Minor Mathematical Mechanisms in Large Language Models

Author: **Denis Sutter**[1], Supervisors: **Tiago Pimentel**[1], **Marius Mosbach**[2]
[1]ETH Zürich, [2]Mila Quebec AI Institute
densutter@ethz.ch, tiago.pimentel@inf.ethz.ch, marius.mosbach@mila.quebec

Note: This is a research project report, and not a publication-ready research.

Abstract

Large language models (LLMs) have transformed the field of natural language processing (NLP) by opening up a new learning paradigm, called in-context learning (ICL). Considerable work has been done to understand the underlying mechanisms driving ICL. However, no conclusive evidence has been found, and seemingly contradictory explanations have emerged. In this work, we investigate two seemingly contrasting viewpoints on in-context learning given mathematical problems. Some studies have demonstrated that large language models can perform mathematical algorithms, such as solving linear regression. However, other work suggests that ICL primarily relies on simpler mechanisms, such as heuristics and similarity computations. This work aims to bridge the gap between these two viewpoints on in-context learning by investigating the extent to which mathematical algorithms influence the predictions of Llama 3.2 3B and Llama 3.1 8B. In doing so, we provide indications that mathematical mechanisms exist in LLMs, vary across different mathematical problems, and have only limited influence on the predictions of the LLM.

1 Introduction

Large language models (LLMs) changed the field of natural language processing (NLP) in recent years. One of the most intriguing insights is that large language models open up a completely new learning paradigm, called in-context learning (ICL). There is considerable research evaluating the mechanisms behind in-context learning. While there exists work proposing a unifying view on in-context learning (e.g., Han et al., 2024), the mechanisms underlying it remain largely obscure and a focus of current research efforts. There exist multiple viewpoints that seem to contradict one another.

Let a **mathematical mechanisms**, be a circuit inside an LLM which uses mathematical steps to solve a problem (e.g. the closed form solution of linear regression)—we put this in contrast to other mechanisms devoid of such explicit mathematical steps, such as, e.g. pattern matching. One viewpoint on ICL is that LLMs use such mathematical mechanisms to solve problems. One problem especially well represented in ICL research is linear regression (e.g. Akyürek et al., 2022, Bai et al., 2023). There exists extensive research showing that, when trained from scratch in synthetic settings, LLMs exhibit algorithmic behaviour (Akyürek et al., 2022); this behaviour not only results in good performance but also enables the computation of regression weights using a mathematical mechanism. Further, Han et al. (2024) and Bai et al. (2023) provide theoretical and empirical evidence for the ability of LLMs to differentiate between mathematical mechanisms, deciding which algortithm to use for a given input. We will refer to this viewpoint as the **mathematical mechanism viewpoint** for the remainder of this work. However, counterintuitively to the prevailing interest in linear regression in ICL research, there is no conclusive evidence on this mathematical mechanism in huge autoregressive transformer models trained on natural language.[1]

A second popular viewpoint is that the mechanisms responsible for ICL in LLMs are far simpler and do not include explicit mathematical mechanisms as suggested in the mathematical mechanism viewpoint. There exists a literature presenting mechanisms on mathematical tasks using easier methods like heuristics (Nikankin et al., 2024) or similarity calculations of samples to predict an an-

[1]Let us note that LLMs encompass a broader range of models and training regimes, and in this work, we focus specifically on large transformers trained on natural language as LLMs. Preliminary research on two current popular and freely available models with a reasonable size (LLama 3.1 8B and LLama 3.2 3B; Dubey et al., 2024) further show only mediocre performance on linear regression.

swer (Yu and Ananiadou, 2024). Let us refer to this viewpoint as the **simple mechanism viewpoint**. We notice that works under this viewpoint focus on finding the primary (most important) mechanisms explaining the behaviour of transformers (e.g. Hanna et al., 2024, Tigges et al., 2024).

In our work, we provide an attempt to bridge these two seemingly contradictory viewpoints. We take inspiration from recent work (Pan et al., 2023; Li et al., 2024) which demonstrates that multiple mechanisms can be included in a large language model and work toward a prediction. Park et al. (2024) further show that multiple mechanisms can be responsible for a model's output; he found that a convex sum of different mechanisms outputs can approximate the models' behaviour nearly perfectly. Here, we will call mechanisms which have a big influence on a model's output as **major mechanisms** and mechanisms with little influence as **minor mechanisms**. In Park et al. (2024), this can be seen as major mechanisms having a big coefficient in the convex sum, and minor mechanisms having a small (but still non-zero) coefficient. Given this state of research we now propose three hypotheses:

1. $\mathcal{H}_1$: LLMs include mathematical mechanisms
2. $\mathcal{H}_2$: LLMs can use different mathematical mechanisms when given different input problems to solve.
3. $\mathcal{H}_3$: Mathematical mechanisms are minor mechanisms in current popular LLMs.

While we are unable to conclusively prove either of the hypotheses above, this work nonetheless makes two valuable contribution to the literature:

- We propose three hypotheses that aim to bridge the gap between the mathematical and the simple mechanism viewpoints.
- Our experiments produce preliminary results that provide justification for further research into these hypotheses.

Our code is available on GitHub at `https://github.com/GenK-ai/In-context-learning`.

2 Background

2.1 Language Modelling

In this section we present a theoretical explanation of language modelling. A language can be defined as a set of finite sequences given a vocabulary $\Omega_{\texttt{voca}}$; these sequences are themselves defined as a string of words or tokens $\boldsymbol{\omega} = (\omega_1, \omega_2, ...) \in \Omega^*_{\texttt{voca}}$. Lets now assume we have two such sequences $\overleftarrow{\boldsymbol{\omega}}$ and $\overrightarrow{\boldsymbol{\omega}}$. We define a language model as the conditional probability distribution:

$$\hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}}) = \prod_{t=1}^{|\overrightarrow{\boldsymbol{\omega}}|} \hat{p}(\overrightarrow{\omega}_t \mid \overleftarrow{\boldsymbol{\omega}}, \overrightarrow{\boldsymbol{\omega}}_{<t}) \tag{1}$$

We note that current autoregressive transformers, to which the evaluated LLMs in this work belong, are built to approximate this.

In this work, we will specifically look at language models implemented as autoregressive transformers. Let us assume we have a transformer with L layers. Let us define $\mathbf{A}_{\iota_i^{\mathtt{a}}}$ as the **activations of the i-th model layer**, given by:

$$\mathbf{A}_{\iota_i^{\mathtt{a}}} = f_{\mathtt{i}}(\mathbf{A}_{\iota_{i-1}^{\mathtt{a}}}) \tag{2}$$

for $0 \leq i \leq L$ and where $\mathbf{A}_{\iota_{-1}^{\mathtt{a}}}$ is the embedding of the input tokens and $f_{\mathtt{i}}$ is the transformer's i-th layer. We will also analyse the sublayers each transformer layer. In the i-th layer, we first compute the **key projection** ($\iota_i^{\mathtt{k}}$), **query projection** ($\iota_i^{\mathtt{q}}$), and **value projection** ($\iota_i^{\mathtt{v}}$) as:

$$\mathbf{A}_{\iota_i^{\mathtt{k}}} = \mathbf{W}_i^{\mathtt{k}} \mathbf{A}_{\iota_{i-1}^{\mathtt{a}}}, \qquad \mathbf{A}_{\iota_i^{\mathtt{q}}} = \mathbf{W}_i^{\mathtt{q}} \mathbf{A}_{\iota_{i-1}^{\mathtt{a}}}$$
$$\mathbf{A}_{\iota_i^{\mathtt{v}}} = \mathbf{W}_i^{\mathtt{v}} \mathbf{A}_{\iota_{i-1}^{\mathtt{a}}} \tag{3}$$

where $\mathbf{W}^{\mathtt{k}}, \mathbf{W}^{\mathtt{q}}, \mathbf{W}^{\mathtt{v}}$ are linear projection matrices. These activations are then used to compute an **output projection** ($\iota_i^{\mathtt{o}}$):

$$\mathbf{A}_{\iota_i^{\mathtt{o}}} = \mathbf{A}_{\iota_{i-1}^{\mathtt{a}}} + f_{\mathtt{att}}(\mathbf{A}_{\iota_i^{\mathtt{k}}}, \mathbf{A}_{\iota_i^{\mathtt{q}}}, \mathbf{A}_{\iota_i^{\mathtt{v}}}) \tag{4}$$

where $f_{\mathtt{att}}(.)$ is the **attention function**, which combines $\mathbf{A}_{\iota_i^{\mathtt{k}}}$, $\mathbf{A}_{\iota_i^{\mathtt{q}}}$ and $\mathbf{A}_{\iota_i^{\mathtt{v}}}$. Finally, we run these activations through a multi-layer perceptron (mlp) ($f_{(\mathtt{mlp,i})}(\cdot)$) resulting in **mlp activations** ($\iota_i^{\mathtt{mlp}}$):

$$\mathbf{A}_{\iota_i^{\mathtt{mlp}}} = f_{(\mathtt{mlp,i})}(\mathbf{A}_{\iota_i^{\mathtt{o}}}) \tag{5}$$

The full transformer layer then ends with a normalisation function ($f_{(\mathtt{norm,i})}(\cdot)$), which computes:

$$\mathbf{A}_{\iota_i^{\mathtt{a}}} = f_{(\mathtt{norm,i})}(\mathbf{A}_{\iota^{\mathtt{mlp}}}) \tag{6}$$

2.2 In-Context Learning (ICL)

Lampinen et al. (2024) state that *"any sequence task in which context non-trivially reduces loss can be interpreted as eliciting a kind of ICL"*. While

we agree on this more general definition, our focus lies in the more restricted field of few-sample in-context learning, which has garnered significant attention by recent research. The task in few-shot in-context learning for a language model is to predict an **output response** $\boldsymbol{\omega}^{\texttt{out}} \in \Omega^*_{\texttt{voca}}$ from: (i) a given **input query** $\boldsymbol{\omega}^{\texttt{inp}} \in \Omega^*_{\texttt{voca}}$, and a set of υ **ICL samples** $\{\widehat{\boldsymbol{\omega}}^{\texttt{inp}}_i, \widehat{\boldsymbol{\omega}}^{\texttt{out}}_i\}_{i=1}^{\upsilon}$. Each of these ICL samples is composed of an example output response and input query, so $(\widehat{\boldsymbol{\omega}}^{\texttt{inp}}_i, \widehat{\boldsymbol{\omega}}^{\texttt{out}}_i) \in \Omega^*_{\texttt{voca}} \times \Omega^*_{\texttt{voca}}$. Given that much current research only uses few-shot samples in their in-context setting, we must note that we also utilize a **context prompt** $\boldsymbol{\omega}^{\text{ctxt}} \in \Omega^*_{\texttt{voca}}$.

