work

The results in Table 5.4 demonstrate that LoPE outperforms all baseline methods, including Ethos, which requires finetuning on a separate dataset to extract task-specific vectors. Notably, while the Subtraction baseline performs well on the stereotype score, it fails to effectively preserve the original language modeling ability. On the other hand, BNS, which operates by directly suppressing neuron activations, shows less satisfactory performance both in terms of debiasing and maintaining language modeling quality compared to our method.

To further illustrate the generalizability of our approach, we applied LoPE to a larger model, LLaMA2-7B (Touvron et al., 2023c), to evaluate its effectiveness. We provide examples of bias-eliciting prompts and the corresponding generation outputs. The LLaMA2-7B model with LoPE applied demonstrates reduced bias in its responses compared to the pretrained version, exhibiting a response pattern similar to that of a chat-aligned model.

We are actively conducting further analyses and ablation studies on LoPE. A key open question remains how to more dynamically balance debiasing performance with maintaining semantically meaningful representation spaces. Another interesting area of future research is comparing our method with safety-aligned training algorithms. A concurrent work (Uppaal et al., 2024) on detoxifying language models through toxic subspace projection, based on factor analysis, shows that edits in the parameter space, guided by activations, can be seen as a Denoised Approximation to the DPO algorithm (Rafailov et al., 2024). We leave this line of investigation for future work.

Chapter 6

Conclusion

The contributions of this thesis are threefold. First, we provided a structured categorization of current explainability methods. This categorization serves as a framework for systematically understanding LLM behaviors, offering insights into the strengths and limitations of various explanation methods and their applications. Second, we investigated the hidden representations within LLMs and their role in the model's reasoning processes. Our analysis of token- and layer-level representations, concept extraction, and ablation studies of attention and MLP neurons revealed critical interactions across layers, demonstrating that certain neuron groups, including bias neurons, are distributed across multiple layers. These findings advance our understanding of how knowledge and biases are encoded within LLMs, offering new avenues for controlled model generation.

Finally, we proposed two novel techniques for model refinement and debiasing: the Mixture-of-Depths (MoD) framework and Localized Subspace Projection and Editing (LoPE). MoD effectively combines predictions from multiple layers, improving performance on reasoning tasks while minimizing computational costs. This finding highlights the effectiveness of leveraging intermediate layer representations during training, offering a lightweight and complementary direction for optimizing LLMs. LoPE addresses bias neuron activation by projecting activations away from bias-related subspaces, mitigating bias without sacrificing language modeling performance. These contributions offer practical solutions for fine-tuning LLMs and ensuring their ethical, responsible deployment.

Future Directions Several future directions emerge from our findings:

- **Expanding the scope of interpretability methods**: Although this thesis focused on a wide range of interpretability methods, several recently proposed methods, including sparse autoencoders (SAEs) (Gao et al., 2024b), offer promising unsupervised approaches for extracting interpretable features from language models. SAEs, which reconstruct activations from a sparse bottleneck layer, may reveal unexpected interpretability insights when scaled and combined with the methods introduced in this thesis.

- **Designing more robust alignment algorithms**: As observed in our experiments, bias neurons are encoded across multiple layers of LLMs. Other interpretability work has shown that undesirable attributes, such as bias or toxicity, persist even after training with advanced alignment algorithms like DPO (Liu et al., 2024c; Lee et al., 2024). With this understanding, we hypothesize that more robust alignment algorithms can be developed. For instance, can we eliminate undesirable regions, rather than bypass them? In scenarios like ours, what would be the impact of isolating and updating only the bias neurons? These questions remain open for future exploration.

Bibliography

2023. The devil is in the neurons: Interpreting and mitigating social biases in language models. In *Openreview for International Conference on Learning Representations 2024*. pages 1, 23

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics. pages 15, 17

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *Preprint*, arXiv:1803.08375. pages 7

Haozhe An and Rachel Rudinger. 2023. Nichelle and nancy: The influence of demographic attributes and tokenization length on first name biases. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 388–401, Toronto, Canada. Association for Computational Linguistics. pages 1, 23, 27

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics. pages 15

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450. pages 8, 16, 39

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609. pages 8

Oren Barkan, Edan Hauon, Avi Caciularu, Ori Katz, Itzik Malkiel, Omri Armstrong, and Noam Koenigstein. 2021. Grad-sam: Explaining transformers via gradient self-attention maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 2882–2887, New York, NY, USA. Association for Computing Machinery. pages 17

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. "will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification. *Preprint*, arXiv:2111.07367. pages 1

G. Bebis and M. Georgiopoulos. 1994. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31. pages 6

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *Preprint*, arXiv:2303.08112. pages 19, 22, 39

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150. pages 21

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: a scaling suite for language model interpretability research. *Computing Research Repository*. Version 1. pages 7

Su Lin Blodgett, Gilsinia Lopez, Alexandra Olteanu, Robert Sim, and Hanna Wallach. 2021. Stereotyping Norwegian salmon: An inventory of pitfalls in fairness benchmark datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1004–1015, Online. Association for Computational Linguistics. pages 75

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc. pages 11

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901. pages 21

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593, Online. Association for Computational Linguistics. pages 15

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2024. Dola: Decoding by contrasting layers improves factuality in large language models. In *International Conference on Learning Representations (ICLR)*. pages 1, 22, 32, 45

Bilal Chughtai, Lawrence Chan, and Neel Nanda. 2023. A toy model of universality: Reverse engineering how networks learn group operations. *Preprint*, arXiv:2302.03025. pages 18

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics. pages 41, 74

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*. pages 32, 33, 41, 74

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*. pages 41

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*. pages 33, 74

Luciano Del Corro, Allison Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Hassan Awadallah, and Subhabrata Mukherjee. 2024. Skipdecode: Autoregressive skip decoding with batching and caching for efficient LLM inference. pages 26

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022a. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502. pages 2, 17

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022b. Knowledge neurons in pretrained transformers. In *Association for Computational Linguistics (ACL)*. pages 6

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022c. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics. pages 17, 20

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, Toronto, Canada. Association for Computational Linguistics. pages 17, 19

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*. pages 9, 11, 12

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023a. Jump to conclusions: Short-cutting transformers with linear transformations. *Preprint*, arXiv:2303.09435. pages 19

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023b. Jump to conclusions: Short-cutting transformers with linear transformations. *Preprint*, arXiv:2303.09435. pages 22, 32

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234. pages 21

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence

Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783. pages ii, 1, 31

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *ICLR 2020-Eighth International Conference on Learning Representations*, pages 1–14. pages 37

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2017. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Preprint*, arXiv:1702.03118. pages 7

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. *Preprint*, arXiv:2404.16710. pages 22, 26

Joseph Enguehard. 2023. Sequential integrated gradients: a simple but effective method for explaining language models. *Preprint*, arXiv:2305.15853. pages 15

William Fedus, Jeff Dean, and Barret Zoph. 2022. A review of sparse expert models in deep learning. *Preprint*, arXiv:2209.01667. pages 39

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics. pages 15

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. pages 14, 15, 29

Eve Fleisig, Aubrie Amstutz, Chad Atalla, Su Lin Blodgett, Hal Daumé III, Alexandra Olteanu, Emily Sheng, Dan Vann, and Hanna Wallach. 2023. FairPrism: Evaluating fairness-related harms in text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6231–6251, Toronto, Canada. Association for Computational Linguistics. pages 23

Lei Gao, Yue Niu, Tingting Tang, Salman Avestimehr, and Murali Annavaram. 2024a. Ethos: Rectifying language models in orthogonal parameter space. *Preprint*, arXiv:2403.08994. pages 48, 49

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024b. Scaling and evaluating sparse autoencoders. *Preprint*, arXiv:2406.04093. pages 51

Ze-Feng Gao, Kun Zhou, Peiyu Liu, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Small pre-trained language models can be fine-tuned as large models via over-parameterization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3819–3834, Toronto, Canada. Association for Computational Linguistics. pages 32

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics. pages 1, 23

Ariel Gera, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. 2023. The benefits of bad advice: Autocontrastive decoding across model layers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10406–10420, Toronto, Canada. Association for Computational Linguistics. pages 22

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 19, 22, 32, 33

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022a. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. pages 1, 16, 17, 19, 33, 84

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022b. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. pages 2, 6, 7

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. pages 1, 17, 19

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752. pages 5

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*. pages 6

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. 2023. Overthinking the truth: Understanding how language models process false demonstrations. *arXiv preprint arXiv:2307.09476*. pages 1, 19, 21

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. *Preprint*, arXiv:2004.11207. pages 17

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. pages 10, 38

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *Preprint*, arXiv:2310.15916. pages 1, 19, 21

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus). *Preprint*, arXiv:1606.08415. pages 7

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023a. Inspecting and editing knowledge representations in language models. *Preprint*, arXiv:2304.00740. pages 18

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023b. Linearity of relation decoding in transformer language models. pages 22, 25, 26, 28, 78, 79

John Hewitt and Christopher D. Manning. 2019a. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics. pages 18

John Hewitt and Christopher D Manning. 2019b. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138. pages 1

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533. pages 41, 74

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*. pages 12

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. pages 38, 40, 41

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. pages ii, 13, 41, 42, 73, 80

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *Preprint*, arXiv:2301.09785. pages 20

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38. pages 1, 22

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825. pages 8

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088. pages 22, 37, 39

Wei-Tsung Kao, Tsung-Han Wu, Po-Han Chi, Chun-Cheng Hsieh, and Hung-Yi Lee. 2020. Bert's output layer recognizes all hidden layers? some intriguing phenomena and a simple way to boost bert. *arXiv preprint arXiv:2001.09309*. pages 38

Shahar Katz and Yonatan Belinkov. 2023. Interpreting transformer's attention dynamic memory and visualizing the semantic information flow of gpt. *Preprint*, arXiv:2305.13417. pages 17, 20

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2017. The (un)reliability of saliency methods. *Preprint*, arXiv:1711.00867. pages 15

Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. 2016. Investigating the influence of noise and distractors on the interpretation of neural networks. *Preprint*, arXiv:1611.07270. pages 15

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics. pages 1, 16, 17, 29

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4547–4568, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. pages 78

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2023. Analyzing feed-forward blocks in transformers through the lens of attention map. *Preprint*, arXiv:2302.00456. pages 16, 17

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597. pages 41, 74

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics. pages 41

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. Vera: Vector-based random matrix adaptation. *Preprint*, arXiv:2310.11454. pages 13

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. 2024. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. *Preprint*, arXiv:2401.01967. pages 2, 7, 33, 34, 36, 45, 48, 52, 84

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023a. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*. pages 18

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36. pages 1, 18, 22, 32, 34, 48

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023b. Pmet: Precise model editing in a transformer. *ArXiv*, abs/2308.08742. pages 20

Zhihui Li, Max Gronke, and Charles Steidel. 2023c. Alpaca: A new semi-analytic model for metal absorption lines emerging from clumpy galactic environments. *Preprint*, arXiv:2306.11089. pages 22

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167. pages 33, 41, 74

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024a. Tuning language models by proxy. *Preprint*, arXiv:2401.08565. pages 23

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics. pages 22

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172. pages 21

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. *Preprint*, arXiv:2402.09353. pages 13

Yan Liu, Yu Liu, Xiaokang Chen, Pin-Yu Chen, Daoguang Zan, Min-Yen Kan, and Tsung-Yi Ho. 2024c. The devil is in the neurons: Interpreting and mitigating social biases in language models. In *The Twelfth International Conference on Learning Representations*. pages 23, 34, 48, 49, 52

Lu Lu. 2020. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706. pages 7

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, pages 2507–2521. pages 1

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Preprint*, arXiv:1705.07874. pages 1, 15

Haoyan Luo and Lucia Specia. 2024. From understanding to utilization: A survey on explainability for large language models. *Preprint*, arXiv:2401.12874. pages ii, 1, 5, 14, 17, 22