Given the formulation of language models in §2.1, we say that a language model demonstrates few-shot ICL ability if it outputs high probability for:

$$\hat{p}(\boldsymbol{\omega}^{\texttt{out}} \mid \boldsymbol{\omega}^{\text{fullinp}}) \tag{7}$$

where $\boldsymbol{\omega}^{\text{fullinp}}$ is a combination of $\{(\widehat{\boldsymbol{\omega}}^{\texttt{inp}}_i, \widehat{\boldsymbol{\omega}}^{\texttt{out}}_i)\}_{i=1}^{\upsilon}$, $\boldsymbol{\omega}^{\text{ctxt}}$ and $\boldsymbol{\omega}^{\texttt{inp}}$. In our setting, this combination $\boldsymbol{\omega}^{\text{fullinp}}$ involves the tokenisation of a prompt generated with a prompt generation function $\hat{f}_{\texttt{PG}}(.,.)$ (see §5.4).[2]

3 Related work

We present related work about analysis methods of in-context learning in §3.1 followed by insights gained using such methods in §3.2. We further present related work on feature attribution in §3.3.

3.1 In-Context Learning Analysis Methods

In this section, we present methods used in recent literature to understand in-context analysis. §3.1.1 presents state-of-the-art models. §3.1.2 presents literature on probing, which is the method we will use in our work.

3.1.1 Mechanistic Interpretability

In recent years the interpretability research of deep models has garnered significant interest. This can be understood by the need for trust in models deployed in practice. We will leave a detailed theoretical discussion on the different approaches to Geiger et al. (2023a). In this section we will present some recent methods for mechanistic interpretability.

[2]Our setting is a subtask of in-context learning according to Lampinen et al. (2024). However, we refer to our setting as in-context learning throughout this report for the sake of reader convenience.

The work of Hanna et al. (2024) seeks to identify the minimal set of neurons and edges that underpin the behaviour of a model, shedding light on its inner workings. This method is used by more recent work (Tigges et al., 2024). Another proposed method for analysing neural circuitry is path patching Goldowsky-Dill et al. (2023). Path patching provides a method for determining how well a selection of paths (in the network) are responsible for the network behaviour on a given input distribution.

One of the most popular methods is distributed alignment search (DAS). This approach was proposed by Geiger et al. (2023b) and uses a rotation on hidden features to find features corresponding to high-level variables (e.g. weights of a linear regression). This method has been used in multiple other studies, including Wu et al. (2024). However, methods like DAS are primarily designed to identify major contributors of predictions, making them unsuitable for our specific application.

3.1.2 Probes

Another approach to understand what an LLM does intrinsically is to analyse whether some **intermediate results** (e.g. weights of linear regression) can be retrieved from the hidden features of an LLM. To this end, a separate model (typically called a probe) is trained to to predict the intermediate results given the hidden features. This model's test loss is then used to estimate the mutual information between our high-level concept and the features/activations of a hidden layer (Pimentel et al., 2020). An example of this method can be seen in Akyürek et al. (2022).

Another recently proposed method is circuit probing, as proposed by Lepori et al. (2023). Instead of using vanilla probing to search for features, it masks weights of previous model parts to retrieve a representation corresponding to a feature. This method also relies on the mechanism under analysis being the major contributor to the predictions.

3.2 Insights from Prior Studies on In-Context Learning

Many different work investigated in-contest learning and the mechanisms underlying it (Pan et al., 2023; Olsson et al., 2022; Hahn and Goyal, 2023; Wang et al., 2023; Zhong et al., 2024; Zhang et al., 2023; Tigges et al., 2024).

A popular opinion is that transformers implicitly implement specific algorithms to solve their tasks (Zhang et al., 2023; Han et al., 2023). However, it

is unclear which kind of algorithms are used. Dong et al. (2022) show the diversity of theories that exist in this field. One example is gradient learning through weights (Akyürek et al., 2022; Von Oswald et al., 2023; Dai et al., 2023). Other work analyses the implementation of linear regression in LLMs (Bai et al., 2023; Akyürek et al., 2022). Akyürek et al. (2022) suggests transformers are able to compute the weights of linear regression. There is also work demonstrating the existence of multiple algorithms at play and how they are combined (Li et al., 2024; Bai et al., 2023; Han et al., 2024; Pan et al., 2023; Park et al., 2024).

Other recent work suggests that less advanced mechanisms are at play. Yu and Ananiadou (2024) suggests a weighting of the few-shot labels $\widehat{\boldsymbol{\omega}}^{\texttt{out}}$ according to the similarity of the few-shot features $\widehat{\boldsymbol{\omega}}^{\texttt{inp}}$ to the features $\boldsymbol{\omega}^{\texttt{inp}}$ of the requested output $\boldsymbol{\omega}^{\texttt{inp}}$. Nikankin et al. (2024) suggests, that given mathematical problems, LLMs resort to a bag of heuristics (example heuristic: if the result given the task of adding the given inputs lies between 0 and 20). Other work Feng et al. (2025) suggest that LLMs learn facts which then are retrieved using extractive structures, enabling out-of-context reasoning.

3.3 Feature Attribution

To determine if an LLM can differentiate between different algorithms, we must first identify where algorithms are likely stored in the LLM. We do this by determining which features are important for a specific output, and take inspiration from the vast literature of **Feature Attribution**. There are multiple proposed approaches to this problem.

First, **model-centric** approaches propose methods specifically developed for specific models, such as convolutional neural networks (CNNs; e.g., Zeiler and Fergus, 2014) or transformers (e.g., Achtibat et al., 2024). While it has been shown for transformers, that those methods are more faithful and faster than model-agnostic methods, they are not yet of-the-shelf applicable to newer models without rewriting the provided library. We therefore leave it for future work to return to this kind of model-centric methods and continue with model-agnostic methods.

Model-agnostic non-gradient based approaches propose methods which rely, e.g., on local approximations of the behaviour of the model to attribute it to input features (e.g. LIME; Ribeiro et al., 2016). While there exist studies suggesting that LIME methods are more effective than gradient based methods (Ahern et al., 2019), non relying on gradients often comes with high computational costs. Therefore, we decided to focus on the methods that leverage gradients. We leave the exploration of non-gradient methods to future work.

Model-agnostic gradient based approaches rely on a model's gradients to perform feature attribution.[3] First, we will use use gradients directly to produce saliency maps (similar to Simonyan et al., 2013), due to its small computational requirements. We also explore more complex methods to mitigate the saturation issue, as described by Sturmfels et al. (2020) and Sundararajan et al. (2017), where the gradients of input features that the model heavily relies on can have small magnitudes. We explore SmoothGrad (Smilkov et al., 2017), Expected Gradients (Sturmfels et al., 2020), and Important Direction Gradient Integration (Yang et al., 2023) for their comparable small computational requirements. These three methods "stabilise" the gradient and mitigate the saturation issue by combining gradients over multiple input points instead of just one.

4 Methods

In this section, we present the methods for probing (§4.1) followed by our method to separate different mechanisms in LLMs (§4.2) and closing with a description on the used activation patching (§4.3).

4.1 Probing

To find indications for mathematical mechanisms ($\mathcal{H}_1$), we use a probing model similar to that of Akyürek et al. (2022). The probe's inputs are the activations $\mathbf{A}_\iota \in \mathbb{R}^{n \times d_p^{\texttt{feat}}}$ of some hidden layer ι, where we have an input consisting of n tokens and a **feature dimension** per token of $d_p^{\texttt{feat}}$. The probing model $m_{\texttt{prob}}(.)$ now includes the following 3 steps. We first linearly project each token representation as in:

$$f_{\texttt{proj}}(\mathbf{A}_\iota) = \mathbf{W}_p \mathbf{A}_\iota^T \tag{8}$$

As second step we aggregate the values across input positions to a single vector:

$$f_{\texttt{aggr}}(\mathbf{A}_\iota) = f_{\texttt{proj}}(\mathbf{A}_\iota) f_{\texttt{sm}}(\mathbf{a}_{1:n}) \tag{9}$$

The last step feeds the vector to a linear transformation:

$$m_{\texttt{prob}}(\mathbf{A}_\iota) = \mathbf{W}_l \, f_{\texttt{aggr}}(\mathbf{A}_\iota) \tag{10}$$

[3] For a comprehensive overview of the field we point to Wang et al. (2024) for a technical review.

Where $\mathbf{W}_p \in \mathbb{R}^{d_p^{\texttt{proj}} \times d_p^{\texttt{feat}}}$ is a projection matrix with projection dimension $d_p^{\texttt{proj}}$. $\mathbf{a}_{1:n} \in \mathbb{R}^n$ is an **attention vector**. The **softmax function** $f_{\texttt{sm}}(.)$ maps these inputs to a probability distribution:

$$f_{\texttt{sm}}(\mathbf{a}_{1:n})_i = \frac{e^{\mathbf{a}_i}}{\sum_{j=1}^{n} e^{\mathbf{a}_j}} \tag{11}$$

$\mathbf{W}_l \in \mathbb{R}^{d_p^{\texttt{IR}} \times d_p^{\texttt{proj}}}$ is a linear layer with $d_p^{\texttt{IR}}$ being the **dimension of the intermediate result** we are probing for. $\mathbf{W}_p, \mathbf{a}, \mathbf{W}_l$ are trainable parameters.

This model is trained to predict an intermediate result (IR) $\mathbf{w}^{\texttt{IR}} \in \mathbb{R}^{d_p^{\texttt{IR}}}$ with mean squared error as its loss function:

$$\mathcal{L}(m_{\texttt{prob}}(\mathbf{A}_\iota), \mathbf{w}^{\texttt{IR}}) = ||m_{\texttt{prob}}(\mathbf{A}_\iota) - \mathbf{w}^{\texttt{IR}}||^2 \tag{12}$$

Lets call the **trained probing model** $m_{\texttt{prob}}(.)$.

To get the **final probing value** $f_{\texttt{prob}}(.,.)$ we approximate the generalisation error $f_{\texttt{prob}}(.,.)$ of the model on a test set $\boldsymbol{D}^{(\texttt{p,test})}$ (p stands for probing):

$$f_{\texttt{prob}}(m_{\texttt{prob}}(.), \boldsymbol{D}^{(\texttt{p,test})}) = \sum_{(\mathbf{A}_\iota, \mathbf{w}^{\texttt{IR}}) \in \boldsymbol{D}^{(\texttt{p,test})}} \mathcal{L}(m_{\texttt{prob}}(\mathbf{A}_\iota), \mathbf{w}^{\texttt{IR}}) \tag{13}$$

This generalisation error then represents how well an intermediate variable is encoded in the input, which in our case is the activations of a hidden layer. Other more sophisticated methods exist to determine the existence of intermediate results. However, most of these methods rely on the mechanism under evaluation being a major contributor to the prediction (compare §3.1). Given our hypothesis that mathematical mechanisms are minor contributors ($\mathcal{H}_3$), we use this simpler method.

4.2 Mechanism Separability

In this section, we present our approach to find indications that the LLM actively selects mathematical mechanisms to inform its predictions ($\mathcal{H}_2$).

In §4.2.1, we present a method to compute the importance of each feature for a given task. These importance values are used to determine how a model distinguishes between different tasks and approaches them differently. In §4.2.2, we then describe how we combine these feature importance values for two different tasks to obtain a task difference importance mapping. This task difference importance mapping will allow us to identify which features are more relevant for task 1 and which for task 2, thereby enabling feature separation.

4.2.1 Feature Attribution

We will describe in this section how we calculate the importance of the features given the activations $\mathbf{A}_\iota$ of a specific hidden layer $\iota \in \Omega_\iota$ in an LLM $\hat{p}(. \mid .)$. Lets assume we have a dataset $\boldsymbol{D}^{\texttt{FA}}$ for feature attribution (FA). A sample of this dataset is given as $(\overleftarrow{\boldsymbol{\omega}}, \overrightarrow{\boldsymbol{\omega}}) \in \boldsymbol{D}^{\texttt{FA}}$, where $\overleftarrow{\boldsymbol{\omega}}$ is an input prompt to the LLM and $\overrightarrow{\boldsymbol{\omega}}$ is the token the LLM should predict given $\overleftarrow{\boldsymbol{\omega}}$. We now consider a feature's importance to be its gradient as in:[4]

$$f_{\texttt{FA}}(\hat{p}(. \mid .), \mathbf{A}_\iota, \boldsymbol{D}^{\texttt{FA}}) = \frac{1}{|\boldsymbol{D}^{\texttt{FA}}|} \sum_{(\overleftarrow{\boldsymbol{\omega}}, \overrightarrow{\boldsymbol{\omega}}) \in \boldsymbol{D}^{\texttt{FA}}} \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})}{\partial \mathbf{A}_\iota} \tag{14}$$

This method was first proposed by Simonyan et al. (2013) and the generated map is referred to as **saliency map.** We note that research has shown multiple problems with salency maps, an example would be noisy attributions (Kim et al., 2019). However, after experimenting with more advanced methods (see App. A and §3.3), we found that using more advanced methods did not or just marginally lead to clearer results, not justifying the additional computation overhead. We assume that averaging over multiple samples tends to reduce the impact of noise. We leave an extensive analysis on more sophisticated feature attribution for future work.

4.2.2 Feature Difference Calculation

We will use the feature importance values from §4.2.1 to assess the model's ability to differentiate between different tasks. We therefore calculate a combined feature importance mapping. Before we can proceed, we must introduce a **ranking function** $f_{\texttt{rank}}(.)$ that maps a vector $\mathbf{x} \in \mathbb{R}^y$ for $y \in \mathbb{N}$ to a vector of the same size, indicating which position an element would have if we sort the vector. Therefore, the biggest element in the vector would be mapped to y, the smallest element to 1 and so on. If all elements in $\mathbf{x}$ are pairwise different we can write it as follows:

$$f_{\texttt{rank}}(.) : \mathbb{R}^y \to \mathbb{N}^y$$
$$f_{\texttt{rank}}(\mathbf{x})_i = 1 + \sum_{j=1}^{y} \mathbb{1}(\mathbf{x}_j > \mathbf{x}_i) \tag{15}$$

[4] We take inspiration in the Captum library: https://captum.ai/api/saliency.html#captum.attr.Saliency.attribute.

where $\mathbb{1}(.)$ is the indicator function:

$$\mathbb{1}(\beta) = \begin{cases} 1 & \beta \\ 0 & otherwise \end{cases} \tag{16}$$

If there are elements in $\mathbf{x}$ that are equal, the ties for these elements are broken randomly.[5]

Lets now assume we have two tasks, τ_1 and τ_2, and two corresponding datasets, $\boldsymbol{D}^{\texttt{FA}}_{\tau_1}$ and $\boldsymbol{D}^{\texttt{FA}}_{\tau_2}$, respectively. We now create a **combined feature importance mapping** $f_{\texttt{FAC}}(.,.,.,.)$. that considers both datasets for an LLM $\hat{p}(.\mid.)$ and the features $\mathbf{A}_\iota$ of a hidden layer $\iota \in \Omega_\iota$:

$$\begin{aligned} f_{\texttt{FAC}}(\hat{p}(.\mid.), \mathbf{A}_\iota, \boldsymbol{D}^{\texttt{FA}}_{\tau_1}, \boldsymbol{D}^{\texttt{FA}}_{\tau_2}) = \\ f_{\texttt{rank}}(f_{\texttt{FA}}(\hat{p}(.\mid.), \mathbf{A}_\iota, \boldsymbol{D}^{\texttt{FA}}_{\tau_1})) \\ - f_{\texttt{rank}}(f_{\texttt{FA}}(\hat{p}(.\mid.), \mathbf{A}_\iota, \boldsymbol{D}^{\texttt{FA}}_{\tau_2})) \end{aligned} \tag{17}$$

This combined feature importance mapping should represent which features are relevant for one task in relation to the other. A large value for a feature should indicate that it is more important for task 1, while a small value indicates more importance for task 2. We are aware that, given current research (e.g. Wu et al., 2024), our method can be seen as less sophisticated. We assign features in the hidden representations of the model to different mechanisms. More sophisticated methods, such as those described in Wu et al. (2024), typically involve applying transformations of the features before assigning the transformed features to the algorithms. However, to the best of our knowledge, those methods rely on the evaluated mechanisms being major contributors. Therefore, we cannot use this methods given our hypothesis that the evaluated mechanisms are minor mechanisms ($\mathcal{H}_3$).

4.3 Activation Patching

Activation patching is used in recent literature (e.g. Yeo et al., 2024) for faithful explanations. In our setting, we need for activation patching given a specific hidden layer $\iota \in \Omega_\iota$ two different inputs. Let us call them base input $\boldsymbol{\omega}^{\texttt{B}}$ and source input $\boldsymbol{\omega}^{\texttt{S}}$. Let now $\mathbf{A}^{\texttt{S}}_\iota$ be the activations of layer ι given the source input $\boldsymbol{\omega}^{\texttt{S}}$ of the LLM $\hat{p}(.\mid\boldsymbol{\omega}^{\texttt{S}})$. Activation patching now exchanges a subset $\hat{\mathbf{A}}_\iota{}^{\texttt{B}}$ of $\mathbf{A}_\iota$ ($\hat{\mathbf{A}}_\iota{}^{\texttt{B}} \subseteq \mathbf{A}_\iota$) in the forward pass of the LLM $\hat{p}(.\mid\boldsymbol{\omega}^{\texttt{B}})$ with the corresponding subset $\hat{\mathbf{A}}_\iota{}^{\texttt{S}} \subseteq \mathbf{A}^{\texttt{S}}_\iota$. Let this patched LLM be defined as:

$$\hat{p}(.\mid\boldsymbol{\omega}^{\texttt{B}})_{\hat{\mathbf{A}}_\iota{}^{\texttt{B}} \rightarrow \hat{\mathbf{A}}_\iota{}^{\texttt{S}}} \tag{18}$$

[5] the implementation uses two chained numpy argmax functions

5 Experiments

In this section, we discuss which LLMs and layers we analyse (§5.1). We further describe the tasks (§5.2) we use to generate our datasets (§5.3) and prompts (§5.4). We further describe the experimental setting of our probing (§5.5) and activation patching (§5.6).

5.1 Evaluated LLM and Layers

In our experiments, we evaluate the Llama-3.2-3B[6] model from meta. The architecture of this model is described in App. E.1. From all 28 layers ($L = 28$), we evaluate the activations on $\iota^{\texttt{q}}_i$, $\iota^{\texttt{k}}_i$, $\iota^{\texttt{v}}_i$, $\iota^{\texttt{o}}_i$, $\iota^{\texttt{mlp}}_i$ for $0 \leq i \leq 27$. We define the set of all layers as Ω_ι:

$$\Omega_\iota = \bigcup_{i \in \{q,k,v,o,mlp\}} \{\iota^{\texttt{i}}_i\}^{27}_{i=0} \tag{19}$$

We also evaluate Llama 3.1 8B[7]. The results for Llama 3.1 8B are presented in App. C.

5.2 Tasks

Two of the most popular mathematical problems used to study ICL are linear regression and linear classification (e.g. Akyürek et al., 2022). In preliminary work, we observed that while we achieved useful results for linear regression, we got no conclusive results for linear classification. We therefore reviewed what we would like from a task in this field to get consistent results:

- The input includes no information about the intermediate results.
- Given a reasonable amount of examples (input and corresponding output) the intermediate results can be reconstructed perfectly.
- We are not aware of any simple algorithm that can generalise to the problem without computing the intermediate results using some kind of mathematical mechanism. Note that for the purpose of this work, it is irrelevant which specific algorithm computes the intermediate results.
- The methods used to solve one task should not be useful for solving the second task, and vice versa.

[6] https://huggingface.co/meta-llama/Llama-3.2-3B

[7] https://huggingface.co/meta-llama/Llama-3.1-8B

Comparing these conditions to linear classification we see:

- The intermediate results (weights of logistic regression) are not the only weights that can satisfy the classification of samples given a limited amount of samples.
- There exist a variety of different valid approaches the model could use to make useful predictions without explicitly computing the logistic regression weights, including, e.g., the nearest neighbour algorithm.

Given the absence of useful results and the mentioned problems of linear classification, we instead opt for a Manhattan distance problem. In the following two sections we describe our two problems.

5.2.1 Linear Regression

Linear regression (LR) can be defined as the problem of computing some output $f_{\texttt{LR}}(\mathbf{x}, \mathbf{w}^{\texttt{IR}})$[8] given some input $\mathbf{x} \in \mathbb{R}^d$ and weights $\mathbf{w}^{\texttt{IR}} \in \mathbb{R}^d$ where d is the number of features fed to the linear regression:

$$f_{\texttt{LR}}(\mathbf{x}, \mathbf{w}^{\texttt{IR}}) = \sum_{i=1}^{d} \mathbf{x}_i \, \mathbf{w}_i^{\texttt{IR}} \tag{20}$$

Given this problem, the intermediate result, we will probe for, is the weight $\mathbf{w}^{\texttt{IR}}$.

5.2.2 Manhattan Distance

This problem computes the **Manhattan distance** (MD) $f_{\texttt{MD}}(\mathbf{x}, \mathbf{w}^{\texttt{IR}})$ from a point $\mathbf{x} \in \mathbb{R}^d$ and weights $\mathbf{w}^{\texttt{IR}} \in \mathbb{R}^d$ encoding a reference point in $\mathbb{R}^d$:

$$f_{\texttt{MD}}(\mathbf{x}, \mathbf{w}^{\texttt{IR}}) = \sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{w}_i^{\texttt{IR}}| \tag{21}$$

Given this problem, the intermediate result we will probe for are the weights $\mathbf{w}^{\texttt{IR}}$.

5.3 Dataset Generation

In our experiments, we will fix $d = 2$, using the two-dimensional versions of the problems in §5.2. We further restrict the input values and intermediate results to the range of integers from 0 to 22. The set of inputs $\mathbf{x}$ and weights $\mathbf{w}^{\texttt{IR}}$:

$$\Omega_N = \{0, 1, 2, ..., 22\}^2 \tag{22}$$

[8] $\texttt{IR}$ = intermediate results

We therefore can ensure that all resulting inputs and outputs of the given problems lie within the interval $[0, 1000]$. We enforce this restriction, as the models we evaluate (from the model families Llama 3.2 and Llama 3.1) tokenise only numbers in the interval $[0, 1000]$ as a single token. We do this to avoid the additional complexity of having multi-token outputs in our analysis.