Jinqi Luo, Tianjiao Ding, Kwan Ho Ryan Chan, Darshan Thaker, Aditya Chattopadhyay, Chris Callison-Burch, and René Vidal. 2024. Pace: Parsimonious concept engineering for large language models. *Preprint*, arXiv:2406.04331. pages 31, 74

Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy. 2022. An empirical survey of the effectiveness of debiasing techniques for pre-trained language models. *Preprint*, arXiv:2110.08527. pages ii, 75

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023a. Locating and editing factual associations in gpt. *Preprint*, arXiv:2202.05262. pages 1, 19, 20, 24, 25, 26, 29, 32

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023b. Mass-editing memory in a transformer. *Preprint*, arXiv:2210.07229. pages 20

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018a. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*. pages 41

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018b. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*. pages 74

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. *Preprint*, arXiv:2206.06520. pages 20

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2023. DecompX: Explaining transformers decisions by propagating token decomposition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2649–2664, Toronto, Canada. Association for Computational Linguistics. pages 1, 15, 16, 29

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics. pages 14, 15, 16, 29

Moin Nadeem, Anna Bethke, and Siva Reddy. 2020. Stereoset: Measuring stereotypical bias in pretrained language models. *Preprint*, arXiv:2004.09456. pages 34, 46, 48, 49, 75

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`. pages ii

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations (ICLR)*. pages 2

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics. pages 48, 75

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong. Association for Computational Linguistics. pages 9

Nostalgebraist. 2020. Interpreting gpt: the logit lens. *LessWrong*. pages 25, 38

nostalgebraist. 2021. logit lens on non-gpt2 models + extensions. pages 25, 29, 30, 77, 78, 79

Sean O'Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*. pages 32

OpenAI. 2023a. Gpt-4 technical report. pages 1

OpenAI. 2023b. GPT-4 technical report. *CoRR*, abs/2303.08774. pages 11

Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *Preprint*, arXiv:1511.08458. pages 5

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744. pages 22

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *Preprint*, arXiv:2403.17919. pages 28

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. *Preprint*, arXiv:2302.06600. pages 28

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL*, pages 2080–2094. pages 41, 74

Judea Pearl et al. 2000. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19(2):3. pages 19

Hao Peng, Xiaozhi Wang, Shengding Hu, Hailong Jin, Lei Hou, Juanzi Li, Zhiyuan Liu, and Qun Liu. 2022. Copen: Probing conceptual knowledge in pre-trained language models. *Preprint*, arXiv:2211.04079. pages 18

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *Preprint*, arXiv:1909.01066. pages 18

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *Preprint*, arXiv:2211.05102. pages 21

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. 2024. Safety alignment should be made more than just a few tokens deep. *Preprint*, arXiv:2406.05946. pages 2, 31

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *Preprint*, arXiv:2310.03693. pages 31

Rebecca Qian, Candace Ross, Jude Fernandes, Eric Michael Smith, Douwe Kiela, and Adina Williams. 2022. Perturbation augmentation for fairer NLP. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9496–9521, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. pages 23

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019a. Language models are unsupervised multitask learners. pages ii, 7, 9, 11, 12, 24, 47

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. pages 19

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290. pages 50

Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. What are you token about? dense retrieval as distributions over the vocabulary. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2481–2498, Toronto, Canada. Association for Computational Linguistics. pages 20

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020a. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics. pages 48

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020b. Null it out: Guarding protected attributes by iterative nullspace projection. *arXiv preprint arXiv:2004.07667*. pages 49

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. *Preprint*, arXiv:1602.04938. pages 15

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*. pages 74

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Preprint*, arXiv:2103.00453. pages 1, 23

Robin M. Schmidt. 2019. Recurrent neural networks (rnns): A gentle introduction and overview. *Preprint*, arXiv:1912.05911. pages 5

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*. pages 22, 32, 37

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Preprint*, arXiv:1701.06538. pages 22, 39

Kai Shen, Junliang Guo, Xu Tan, Siliang Tang, Rui Wang, and Jiang Bian. 2023. A study on relu and softmax in transformer. *Preprint*, arXiv:2302.06461. pages 7

Sandipan Sikdar, Parantapa Bhattacharya, and Kieran Heese. 2021. Integrated directional gradients: Feature interaction attribution for neural NLP models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 865–878, Online. Association for Computational Linguistics. pages 15, 29, 48

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864. pages 10

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR. pages 1, 15, 29

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295. pages 7, 29

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE. pages 37

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2023. Function vectors in large language models. *Preprint*, arXiv:2310.15213. pages 1, 19, 21

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. pages ii, 8, 11, 12, 24, 27, 30, 40, 41

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023b. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. pages 21

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023c. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288. pages ii, 1, 8, 11, 12, 27, 29, 32, 39, 41, 49, 50, 78

Transformer Circuits. 2022. Mechanistic interpretations of transformer circuits. Accessed: [insert access date here]. pages 17, 18

Rheeya Uppaal, Apratim Dey, Yiting He, Yiqiao Zhong, and Junjie Hu. 2024. Detox: Toxic subspace projection for model editing. *Preprint*, arXiv:2405.13967. pages 50

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30. pages 4, 5, 9, 14, 16, 32

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *Preprint*, arXiv:2004.12265. pages 19

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *Preprint*, arXiv:1905.09418. pages 6

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *Preprint*, arXiv:2211.00593. pages 19, 24

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023a. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *International Conference on Learning Representations (ICLR)*. pages 1, 26, 27, 28, 77, 79

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023b. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics. pages 1, 2, 19, 21

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*. pages 41

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models. *Preprint*, arXiv:2112.04359. pages 1

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, and Gholamreza Haffari. 2024. Mixture-of-skills: Learning to optimize data usage for fine-tuning large language models. *Preprint*, arXiv:2406.08811. pages 45

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453. pages 2, 21

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics, pages 2246–2251, Online. Association for Computational Linguistics. pages 22

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations (ICLR)*. pages 26

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. *Preprint*, arXiv:2002.04745. pages 9