For our datasets, we create for each task a dataset of datasets indexed by the intermediate weight:

$$\boldsymbol{D}^{\texttt{LR}} = \{(\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}^{\texttt{LR}}_{\mathbf{w}^{\texttt{IR}}}) \mid \mathbf{w}^{\texttt{IR}} \in \Omega_N\} \tag{23}$$

where $\boldsymbol{D}^{\texttt{LR}}_{\mathbf{w}^{\texttt{IR}}}$ is the dataset of all possible inputs with the corresponding outputs using the weight $\mathbf{w}^{\texttt{IR}}$:

$$\boldsymbol{D}^{\texttt{LR}}_{\mathbf{w}^{\texttt{IR}}} = \{(\mathbf{x}, y) \mid \mathbf{x} \in \Omega_N, f_{\texttt{LR}}(\mathbf{x}, \mathbf{w}^{\texttt{IR}}) = y\} \tag{24}$$

The same can be done for the Manhattan distance problem:

$$\boldsymbol{D}^{\texttt{MD}} = \{(\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}^{\texttt{MD}}_{\mathbf{w}^{\texttt{IR}}}) \mid \mathbf{w}^{\texttt{IR}} \in \Omega_N\} \tag{25}$$

$$\boldsymbol{D}^{\texttt{MD}}_{\mathbf{w}^{\texttt{IR}}} = \{(\mathbf{x}, y) \mid \mathbf{x} \in \Omega_N, f_{\texttt{MD}}(\mathbf{x}, \mathbf{w}^{\texttt{IR}}) = y\} \tag{26}$$

We note that we use the above dataset as is for our feature attribution experiments, while splitting it into a training and testing set for probing:

$$\boldsymbol{D}^{\texttt{LR}}_{\texttt{tst}} = f_{\texttt{sample}}(40, \boldsymbol{D}^{\texttt{LR}}), \quad \boldsymbol{D}^{\texttt{LR}}_{\texttt{trn}} = \boldsymbol{D}^{\texttt{LR}} \setminus \boldsymbol{D}^{\texttt{LR}}_{\texttt{tst}} \tag{27}$$

We can do the same for the Manhattan distance problem dataset:

$$\boldsymbol{D}^{\texttt{MD}}_{\texttt{tst}} = f_{\texttt{sample}}(40, \boldsymbol{D}^{\texttt{MD}}), \quad \boldsymbol{D}^{\texttt{MD}}_{\texttt{trn}} = \boldsymbol{D}^{\texttt{MD}} \setminus \boldsymbol{D}^{\texttt{MD}}_{\texttt{tst}} \tag{28}$$

Where $f_{\texttt{sample}}(x, y)$ returns a set of x elements sampled from y without replacement.

5.4 Prompts

Given a sample from some dataset $\boldsymbol{D}$ created in §5.3: $(\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}}) \in \boldsymbol{D}$. We then create a prompt, as described in Algorithm $\hat{f}_{\texttt{PG}}(.,.)$ (see Alg. 1). Where $f_{\texttt{to}}(.)$ is the tokeniser of the LLM and:

- When we have a linear regression dataset: $\boldsymbol{\omega}^{\text{ctxt}} = \boldsymbol{\omega}^{\text{ctxt}}_{\texttt{LR}}$ ="The output represents the result of a linear regression given 2 dimensions with output range [0-1000]: "
- When we have a Manhattan distance dataset: $\boldsymbol{\omega}^{\text{ctxt}} = \boldsymbol{\omega}^{\text{ctxt}}_{\texttt{MD}}$ ="The output represents the Manhattan Distance to a point given 2 dimensions with output range [0-1000]: "

Algorithm 1
$\hat{f}_{\texttt{PG}}(\text{Sample: } (\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}}), \text{Task Context: } \boldsymbol{\omega}^{\texttt{ctxt}})$

$\texttt{prompt}_n \leftarrow$ “”
$\texttt{prompt}_o \leftarrow$ “”
while $|f_{\texttt{to}}(\texttt{prompt}_n)| \leq 600$ **do**
  $\texttt{prompt}_o \leftarrow \texttt{prompt}_n$
  if $|\texttt{prompt}_n| > 0$ **then**
    $\texttt{prompt}_n \leftarrow \texttt{prompt}_n +$ “\n\n"
  end if
  $(\mathbf{x}, y) \leftarrow f_{\texttt{sample}}(1, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}})[0]$
  for $k = 0,\ k{+}{+},$ while $k < |\mathbf{x}|$ **do**
    $\texttt{prompt}_n \leftarrow \texttt{prompt}_n +$ “\ n" + “Feature " $+ k +$ “: " $+ \mathbf{x}[k]$
  end for
  $\texttt{prompt}_n \leftarrow \texttt{prompt}_n +$ “Output:" $+ y$
end while
$\texttt{result} \leftarrow \texttt{prompt}_o[-1]$
$\texttt{prompt}_o \leftarrow \texttt{prompt}_o[:-1]$
$\texttt{prompt}_o \leftarrow \boldsymbol{\omega}^{\texttt{ctxt}} +$ “\n\n\n" $+ \texttt{prompt}_o$
return $\texttt{prompt}_o, \texttt{result}$

5.5 Probing

In this section we present the experiments used to determine if intermediate results are computed (§5.5.1) and how well a model can differentiate between different mathematical mechanisms given different tasks (§5.5.2).

5.5.1 Are intermediate results computed

We will probe for the activations $\mathbf{A}_\iota$ given a layer $\iota \in \Omega_\iota$ for the linear regression utilizing its three datasets ($\boldsymbol{D}^{\texttt{LR}}, \boldsymbol{D}^{\texttt{LR}}_{\texttt{trn}}$ and $\boldsymbol{D}^{\texttt{LR}}_{\texttt{tst}}$) created in §5.3. We first train the probing model (using $d_p^{\texttt{proj}} = 512$). We iterate over 20000 training steps. For each training step we first sample an input:

$$(\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}}) \in \boldsymbol{D}^{\texttt{LR}}_{\texttt{trn}} \tag{29}$$

We then create a prompt $\overleftarrow{\boldsymbol{\omega}}$, as described in §5.4:

$$\overleftarrow{\boldsymbol{\omega}}, \overrightarrow{\boldsymbol{\omega}} = \hat{f}_{\texttt{PG}}((\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}}), \boldsymbol{\omega}^{\texttt{ctxt}}_{\texttt{LR}}) \tag{30}$$

This prompt is then used as input to the LLM we are evaluating, from which the activations $\mathbf{A}_\iota$ of the the hidden layer $\iota \in \Omega_\iota$ are extracted and used as input of the probing model $m_{\texttt{prob}}(.)$. We then minimise the loss with the Adam optimizer (Kingma and Ba, 2014), using one sample per training step and a learning rate of 0.0001:

$$\mathcal{L}(m_{\texttt{prob}}(\mathbf{A}_\iota), \mathbf{w}^{\texttt{IR}}) \tag{31}$$

getting $m_{\texttt{prob}}(.)$.

For the testing, we create $\boldsymbol{D}^{(\texttt{p},\texttt{test})}$ by utilizing the testing dataset $\boldsymbol{D}^{\texttt{LR}}_{\texttt{tst}}$. We sample 100 prompts per element in $\boldsymbol{D}^{\texttt{LR}}_{\texttt{tst}}$ using $\hat{f}_{\texttt{PG}}(.,.)$. For each of this prompts the activation $\mathbf{A}_\iota$ can be extracted by using the $f_{\texttt{to}}(.)$ and LLM $\hat{p}(.\mid .)$). This activation $\mathbf{A}_\iota$ forms with the corresponding intermediate result $\mathbf{w}^{\texttt{IR}}$ a sample of $\boldsymbol{D}^{(\texttt{p},\texttt{test})}$ (i.e., $(\mathbf{A}_\iota, \mathbf{w}^{\texttt{IR}}) \in \boldsymbol{D}^{(\texttt{p},\texttt{test})}$). We therefore have a test set with $|\boldsymbol{D}^{(\texttt{p},\texttt{test})}| = 40{*}100 = 4000$ instances. $\boldsymbol{D}^{(\texttt{p},\texttt{test})}$ then can be used to compute the probing value:

$$r^{\texttt{prob}}_{\texttt{LR}} = f_{\texttt{prob}}(m_{\texttt{prob}}(.), \boldsymbol{D}^{(\texttt{p},\texttt{test})}) \tag{32}$$

This provides insights if the hidden representations for linear regression could be found in the activations $\mathbf{A}_\iota$ of layer ι. We can compute the probing value $r^{\texttt{prob}}_{\texttt{MD}}$ for the Manhattan distance problem similarly by using $\boldsymbol{D}^{\texttt{MD}}, \boldsymbol{D}^{\texttt{MD}}_{\texttt{trn}}$ and $\boldsymbol{D}^{\texttt{MD}}_{\texttt{tst}}$ instead of $\boldsymbol{D}^{\texttt{LR}}, \boldsymbol{D}^{\texttt{LR}}_{\texttt{trn}}$ and $\boldsymbol{D}^{\texttt{LR}}_{\texttt{tst}}$.

5.5.2 Differentiation Ability

In a second step, we investigate a model's ability to differentiate between different tasks. Given activations $\mathbf{A}_\iota$ of a layer $\iota \in \Omega_\iota$, we can compute the importance difference mapping by sampling first 1000 samples from $\boldsymbol{D}^{\texttt{LR}}$ and $\boldsymbol{D}^{\texttt{MD}}$ with replacement. We use those to create datasets as following:

$$\boldsymbol{D}^{\texttt{FA}}_\tau = \bigcup_{(\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}}) \in f_{\texttt{sample}}(1000, \boldsymbol{D}^\tau)} \left\{ (f_{\texttt{to}}(\overleftarrow{\boldsymbol{\omega}}), f_{\texttt{to}}(\overrightarrow{\boldsymbol{\omega}})) \right\}$$
$$\text{s.t.} \quad (\overleftarrow{\boldsymbol{\omega}}, \overrightarrow{\boldsymbol{\omega}}) = \hat{f}_{\texttt{PG}}((\mathbf{w}^{\texttt{IR}}, \boldsymbol{D}_{\mathbf{w}^{\texttt{IR}}}), \boldsymbol{\omega}^{\texttt{ctxt}}_\tau) \tag{33}$$

We create two datasets by using $\tau = \texttt{LR}$ for linear regression ($\boldsymbol{D}^{\texttt{FA}}_{\texttt{LR}}$) and $\tau = \texttt{MD}$ for the Manhattan distance ($\boldsymbol{D}^{\texttt{FA}}_{\texttt{MD}}$).

The importance difference values ξ for $\mathbf{A}_\iota$ of layer ι for LLM $\hat{p}(. \mid .)$ are then given through:

$$\xi = f_{\texttt{FAC}}(\hat{p}(. \mid .), \mathbf{A}_\iota, \boldsymbol{D}^{\texttt{FA}}_{\texttt{LR}}, \boldsymbol{D}^{\texttt{FA}}_{\texttt{MD}}) \tag{34}$$

Let us now introduce a feature filter function $f_{\texttt{filter}}(.,.,.,.)$. Lets assume we have features $\mathbf{g} \in \mathbb{R}^{d^{\texttt{feat}}}$ and some value $0 \leq h \leq 1$ and $\odot \in \{\leq, \geq\}$. We than can define for $1 \leq i \leq d^{\texttt{feat}}$:

$$f_{\texttt{filter}}(h, \odot, \xi, \mathbf{g})_i = \begin{cases} \mathbf{g}_i, \text{ if } h \odot \frac{\sum_{j=1}^{d^{\texttt{feat}}} \mathbb{1}(\xi_i \leq \xi_j)}{d^{\texttt{feat}}} \\ 0, \text{ otherwise} \end{cases} \tag{35}$$

Now assuming that ξ represents to some degree faithfully which features are more important for task 1 and task 2, we can use this formula to obtain additional probing results using only the most important features of the two tasks. We define this **filtered probing settings** with $r^{\texttt{prob}}_{(\tau', \odot, x)}$. Where τ' can be ether linear regression (LR) or Manhattan distance (MB), similar to $r^{\texttt{prob}}_{\texttt{LR}}$ or $r^{\texttt{prob}}_{\texttt{MD}}$ receptively (§5.5.1). But we use as input the filtered features given by $f_{\texttt{filter}}(x, \odot, \xi, y)$. $\odot$ can ether be $\leq$ or $\geq$ and we evaluate experiments for $x \in \{0.5, 0.25\}$ if $\odot = \geq$ and $x \in \{0.5, 0.75\}$ if $\odot = \leq$.

We then analyse this 8 additional probing results to investigate whether there exists an indicator that the model successfully is able to differentiate between tasks. If such an indicator exists in some layer, we would only get good (low) values for $r^{\texttt{prob}}_{(\texttt{LR},\geq,50)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,50)}$ but not for $r^{\texttt{prob}}_{(\texttt{LR},\leq,50)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\geq,50)}$. We also evaluate the behavior at $r^{\texttt{prob}}_{(\texttt{LR},\geq,25)}, r^{\texttt{prob}}_{(\texttt{LR},\leq,0.75)}, r^{\texttt{prob}}_{(\texttt{MD},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,0.75)}$ to determine if they shows consistent behavior to $r^{\texttt{prob}}_{(\texttt{LR},\geq,50)}, r^{\texttt{prob}}_{(\texttt{MD},\leq,50)}, r^{\texttt{prob}}_{(\texttt{LR},\leq,50)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\geq,50)}$. If the behavior for the 25% most important features for predictions of the two tasks differ, the 25% most important features may include another approach ignoring the intermediate results relevant for mathematical mechanisms.

5.6 Activation Patching

To apply activation patching on our LLM, we first sample for linear regression 500 prompts $\overleftarrow{\boldsymbol{\omega}}_i$ and results $\overrightarrow{\boldsymbol{\omega}}_i$ $(1 \leq i \leq 500)$ by using $\hat{f}_{\texttt{PG}}(.,.)$ on 500 random samples of $\boldsymbol{D}^{\texttt{LR}}$:

$$\boldsymbol{D}^{\texttt{LR}}_{\texttt{p}} = \{(\overleftarrow{\boldsymbol{\omega}}_i, \overrightarrow{\boldsymbol{\omega}}_i)\}_{i=1}^{500} \tag{36}$$

We sample a second such dataset as well:

$$\boldsymbol{D}^{\texttt{LR}*}_{\texttt{p}} = \{(\overleftarrow{\boldsymbol{\omega}}_i, \overrightarrow{\boldsymbol{\omega}}_i)\}_{i=1}^{500} \tag{37}$$

The same approach can be applied to the Manhattan Distance by using $\boldsymbol{D}^{\texttt{MD}}$ instead of $\boldsymbol{D}^{\texttt{LR}}$, resulting in $\boldsymbol{D}^{\texttt{MD}}_{\texttt{p}}$ and $\boldsymbol{D}^{\texttt{MD}*}_{\texttt{p}}$.

First lets define $\mathbf{a}[x]$ as x-th element of tuple $\mathbf{a}$. Now, to get our patching metric we first get the max token given $\boldsymbol{D}^*_{\tau_1}[i][0]$ with and without patching as $r^{\texttt{NP}}_i$ and $r^{\texttt{P}}_i$ respectively:

$$\begin{aligned} r^{\texttt{NP}}_i &= \texttt{argmax}_{\omega \in \Omega_{\texttt{voca}}}(\hat{p}(\omega \mid \boldsymbol{D}^*_{\tau_1}[i][0])) \\ r^{\texttt{P}}_i &= \texttt{argmax}_{\omega \in \Omega_{\texttt{voca}}}(\hat{p}(\omega \mid \boldsymbol{D}^*_{\tau_2}[i][0])_{\mathbf{A}^{\texttt{B}}_\iota \to \mathbf{A}^{\texttt{S}}_\iota}) \end{aligned} \tag{38}$$

Where $\hat{p}(\omega \mid \boldsymbol{D}^*_{\tau_2}[i][0])_{\mathbf{A}^{\texttt{B}}_\iota \to \mathbf{A}^{\texttt{S}}_\iota}$ is the patched model as described in §4.3, where we use the following $\mathbf{A}^{\texttt{B}}_\iota$ and $\mathbf{A}^{\texttt{S}}_\iota$:

$$\begin{aligned} \mathbf{A}^{\texttt{B}}_\iota &= f_{\texttt{filter}}(0.5, \odot, \xi, \mathbf{A}_\iota(\boldsymbol{D}^*_{\tau_2}[i][0])) \\ \mathbf{A}^{\texttt{S}}_\iota &= f_{\texttt{filter}}(0.5, \odot, \xi, \mathbf{A}_\iota(\boldsymbol{D}^*_{\tau_3}[i][0])) \end{aligned} \tag{39}$$

$\mathbf{A}_\iota(q)$ stands for the activations of layer ι from $\hat{p}(. \mid q)$. Our metric then can be defined as the proportion of overlap between the original and patched prediction:

$$\rho = \sum_{i=1}^{500} \mathbb{1}(r^{\texttt{NP}}_i == r^{\texttt{P}}_i) \tag{40}$$

We are now conducting experiments which we call $\rho_{(\tau_1,u,v)}$. We analyse 8 settings where τ_1 and u can be the linear regression (LR) or Manhattan distance (MD) problem and $v \in \{B, S\}$[9]. If τ_1 is LR, we use $\leq$ as $\odot$ and if τ_1 is MD, we use $\geq$ as $\odot$ in our patching experiments. If v is B we use $\tau_2 = \tau_1$ and $\tau_3 = u$.If q is S we use $\tau_3 = \tau_1$ $\tau_2 = u$.

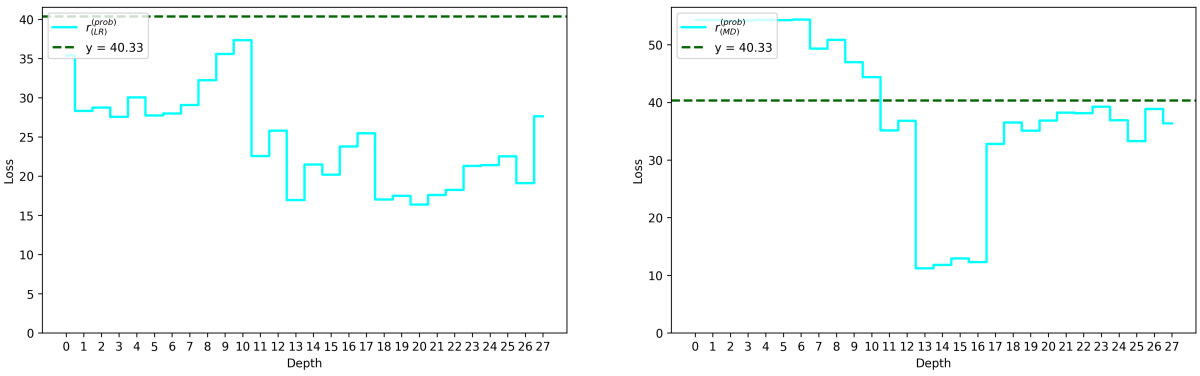
6 Results

In this section we present the results for Llama 3.2 3B, highlighting key findings from this evaluation. We additionally evaluated Llama 3.1 8B and present the results in App. C.

6.1 Probing

We present the results $r^{\texttt{prob}}_{\texttt{LR}}$ for the layers $\iota^{\texttt{k}}_i$ $(0 \leq i \leq 27)$ in Fig. 1. While other layer types, such as $\iota^{\texttt{v}}$, exhibit similar dynamics, $\iota^{\texttt{k}}$ is the most intriguing for the following result sections. The plots for

[9] B stands for our basic setting and S for our swapped setting (swapping source and base task in patching $r^{\texttt{P}}$).



(a) $r^{\texttt{prob}}_{\texttt{LR}}$ (b) $r^{\texttt{prob}}_{\texttt{MD}}$

Figure 1: $r^{\texttt{prob}}_{\texttt{LR}}$ and $r^{\texttt{prob}}_{\texttt{MD}}$ for Llama 3.2 3B for the layers $\iota^{\texttt{k}}_i$ ($0 \leq i \leq 27$). 44.33 is the value a model would take if it ether predicts randomly using an uniform distribution over range $[0 - 22]$ or the optimal constant prediction (11).

the other layer types can be seen on our github[10]. We observe that there are indications that the intermediate results for the Manhattan Distance (second unknown point) are approximated by the model in $\iota^{\texttt{k}}_i$ for $12 \leq i \leq 16$ (Fig. 1b). The results for regression are less clear. However, we see that the probing loss is lower compared to using a uniform random prediction. This suggests that the regression weights are approximated in the layers $\iota^{\texttt{k}}_i$ for $11 \leq i \leq 26$ (Fig. 1a).

Therefore, we observe that for both problems there exists indications that the intermediate results are at least approximated by some mechanism. This serves an indication that a mathematical mechanism exists ($\mathcal{H}_1$), as the intermediate results are computed and needed by mathematical mechanisms.

6.2 Differentiation Ability

We present the importance difference mapping ξ for the two different tasks for $\iota^{\texttt{k}}_i$ ($0 \leq i \leq 27$) in Fig. 2. One interesting observation is that we can clearly see the different attention heads. This suggests that different attention heads have different importances across different tasks.