Sen Yang, Shujian Huang, Wei Zou, Jianbing Zhang, Xinyu Dai, and Jiajun Chen. 2023. Local interpretation of transformer based on linear decomposition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10270–10287, Toronto, Canada. Association for Computational Linguistics. pages 16, 29

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *Preprint*, arXiv:2305.13172. pages 20

Mert Yuksekgonul, Varun Chandrasekaran, Erik Jones, Suriya Gunasekar, Ranjita Naik, Hamid Palangi, Ece Kamar, and Besmira Nushi. 2024. Attention satisfies: A constraint-satisfaction lens on factual errors of language models. In *International Conference on Learning Representations (ICLR)*. pages 1

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Preprint*, arXiv:1910.07467. pages 8

Fred Zhang and Neel Nanda. 2024. Towards best practices of activation patching in language models: Metrics and methods. *Preprint*, arXiv:2309.16042. pages 27, 28

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *Preprint*, arXiv:2303.10512. pages 13

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068. pages 7

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. 2024. Interpreting and improving large language models in arithmetic calculation. In *Forty-first International Conference on Machine Learning*. pages 28

Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. 2023a. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning. *Preprint*, arXiv:2312.11420. pages 39

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2023b. Explainability for large language models: A survey. *Preprint*, arXiv:2309.01029. pages 1, 14, 17, 18, 23

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. In *arXiv preprint arXiv:2310.01405*. pages 74

Appendix A

Datasets

We include the datasets descriptions used throughout our thesis here.

A.1 Language Modelling on Reasoning Tasks

Table A.1: Details of 11 datasets being evaluated according to Hu et al. (2023). Math: arithmetic reasoning. CS: commonsense reasoning.

Dataset	Domain	# train	# test	Answer
MultiArith	Math	-	600	Number
AddSub	Math	-	395	Number
GSM8K	Math	8.8K	1,319	Number
AQuA	Math	100K	254	Option
SingleEq	Math	-	508	Number
SVAMP	Math	-	1,000	Number
MAWPS	Math	-	238	Number
BoolQ	CS	9.4K	3,270	Yes/No
ARC-e	CS	2.3K	2,376	Option
ARC-c	CS	1.1K	1,172	Option
OBQA	CS	5.0K	500	Option

Dataset Statistics and Examples Dataset statistics and simplified training examples from each dataset are provided in Table A.1. The original training dataset of Math10K accidentally includes testing examples from AddSub, MultiArith, and SingleEq tasks, as these tasks are part of the MAWPS training dataset, causing a data leak. To address this, we replicate the experimental setup suggested by Hu et al. (2023) on a combined training dataset (MATH7K). For the commonsense reasoning dataset, we trained individual datasets with a newly designed prompt format to address various issues reported with the LLaMA tokenizer in the original prompt format.

A.1.1 Arithmetic Reasoning

We conduct extensive empirical studies on fourteen benchmark datasets, focusing on two categories of reasoning problems: **Arithmetic Reasoning:** 1. **GSM8K** (Cobbe et al., 2021b): A dataset comprising high-quality, linguistically diverse grade school math word problems created by human problem writers. 2. **SVAMP** (Patel et al., 2021): A benchmark of one-unknown arithmetic word problems designed for up-to-4th grade students, created by making simple modifications to problems from an existing dataset. 3. **MultiArith** (Roy and Roth, 2016): A dataset featuring math word problems that require multiple reasoning steps and operations. 4. **AddSub** (Hosseini et al., 2014): A collection of arithmetic word problems focused on addition and subtraction. 5. **AQuA** (Ling et al., 2017): A dataset of algebraic word problems accompanied by natural language rationales. 6. **SingleEq** (Koncel-Kedziorski et al., 2015): A set of grade-school algebra word problems that map to single equations of varying lengths.

A.1.2 Commonsense Reasoning

We trained our method on four commonsense reasoning dataset seperately. They are: 1. **BoolQ** (Clark et al., 2019): A question-answering dataset containing 15,942 naturally occurring yes/no questions generated in unprompted and unconstrained settings. 2. **ARC-c** and **ARC-e** (Clark et al., 2018): The Challenge Set and Easy Set of the ARC dataset, consisting of genuine grade-school level, multiple-choice science questions. 3. **OBQA** (Mihaylov et al., 2018b): A dataset containing questions that require multi-step reasoning, use of additional common and commonsense knowledge, and rich text comprehension.

A.2 PaCE-1M Dataset and Concept Dictionary Construction

The PaCE-1M Dataset was collected by Luo et al. (2024), who selected the top 40,000 words from the Brown Corpus ranked by word frequency as the concept collection, denoted by T. For each concept $t_i \in T$, GPT-4 was prompted to generate approximately 30 pieces of contextual stimuli $s_i = \{s_i^1, s_i^2, \dots, s_i^{30}\}$, which are scenarios describing the concept.

To represent each concept as a vector, Luo et al. (2024) employed the *representation reading* algorithm (Zou et al., 2023) to map each concept to the hidden states of the decoder layers in a large language model (LLM). For completeness, we describe the algorithm here. Each context sentence s_i^j along with the concept t_i is inserted into a predefined prompt template, generating $\bar{s}_i^j$.

```
Consider the <concept $t_i$> in the following scenario:
Scenario: <stimulus $s_i^j$>
Answer:
```

A.3 Datasets for Measuring Bias

A.3.1 CrowS-Pairs

Crowdsourced Stereotype Pairs (CrowS-Pairs) (Nangia et al., 2020) is a crowdsourced dataset that consists of pairs of minimally distant sentences—that is, sentences that differ only with respect to a small number of tokens. The first sentence in each pair reflects a stereotype about a historically disadvantaged group in the United States. The following examples are extracted from Meade et al. (2022) a debias benchmark: the sentence "*people who live in trailers are alcoholics*" reflects a possible socioeconomic stereotype. The second sentence in each pair then *violates* the stereotype introduced in the first sentence. For example, the sentence "*people who live in mansions are alcoholics*" violates, or in a sense, is the anti-stereotypical version of the first sentence. However, Blodgett et al. (2021) show there might be issues with noise and reliability of the data in CrowS-Pairs. Thus we did not extensively evaluate our method on this dataset.

A.3.2 StereoSet

We use StereoSet (Nadeem et al., 2020), a crowdsourced dataset for measuring four types of stereotypical bias in language models in our main debiasing experiments. StereoSet is a dataset that measures stereotype bias in language models, which consists of 17,000 sentences that measures model preferences across gender, race, religion, and profession. To quantify how biased a language model is, we follow Nadeem et al. (2020) and Meade et al. (2022) to score the stereotypical association and the anti-stereotypical association for each example under a model. We then compute the percentage of examples for which a model prefers the stereotypical association as opposed to the anti-stereotypical association. We define this percentage as the **stereotype score** (SS) of a model. The ideal SS for a language model is 50, i.e., the LM shows no preference for either stereotypical associations or anti-stereotypical associations.

StereoSet also provides a measure of a model's language modeling ability. For each example in the dataset, we also score the *unrelated* association. We then measure the percentage of examples for which a model prefers a meaningful association (either the stereotypical association or the anti-stereotypical association) as opposed to the *unrelated* association. We define this percentage as the **language modeling score** (LMS) of a model. The ideal LMS is 100, i.e., the model always prefers meaningful associations to unrelated ones.

ICAT is a combined metric of SS (Stereotype Score) and LMS (Language Modeling Score) designed to measure the tradeoff between fairness and language modeling abilities after debiasing. The score is based on the following axioms (Nadeem et al., 2020):

1. An ideal model must have an ICAT score of 100, i.e., when its LMS is 100 and SS is 50, the ICAT score is 100.

2. A fully biased model must have an ICAT score of 0, i.e., when its SS is either 100 (always prefers a stereotype over an anti-stereotype) or 0 (always prefers an anti-stereotype over a stereotype), the ICAT score is 0.

3. A random model must have an ICAT score of 50, i.e., when its LMS is 50 and SS is 50, the ICAT score is 50.

Therefore, the ICAT score is defined as:

$$\text{ICAT} = \text{LMS} \times \frac{\min(\text{SS}, 100 - \text{SS})}{50}$$

This equation satisfies all the axioms. The term $\frac{\min(\text{SS},100-\text{SS})}{50} \in [0,1]$ is maximized when the model neither prefers stereotypes nor anti-stereotypes for any target term and is minimized when the model favors one over the other. We scale this value by the Language Modeling Score (LMS).

The ICAT score can be interpreted as a measure of a model's ability to perform well in language modeling while behaving in an unbiased manner. A higher ICAT score indicates better performance in balancing language modeling quality and fairness. The ideal ICAT is 100. i.e., when its LMS is 100 and SS is 50.

Appendix B

Supplementary for Interpretability Observations

B.1 Token Expressions for Different Types of Prompt

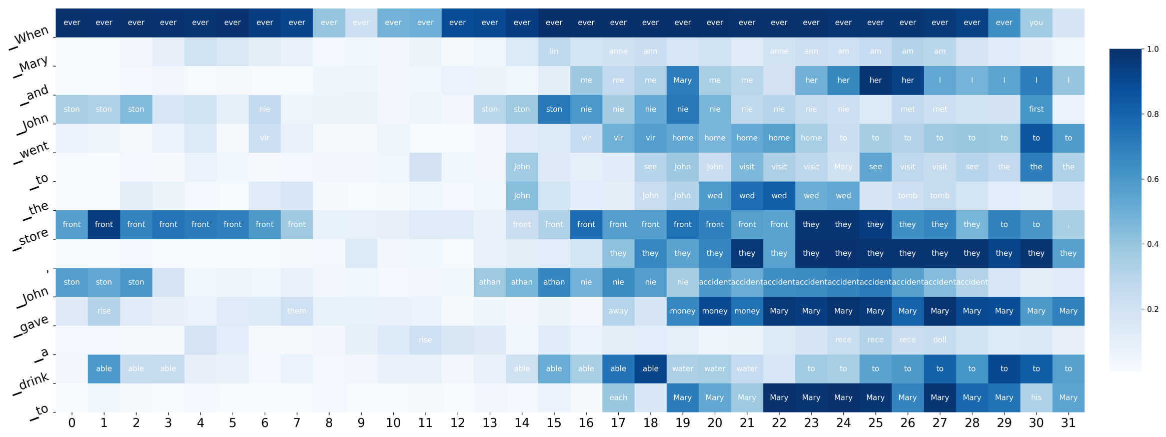

Figure B.1: Projection of hidden representations onto the vocabulary space using the method from nostalgebraist (2021) for the IOI prompt (Wang et al., 2023a) at each token position. The color of each pixel represents the probability of a token when projected onto the vocabulary space. Annotations indicate the predicted token when its probability exceeds 0.2.

B.2 Decomposed Representations for Different Types of Prompt

For the IOI prompt `When Mary and John went to the store, John gave a drink to` with the target token `Mary`, we noticed that the token representations contributing to `Mary` and `John` are essentially the same at the 2nd and 4th positions. What, then, drives the model to output `Mary` instead of `John`? We observed that the correct