In Fig. 3, we present the $r^{\texttt{prob}}_{(\texttt{LR},\geq,50)}$, $r^{\texttt{prob}}_{(\texttt{LR},\leq,50)}$, $r^{\texttt{prob}}_{(\texttt{MD},\geq,50)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,50)}$. A result supporting the hypothesis that the model is able to separate between different mathematical mechanisms ($\mathcal{H}_2$), would show up in a layer where the following 2 conditions hold:

$$\begin{aligned} r^{\texttt{prob}}_{(\texttt{LR},\geq,50)} &< r^{\texttt{prob}}_{(\texttt{LR},\leq,50)} \\ r^{\texttt{prob}}_{(\texttt{MD},\geq,50)} &> r^{\texttt{prob}}_{(\texttt{MD},\leq,50)} \end{aligned} \tag{41}$$

[10] https://github.com/GenK-ai/In-context-learning/

We can see that there exist multiple layers (especially layer $\iota^{\texttt{k}}_{16}$) fulfilling this conditions. While this results suggest that the model can decide to apply different mathematical methods to different problems, the indication is less conclusive than we had hoped for for linear regression. Recent work (Geiger et al., 2023b) suggests using transformed features/activations generated with a reversible transformation to split intermediate results, rather than the hidden representations as features. We also employed a hard threshold of 50%, leaving room for exploration of more optimal thresholds. We leave it for future work to propose more sophisticated methods that can identify different mechanisms for different tasks, without relying on the evaluated mechanisms being the major mechanisms.

6.3 Minor Mechanism

In Fig. 4 we present $r^{\texttt{prob}}_{(\texttt{LR},\geq,25)}$, $r^{\texttt{prob}}_{(\texttt{LR},\leq,0.75)}$, $r^{\texttt{prob}}_{(\texttt{MD},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,0.75)}$.

In Fig. 3b we observe that in layer $\iota^{\texttt{k}}_{17}$ contains approximation of the intermediate results. If we now look at the layer $\iota^{\texttt{k}}_{17}$ in Fig. 4b where we only use the 25% most important features for the Manhattan Distance problem (in regard to linear regression), we see that probing is less successful in finding the intermediate results. This is an indicator that given our combined feature importance mapping, the mathematical mechanism, while existing at $\iota^{\texttt{k}}_{17}$, does not belong to the major differing features of the Manhattan task. This is an indicator that the mathematical mechanism is a minor contributor to the output ($\mathcal{H}_3$). To a lesser extend, we can observe similar behaviour in $\iota^{\texttt{k}}_{22}$ in linear regression.

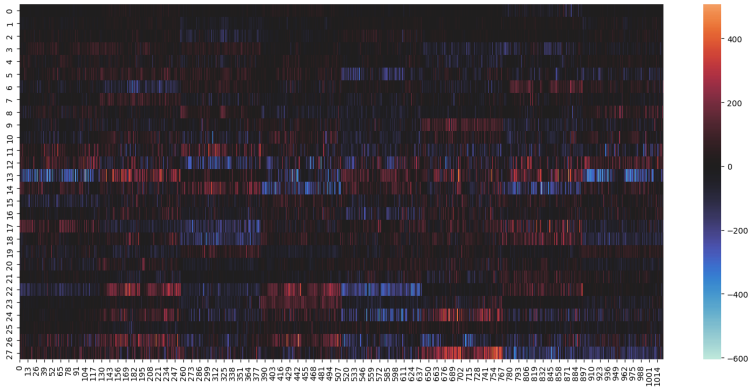
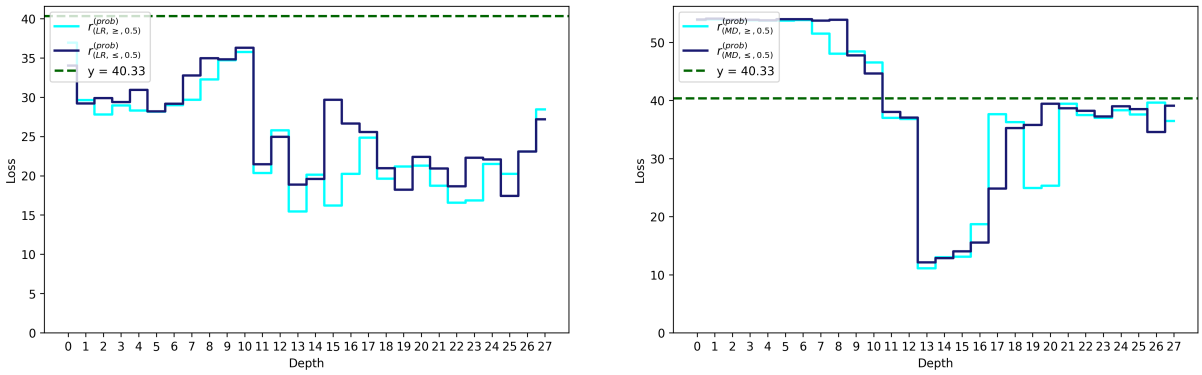


Figure 2: The combined feature importance mapping ξ for Llama 3.2 3B for linear regression as first task and Manhattan distance as second task for $\iota_i^{\mathtt{k}}$ $(0 \leq i \leq 27)$. The y-axis is the depth i and the x-axis are the features. Red indicates that a feature is more important for linear regression and blue indicates that a feature is more important for Manhattan distance.



(a) $r^{\mathtt{prob}}_{(\mathtt{LR},\geq,50)}$ and $r^{\mathtt{prob}}_{(\mathtt{LR},\leq,50)}$

(b) $r^{\mathtt{prob}}_{(\mathtt{MD},\geq,50)}$ and $r^{\mathtt{prob}}_{(\mathtt{MD},\leq,50)}$

Figure 3: $r^{\mathtt{prob}}_{(\mathtt{LR},\geq,50)}$, $r^{\mathtt{prob}}_{(\mathtt{LR},\leq,50)}$, $r^{\mathtt{prob}}_{(\mathtt{MD},\geq,50)}$ and $r^{\mathtt{prob}}_{(\mathtt{MD},\leq,50)}$ for Llama 3.2 3B for the layers $\iota_i^{\mathtt{k}}$ $(0 \leq i \leq 27)$. 44.33 is the value a model would take if it ether predicts randomly using an uniform distribution over range $[0 - 22]$ or the optimal constant prediction (11).

6.4 Activation Patching

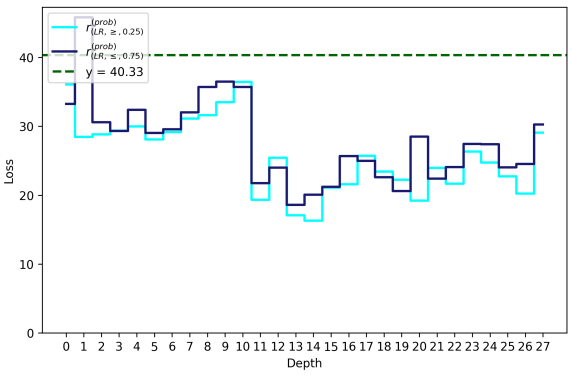
Our results are given in Fig. 5. We can see that $\rho_{(\mathtt{LR},\mathtt{LR},\mathtt{B})}$, $\rho_{(\mathtt{LR},\mathtt{MD},\mathtt{B})}$, $\rho_{(\mathtt{MD},\mathtt{MD},\mathtt{B})}$ and $\rho_{(\mathtt{MD},\mathtt{LR},\mathtt{B})}$ are significantly worse than 500 across all $\iota^{\mathtt{k}}$ layers. Therefore, we observe that the model exhibits a change in behaviour when employing patching, as expected.

We also see that $\rho_{(\mathtt{LR},\mathtt{LR},\mathtt{S})}$ is worse than $\rho_{(\mathtt{LR},\mathtt{LR},\mathtt{B})}$ and $\rho_{(\mathtt{MD},\mathtt{MD},\mathtt{S})}$ is worse than $\rho_{(\mathtt{MD},\mathtt{MD},\mathtt{B})}$ for all $\iota^{\mathtt{k}}$ layers. However, in an ideal scenario, the opposite should hold. This can be attributed to two possible causes. First, it could indicate that we need to find a better way to separate features according to their influence in tasks. However, given this result, it can also indicate that the major contributing mechanism is not captured by the combined importance metric ξ (highlighting differences between tasks) and therefore is for both tasks the same. Given the scope of our current work, it is left to future research to determine the influence of these two potential causes.

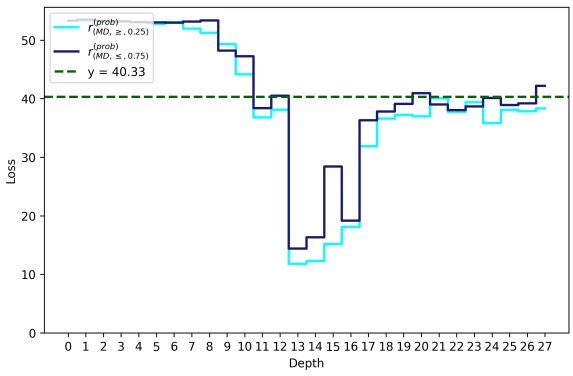
7 Conclusion

We find evidence supporting all three hypotheses presented in the introduction. While further research is necessary to solidify our conclusions, our results suggest that current LLMs incorporate mathematical methods into their operations. Additionally, we observe that LLMs exhibit mechanism selection dynamics on minor mathematical mechanisms analogous to those described in the literature. However, existing indications suggest that mathematical mechanisms may play a minor role in the model's predictions.

Our work provides insights that could help bridge the gap between research suggesting that

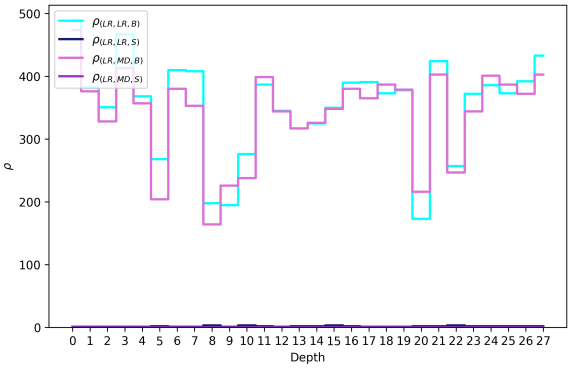


(a) $r^{\texttt{prob}}_{(\texttt{LR},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{LR},\leq,0.75)}$

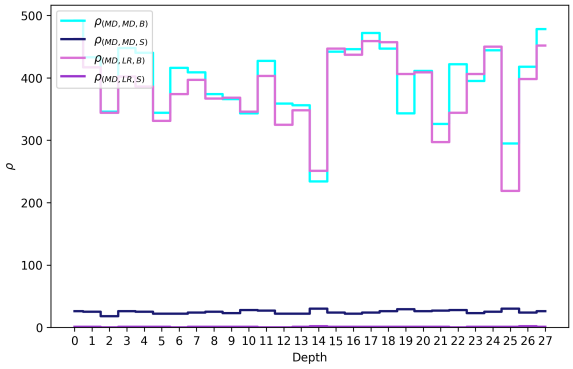


(b) $r^{\texttt{prob}}_{(\texttt{MD},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,0.75)}$

Figure 4: $r^{\texttt{prob}}_{(\texttt{LR},\geq,25)}, r^{\texttt{prob}}_{(\texttt{LR},\leq,0.75)}, r^{\texttt{prob}}_{(\texttt{MD},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,0.75)}$ for Llama 3.2 3B for the layers $\iota^{\texttt{k}}_i$ ($0 \leq i \leq 27$). 44.33 is the value a model would take if it ether predicts randomly using an uniform distribution over range $[0 - 22]$ or the optimal constant prediction (11).