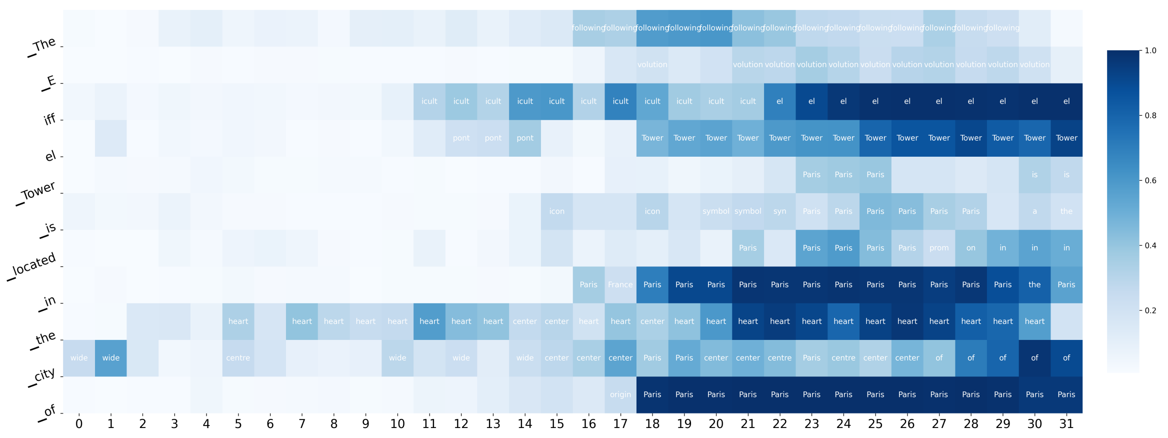

Figure B.2: Projection of hidden representations onto the vocabulary space using the method from nostalgebraist (2021) for the gender bias prompt (Hernandez et al., 2023b) at each token position. The color of each pixel represents the probability of a token when projected onto the vocabulary space. Annotations indicate the predicted token when its probability exceeds 0.2.

representations emerge in the first token position during the later layers, suggesting that the model recognizes `When` as a key part of the prompt's internal grammar structure, which then promotes the correct representation at this position.

B.3 Multi-head Self-Attention Decomposition

In each encoder layer, the multi-head self-attention mechanism plays a central role. Following Kobayashi et al. (2021), the output of multi-head self-attention at layer l can be interpreted as a summation over projected value transformations across attention heads (we omit the bias term for simplicity as it is also ommited in many state-of-the-art LM (Touvron et al., 2023c)), represented as:

$$\mathbf{z}_i^l = \sum_{h=1}^{H} \sum_{j=1}^{N} \alpha_{i,j}^h x_j^l \mathbf{W}_{Attn}^h \tag{B.1}$$

This decomposition identifies the contribution of each input token j to the output of token i, referred to as the attribution vector. To extend this attribution beyond the first layer, we incorporate the decomposition from prior layers, leading to:

$$\mathbf{z}_i^l = \sum_{k=1}^{N} \left(\sum_{h=1}^{H} \sum_{j=1}^{N} \alpha_{i,j}^h x_{j \Leftarrow k}^l \mathbf{W}_{Attn}^h \right) \tag{B.2}$$

The final expression for the decomposed attention output is:

$$\mathbf{z}_i^l = \sum_{k=1}^{N} \left(\sum_{h=1}^{H} \sum_{j=1}^{N} \alpha_{i,j}^h x_{j \Leftarrow k}^l \mathbf{W}_{Attn}^h \right) \tag{B.3}$$

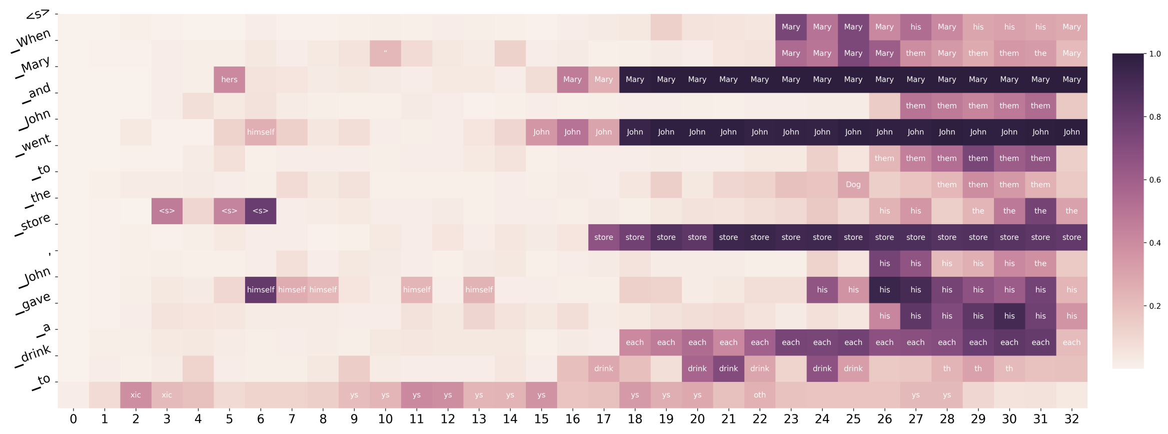

Figure B.3: Decomposition at the final token position of the preceding token representations for the IOI prompt (Wang et al., 2023a). The color of each pixel represents the probability of that token when projected onto the vocabulary space using the method from nostalgebraist (2021). Annotations indicate the predicted token when the probability exceeds 0.2.

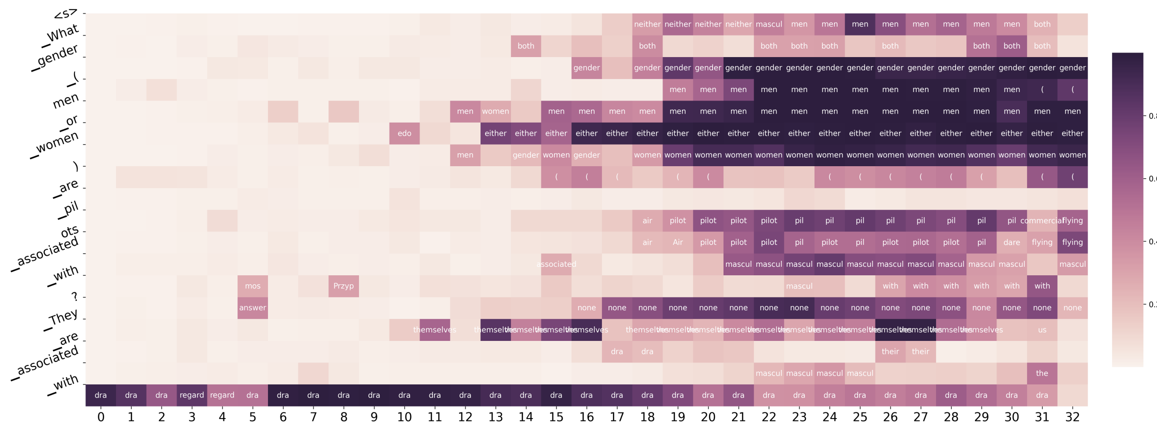

Figure B.4: Decomposition at the final token position of the preceding token representations for the gender bias prompt (Hernandez et al., 2023b). The color of each pixel represents the probability of that token when projected onto the vocabulary space using the method from nostalgebraist (2021). Annotations indicate the predicted token when the probability exceeds 0.2.