(a) $\rho_{(\texttt{LR},\texttt{LR},\texttt{B})}, \rho_{(\texttt{LR},\texttt{LR},\texttt{S})}, \rho_{(\texttt{LR},\texttt{MD},\texttt{B})}, \rho_{(\texttt{LR},\texttt{MD},\texttt{S})}$



(b) $\rho_{(\texttt{MD},\texttt{LR},\texttt{B})}, \rho_{(\texttt{MD},\texttt{LR},\texttt{S})}, \rho_{(\texttt{MD},\texttt{MD},\texttt{B})}, \rho_{(\texttt{MD},\texttt{MD},\texttt{S})}$

Figure 5: The results for our activation patching experiments for LLama 3.2 3B.

mathematical methods can emerge in transformers and studies indicating that simpler mechanisms are the primary drivers of predictions. This presents two promising avenues for future research: (1) developing more sophisticated methods to obtain more conclusive results and better elucidate the role of minor algorithms, and (2) investigating training regimes that augment these minor algorithms' influence in LLMs to enhance performance on mathematical problems. We discuss in App. D some strategies for refining our pipeline.

8 Limitations

While the results are promising, some limitations must be noted. The feature importance evaluation is a simple method with known drawbacks (e.g. noise). Additionally, our feature assignment to different tasks exhibits noise, likely due to its simplicity. Consequently, we observe noisy results. While we have found indications supporting our three hypotheses, future experiments with more sophisticated methods are needed to identify and distinguish between minor and major mechanisms in an LLM.

Acknowledgments

We would like to thank Prof. Dr. Thomas Hofmann for making this project possible. We also thank OpenAI for providing free access to ChatGPT[11] and Meta for providing free access to Llama 3.2 3B and Llama 3.1 8B. ChatGPT and Llama 3.2 3B were used for correcting lexical and syntactical mistakes and improve the writing. We would like to extend our gratitude to Urban Moser and Leo Schefer for their technical support, as well as the Data Analytics Lab at ETH for providing access to their cluster.

[11] https://chat.openai.com/

References

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. 2024. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 135–168. PMLR.

Isaac Ahern, Adam Noack, Luis Guzman-Nateras, Dejing Dou, Boyang Albert Li, and Jun Huan. 2019. Normlime: A new feature importance metric for explaining deep neural networks. *ArXiv*, abs/1909.04200.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *ArXiv*, abs/2211.15661.

Yu Bai, Fan Chen, Haiquan Wang, Caiming Xiong, and Song Mei. 2023. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *ArXiv*, abs/2306.04637.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Bap tiste 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 Cantón 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 A. AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriele Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guanglong Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Laurens 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, Ju-Qing Jia, Kalyan Vasuden Alwala, K. Upasani, Kate Plawiak, Keqian Li, Ken-591 neth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuen ley 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 Babu Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melissa Hall Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri S. Chatterji, Olivier Duchenne, Onur cCelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasić, 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, Ro main 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 Chandra 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, Whit ney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yiqian Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zhengxu Yan, Zhengxing Chen, Zoe Papakipos, Aaditya K. Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adi 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, Ben Leonhardi, Po-Yao (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, Shang-Wen 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 Guzm'an, Frank J. Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory G. Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Han Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, 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, Kaixing(Kai) Wu, U KamHou, 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, A Lavender, 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 Dollár, 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, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sung-Bae 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, Vlad Ionescu, Vlad Andrei Poenaru, Vlad T. Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xia 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. *ArXiv*, abs/2407.21783.

Jiahai Feng, Stuart Russell, and Jacob Steinhardt. 2025. Extractive structures learned in pretraining enable generalization on finetuned facts. *Preprint*, arXiv:2412.04614.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah D. Goodman, Christopher Potts, and Thomas F. Icard. 2023a. Causal abstraction: A theoretical foundation for mechanistic interpretability.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas F. Icard, and Noah D. Goodman. 2023b. Finding alignments between interpretable causal variables and distributed neural representations. *ArXiv*, abs/2303.02536.

Nicholas W. Goldowsky-Dill, Chris MacLeod, Lucas Jun Koba Sato, and Aryaman Arora. 2023. Localizing model behavior with path patching. *ArXiv*, abs/2304.05969.

Michael Hahn and Navin Goyal. 2023. A theory of emergent in-context learning as implicit structure induction. *ArXiv*, abs/2303.07971.

Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. 2023. Explaining emergent in-context learning as kernel regression. *Preprint*, arXiv:2305.12766.

Seungwook Han, Jinyeop Song, Jeff Gore, and Pulkit Agrawal. 2024. Emergence of abstractions: Concept encoding and decoding mechanism for in-context learning in transformers. *ArXiv*, abs/2412.12276.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *ArXiv*, abs/2403.17806.

Beomsu Kim, Junghoon Seo, Seunghyeon Jeon, Jamyoung Koo, Jeongyeol Choe, and Taegyun Jeon. 2019. Why are saliency maps noisy? cause of and solution to noisy saliency maps. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4149–4157.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Andrew Kyle Lampinen, Stephanie C. Y. Chan, Aaditya K. Singh, and Murray Shanahan. 2024. The broader spectrum of in-context learning. *ArXiv*, abs/2412.03782.

Michael A. Lepori, Thomas Serre, and Ellie Pavlick. 2023. Uncovering intermediate variables in transformers using circuit probing. *ArXiv*, abs/2311.04354.

Jiaoda Li, Yifan Hou, Mrinmaya Sachan, and Ryan Cotterell. 2024. What do language models learn in context? The structured task hypothesis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12365–12379, Bangkok, Thailand. Association for Computational Linguistics.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *ArXiv*, abs/2410.21272.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova Dassarma, Tom Henighan, Benjamin Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, John Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom B. Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Christopher Olah. 2022. In-context learning and induction heads. *ArXiv*, abs/2209.11895.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319, Toronto, Canada. Association for Computational Linguistics.

Core Francisco Park, Ekdeep Singh Lubana, Itamar Pres, and Hidenori Tanaka. 2024. Competition dynamics shape algorithmic phases of in-context learning. *ArXiv*, abs/2412.01003.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. 2020. Visualizing the impact of feature attribution baselines. *Distill*. Https://distill.pub/2020/attribution-baselines.

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*, ICML'17, page 3319–3328. JMLR.org.

Curt Tigges, Michael Hanna, Qinan Yu, and Stella Biderman. 2024. Llm circuit analyses are consistent across training and scale. *Preprint*, arXiv:2407.10827.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. 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.

Yongjie Wang, Tong Zhang, Xu Guo, and Zhiqi Shen. 2024. Gradient based feature attribution in explainable ai: A technical review. *ArXiv*, abs/2403.10415.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah D. Goodman. 2024. Interpretability at scale: identifying causal mechanisms in alpaca. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Ruo Yang, Binghui Wang, and Mustafa Bilgic. 2023. Idgi: A framework to eliminate explanation noise from integrated gradients. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23725–23734.

Wei Jie Yeo, Ranjan Satapathy, and Erik Cambria. 2024. Towards faithful natural language explanations: A study using activation patching in large language models. *ArXiv*, abs/2410.14155.

Zeping Yu and Sophia Ananiadou. 2024. How do large language models learn in-context? query and key matrices of in-context heads are two towers for metric learning. *EMNLP*.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. 2023. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *ArXiv*, abs/2305.19420.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. 2024. The clock and the pizza: two stories in mechanistic explanation of neural networks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

A Feature Attribution

In our work we used the salency maps for feature attribution. However, salency maps have been shown to be prone to noisy attributions (Kim et al., 2019). In preliminary experiments we additionally tried Gradient×Input (App. A.3), SmoothGrad (App. A.2) and Expected Gradients (App. A.5), however they did not perform recognizably better given our setting. There also exist other methods (e.g. App. A.6 and App. A.7). We note that using salency maps may not be the optimal feature attribution method, but a comprehensive evaluation of alternative methods is beyond the scope of this work.

A.1 Vanilla Gradient

As discussed, salency maps uses the gradient of a feature $\mathbf{A}_\iota$ (given some layer $\iota \in \Omega_\iota$) of the output logit $\hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})$ given the true expected token $\overrightarrow{\boldsymbol{\omega}}$:

$$\frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})}{\partial \mathbf{A}_\iota} \tag{42}$$

A.2 SmoothGrad

SmoothGrad was proposed by (Smilkov et al., 2017). It gets its feature attribution maps similarly to salency maps (compare App. A.1). Smooth Grad improves upon its predecessor by perturbing the feature $\mathbf{A}_\iota$ in the forward pass K times with some noise sampled from a normal distribution $\varepsilon_i \sim \mathcal{N}(0, 0.1 * std(\mathbf{A}_\iota))$ $(1 \leq i \leq K)$:

$$\mathbf{A}'_{\iota,i} = \mathbf{A}_\iota + \varepsilon_i \tag{43}$$

Let $\hat{p}(. \mid \overleftarrow{\boldsymbol{\omega}})'_i$ be the large language model where the forward run $\mathbf{A}_\iota$ is perturbed by ε_i. Smooth grad then can be written as:

$$\frac{1}{K} \sum_{i=1}^{k} \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})'_i}{\partial \mathbf{A}'_{\iota,i}} \tag{44}$$

A.3 Input×Gradient

Another simple method suggested in the literature to improve upon the simple gradient feature attribution (compare App. A.1) is the Input×Gradient method (Shrikumar et al., 2017). This method involves multiplying the gradient by the input, effectively scaling the gradient by the input values:

$$\mathbf{A}_\iota \odot \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})}{\partial \mathbf{A}_\iota} \tag{45}$$

where $\odot$ is a element-wise multiplication.

A.4 Integrated Gradient

Integrated gradient (Sundararajan et al., 2017) computes the feature attribution for the feature $\mathbf{A}_\iota$ (given some layer $\iota \in \Omega_\iota$) by integrating over gradients. To do this, we first define the features of a baseline $\mathbf{A}^*_\iota$. This baseline has the same dimensions as the features $\mathbf{A}_\iota$ and should represent intuitively the absence of interesting features (e.g. black image in vision context). Lets now define

$$\mathbf{A}^*_{\iota,x} = (1 - x) * \mathbf{A}^*_\iota + x * \mathbf{A}_\iota \tag{46}$$

for $0 \leq x \leq 1$ and $\hat{p}(. \mid \overleftarrow{\boldsymbol{\omega}})^*_x$ as the LLM where we exchange the activation $\mathbf{A}_\iota$ with $\mathbf{A}^*_{\iota,x}$ in the forward pass. Lets say the true prediction is the token at position $\overrightarrow{\boldsymbol{\omega}}$. Integrated Gradient feature attribution can then be defined as follows:

$$(\mathbf{A}_\iota - \mathbf{A}^*_\iota) \int_0^1 \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_x}{\partial \mathbf{A}^*_{\iota,x}} dx \tag{47}$$

A.5 Expected Gradients

Expected Gradients, proposed by Sturmfels et al. (2020), builds upon on the integrated gradient method presented in App. A.4.