Appendix C

MoD Supplementary Results

C.1 MoD Experiment Settings

We mainly follow the experimental settings of Hu et al. (2023). We maintain a batch size of 16 and set the learning rate for all methods to 3e-4. Each method is fine-tuned for two epochs on each dataset.

C.2 Mean and Variance for Routing Pattern Across Tokens

In this section, we analyze the routing patterns learned with MoD for the $\mathcal{K}$ ensemble layers during training. We measure the mean and variance of the weights across the training tokens. A higher mean suggests that the model consistently chooses this route, while a higher variance indicates variability in the routes learned for different tokens. We evaluate MoD trained on top of LoRA and MoD trained without k LoRA layers using the LLaMA 7B model on the ARC easy subset.

According to Figure C.1, for the mean metric, we observe a reverse trend with respect to the sparsity score in Figure 5.2. This aligns with our intuition that when the sparsity score of the current route is low, the routing value will be relatively larger than other routes. For the variance, we notice that when MoD is trained without k LoRA layers, it maintains a high variance throughout tuning. This suggests that many tokens are trained to select this route, but they are dynamically changing. When MoD is trained with LoRA, both the variance and mean levels stay low, indicating that the other two layers primarily contribute to the final ensemble logits. This suggests that the additional k trainable module within the MoD framework provides more predictive power to the ensemble layers, aligning with our analysis in §5.1.3.

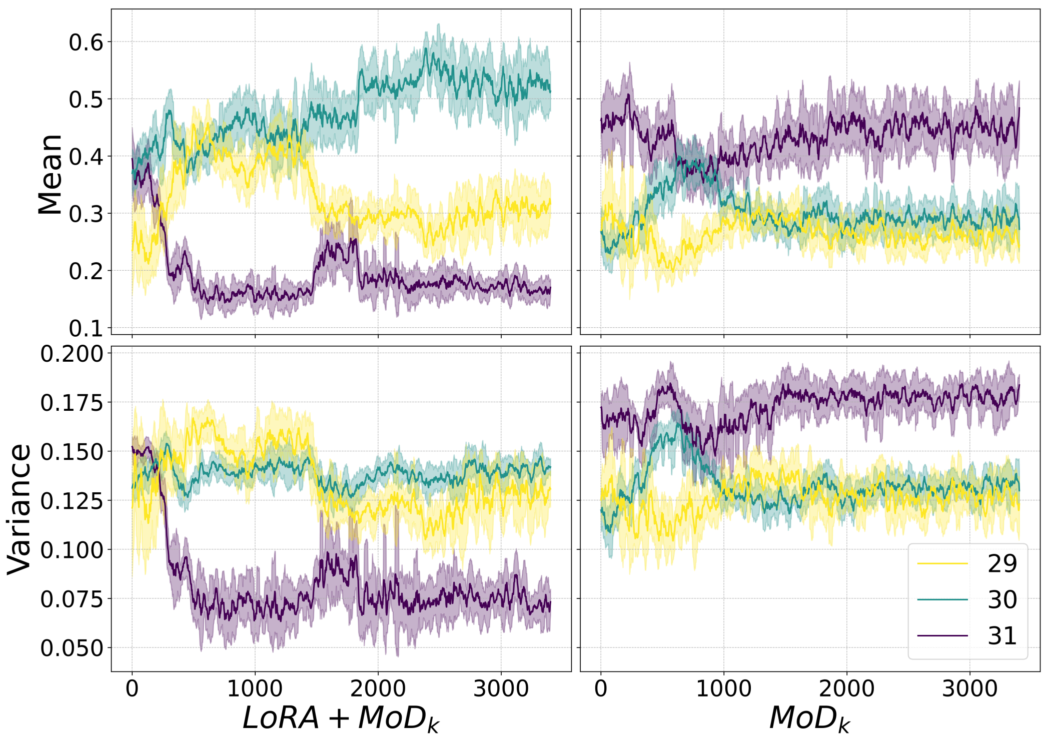

Figure C.1: Mean and variance for MoD (right) and MoD trained with k LoRA layers (left). The curve is smoothed using moving average smoothing with a window size of 3 and $k = 3$.

C.3 MoD Sparse Routing with Different Top-K Values

We also select a larger ensemble layer range to increase opportunities for early exit. We use $k = 4$ for this section, with results for BoolQ, OBQA, and MAWPS presented in Figure C.2

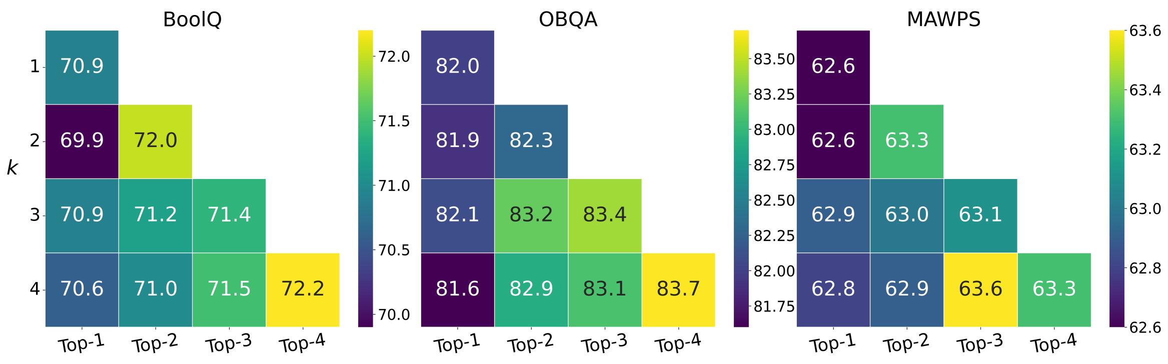

Figure C.2: Accuracy scores for different k ensemble layer ranges and Top-K sparse routing values. Lighter colors indicate better performance. Results evaluated on BoolQ, OBQA, and MAWPS testset.

Appendix D

Miscellaneous

D.1 Related Work Categorization

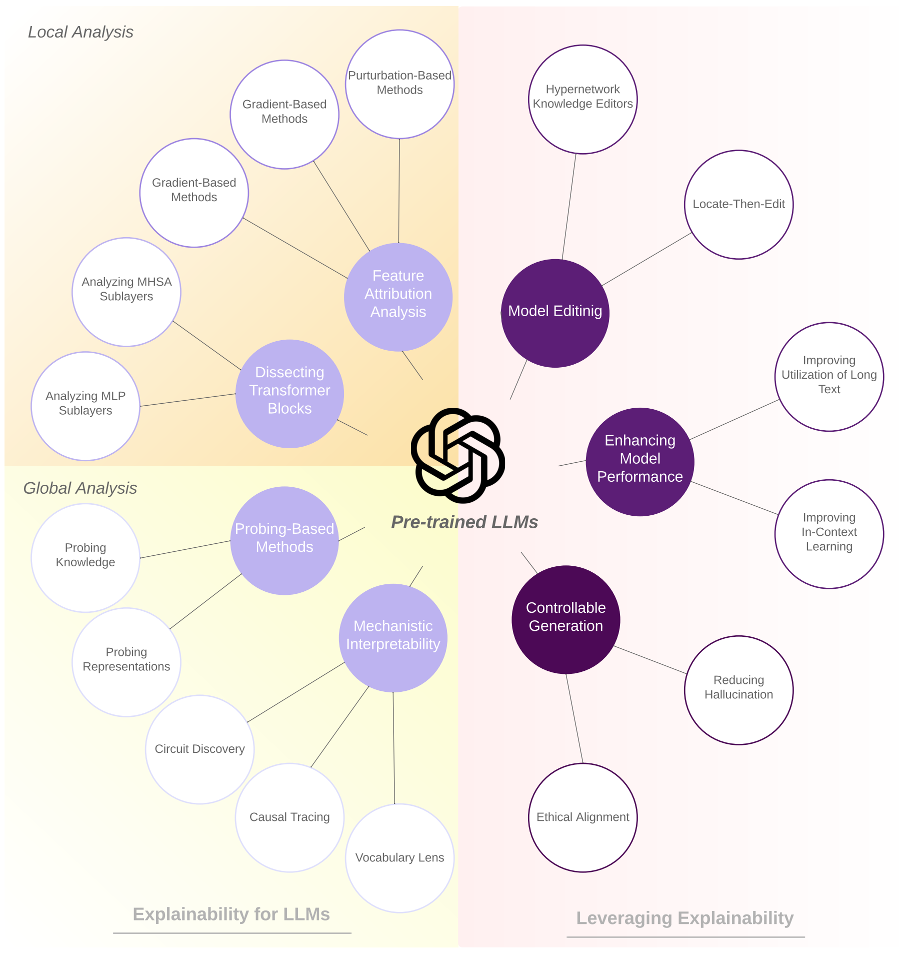

Figure D.1: Categorization of literature on explainability in LLMs, focusing on techniques (left, Section 3.1) and their applications (right, Section 3.2).

Figure D.1 presents a structured categorization of explainability methods for pre-trained language models (LMs). We divide these into two broad domains: Local

Analysis and Global Analysis. Local Analysis covers feature attribution and transformer block analysis, delving into detailed operations of models. Global Analysis, on the other hand, includes probing-based methods and mechanistic interpretability, offering a comprehensive understanding of model behaviors and capacities. Beyond understanding, we also explore applications of these insights in enhancing LLM capabilities, focusing on model editing, capability enhancement, and controlled generation. For detailed related work section please refer to §3.

D.2 Projecting Value Vectors onto Vocabulary Space

In this section we provide details from Geva et al. (2022a) and Lee et al. (2024) that demonstrate that MLP value vectors promote or suppress the likelihood of tokens.

We start from Equation 2.6:

$$\texttt{MLP}^{\ell}(\mathbf{x}^{\ell}) = \sum_{i=1}^{d_{mlp}} \sigma(\mathbf{x}^{\ell} \cdot \mathbf{k}_i^{\ell})\mathbf{v}_i^{\ell} = \sum_{i=1}^{d_{mlp}} m_i^{\ell}\mathbf{v}_i^{\ell}.$$

Thus, we can consider the update from $\texttt{MLP}^{\ell}$ as d_{mlp} *sub-updates*, each sub-update being $m_i^{\ell}\mathbf{v}_i^{\ell}$.

We can then analyze the influence that each sub-update has on the output distribution, or the probability of generating token $w \in V$ (taken from Geva et al. (2022a) and Lee et al. (2024)):

$$p\big(w \mid \mathbf{x}^{\ell} + m_i^{\ell}\mathbf{v}_i^{\ell}, E\big) = \frac{\exp\big(\mathbf{e}_w \cdot \mathbf{x}^{\ell} + \mathbf{e}_w \cdot m_i^{\ell}\mathbf{v}_i^{\ell}\big)}{Z\big(E(\mathbf{x}^{\ell} + m_i^{\ell}\mathbf{v}_i^{\ell})\big)} \propto \exp\big(\mathbf{e}_w \cdot \mathbf{x}^{\ell}\big) \cdot \exp\big(\mathbf{e}_w \cdot m_i^{\ell}\mathbf{v}_i^{\ell}\big) \tag{D.1}$$

where $\mathbf{e}_w$ is the token embedding of w, and Z is the softmax normalization factor. This indicates that when $\mathbf{e}_w \cdot m_i^{\ell}\mathbf{v}_i^{\ell} > 0$, the likelihood of w increases, while $\mathbf{e}_w \cdot m_i^{\ell}\mathbf{v}_i^{\ell} < 0$ decreases the likelihood.