Expected Gradient now suggests that rather than using a single baseline vector $x^{(b)}$, it is more effective to integrate over a distribution of baseline vectors Ξ. This allows mitigating problems like colour blindness in integrated gradients. Colour blindness results in integrated gradients assigning 0 importance to black pixels when using a black

picture as baseline (Sturmfels et al., 2020). Therefore, Expected Gradient is written as follows:

$$w_j = \mathbb{E}_{\mathbf{A}^*_\iota \in \Xi}\left[(\mathbf{A}_\iota - \mathbf{A}^*_\iota)\int_0^1 \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_x}{\partial \mathbf{A}^*_{\iota,x}} dx\right] \quad (48)$$

Approximated by Riemann integration it can be written as:

$$\frac{1}{M}\sum_{m=1}^{M}(\mathbf{A}_\iota - \mathbf{A}^{(m)}_\iota)\frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^{(m)}}{\partial \mathbf{A}^{(m)}_\iota} \quad (49)$$

Where $\mathbf{A}^{(m)}_\iota$ is the m-th sampled baseline from the distribution Ξ. $\hat{p}(. \mid \overleftarrow{\boldsymbol{\omega}})^{(m)}$ is the langauge model where in the forward pass $\mathbf{A}_\iota$ is exchanged with $\mathbf{A}^{(m)}_\iota$ and $\overrightarrow{\boldsymbol{\omega}}$ is the the expected true output token. It has to be noted that in the case where $\mathbf{A}_\iota - \mathbf{A}^{(m)}_\iota = 1$ and the distribution Ξ is set to a Gaussian with mean $\mathbf{A}_\iota$ and standard deviation $0.1 * std(\mathbf{A}_\iota)$, we retrieve SmoothGrad (Sturmfels et al., 2020).

Sturmfels et al. (2020) observed that samples from the training or testing sets serve as a proxy for Ξ. In our preliminary experiments, we use the features generated from samples of the test sets of both tasks as baselines.

A.6 Important Direction Gradient Integration

Important Direction Gradient Integration was proposed by Yang et al. (2023). It is a method to improve integrated gradient methods like Expected Gradients (Sturmfels et al., 2020). It is based on insights gained from the analysis of the Riemann integration used by implementations of integrated gradient methods. The Riemann integration in the Integrated Gradient context works as follows (using notation from App. A.5):

$$\int_0^1 \boldsymbol{\Delta}_x dx \approx \sum_{t=1}^{T-1}(\mathbf{A}^*_{\iota,(\frac{t+1}{T})} - \mathbf{A}^*_{\iota,(\frac{t}{T})}) * \boldsymbol{\Delta}_{\frac{t}{T}}$$
$$\boldsymbol{\Delta}_y = \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_y}{\partial \mathbf{A}^*_{\iota,y}} \quad (50)$$

They state that in Riemann integration over a path there exists both an important direction and a noise direction in $(\mathbf{A}^*_{\iota,(\frac{t+1}{T})} - \mathbf{A}^*_{\iota,(\frac{t}{T})})$. Where the noise direction should have no influence on the result of the integral, but it can introduce noise in the Riemann integration. Therefore they suggest replacing

$$(\mathbf{A}^*_{\iota,(\frac{t+1}{T})} - \mathbf{A}^*_{\iota,(\frac{t}{T})}) * \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_{(\frac{t}{T})}}{\partial \mathbf{A}^*_{\iota,(\frac{t}{T})}} \quad (51)$$

in Riemann integration, with:

$$\frac{g_j * g_j * d}{< g, g >} \quad (52)$$

where $< ., . >$ denotes the dot product and:

$$g = \frac{\partial \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_{(\frac{t}{T})}}{\partial \mathbf{A}_\iota} \quad (53)$$

$$d = \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_{(\frac{t+1}{T})} - \hat{p}(\overrightarrow{\boldsymbol{\omega}} \mid \overleftarrow{\boldsymbol{\omega}})^*_{(\frac{t}{T})} \quad (54)$$

A.7 Further methods

There are several other methods used for feature attribution. Some additional methods for feature attribution are discussed in Wang et al. (2024). However, we do not discuss other methods here, as most of those not mentioned are computationally more expensive and thus beyond our capabilities.

B Distributed Alignment Search (DAS)

B.1 Method

Distributed Alignment Search (DAS) was proposed by Geiger et al. (2023b). We leave a detailed description to Geiger et al. (2023b). For our statement it is only necessary to know that DAS employs a back-propagation based approach to identify transformed features corresponding to intermediate results which are responsible for the prediction. They try to find as a transformation a rotation of the features, such that a small subset of the transformed features corresponds to an intermediate result (e.g. weights of regression). While their method is more sophisticated and adaptable in finding intermediate results within the activations of a layer in an LLM, it also relies on these intermediate results being part of the major mechanism in the model. Although this is intentional for their research question, it makes DAS unsuitable for our own research.

Wu et al. (2024) further adapted DAS (boundless DAS) to make it useful for larger models by improving the remaining brute-force search (the number of transformed features representing intermediate results) by using gradient descent. We will conduct two experiments with boundless DAS to show that the method is not suited for our research.

B.2 Results

We adapted the code Wu et al. (2024) to examine layer $\iota^{\mathtt{k}}_{16}$. We further restricted our experiments on the last token. We acknowledge that conducting a more comprehensive experiment (all layers and all tokens), as suggested by Wu et al. (2024), would be ideal, but due to computational constraints and this being only supplementary experiments, we opted to present the results obtained from this restricted experiments. The computation time was approximately two days for our implementation, given Llama 3.2 3B, one token, one layer, three training epochs and (the LLM distributed on) two NVIDIA RTX A5000.

We achieve an accuracy of 0.18 for the linear regression problem, which is deemed insufficient for using boundless DAS Wu et al. (2024). The accuracy applying boundless DAS intervention is 0.04, which is not insightful, given that boundless DAS in Wu et al. (2024) resulted in over 50% of the original task performance in the last evaluated token for layers, which included the intermediate results to some extent.

The same can be seen for the Manhattan distance problem with an accuracy of only 0.28 and 0.06 accuracy using DAS intervention.

C Results Llama-3.1 8B

In this section we present the results for Llama-3.1 8B [12].

C.1 Probing

In Fig. 6 we can see that Llama 3.1 8B also approximates the intermediate results.

C.2 Differentiation Ability

We present the combined feature importance mapping in Fig. 7. We can again observe that the different heads of the multi-head attention mechanism can be identified.

Fig. 8 presents $r^{\mathtt{prob}}_{(\mathtt{LR},\geq,50)}, r^{\mathtt{prob}}_{(\mathtt{LR},\leq,50)}, r^{\mathtt{prob}}_{(\mathtt{MD},\geq,50)}$ and $r^{\mathtt{prob}}_{(\mathtt{MD},\leq,50)}$ for Llama 3.1 8B. We can again observe some layers satisfying Eq. (41). We highlight the layer $\iota^{\mathtt{k}}_{15}$ as a notable example that satisfies the equations.

C.3 Minor Mechanism

We present the results for $r^{\mathtt{prob}}_{(\mathtt{LR},\geq,25)}, r^{\mathtt{prob}}_{(\mathtt{LR},\leq,0.75)}, r^{\mathtt{prob}}_{(\mathtt{MD},\geq,25)}$ and $r^{\mathtt{prob}}_{(\mathtt{MD},\leq,0.75)}$ for Llama 3.1 8B in Fig. 9 and would like to point out layer $\iota^{\mathtt{k}}_{17}$ for the linear regression problem and $\iota^{\mathtt{k}}_{18}$ for the Manhattan distance problem. Both of these layers demonstrate a good approximation of the intermediate results for the 50% probing, whereas the 25% top features probing results in a worse approximation.

C.4 Patching

In Fig. 10 we present the activation patching results for Llama 3.1 8B. We observe that Llama 3.1 8B behaves similarly to Llama 3.2 3B. We further see that at layer $\iota^{\mathtt{k}}_{15}$ we get a larger change in predictions than its neighbouring layers. This indicates that the mechanisms differing in features have more influence over the prediction than in its neighbouring layers. This concordance suggests that the LLM puts more focus on separate mechanisms for different problems in layer $\iota^{\mathtt{k}}_{15}$ than in its neighbouring layers.

D Future Work

We provide a solid foundation for the questions we investigated in this work. However, there is still much to be done to provide more conclusive evidence for the hypotheses mentioned in the introduction.

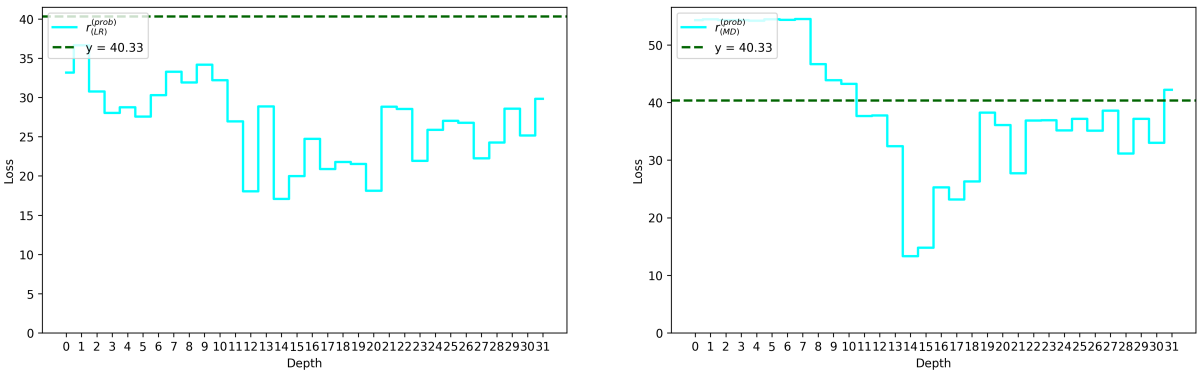
D.1 Probing

To the best of our knowledge is probing the best method to find minor mechanisms in a LLM. More sophisticated methods that we are aware of rely on the evaluated mechanism being the major mechanism. However, the training for our probing was not fine-tuned with respect to learning rate and number of training steps. We further use a small batch size of 1, which can lead to training instabilities. This shortcomings are mainly caused by our restricted time-frame and limited GPU memory. We therefore suggest for future research a second version of this work that evaluates minor mechanisms in LLMs with dynamic training steps, early stopping, larger batch sizes, and learning rate tuning.

D.2 Differentiation Ability

Our experiments on differentiation ability uses feature attribution by combining plain gradients to create combined feature importance values. Given our time-frame, we were unable to conduct an extensive analysis of feature attribution methods and importance combination calculations.

[12] https://huggingface.co/meta-llama/Llama-3.1-8B



(a) $r^{\texttt{prob}}_{\texttt{LR}}$ (b) $r^{\texttt{prob}}_{\texttt{MD}}$

Figure 6: $r^{\texttt{prob}}_{\texttt{LR}}$ and $r^{\texttt{prob}}_{\texttt{MD}}$ for Llama 3.1 8B for the layers $\iota^{\texttt{k}}_i$ ($0 \leq i \leq 27$). 44.33 is the value a model would take if it ether predicts randomly using an uniform distribution over range $[0 - 22]$ or the optimal constant prediction (11).

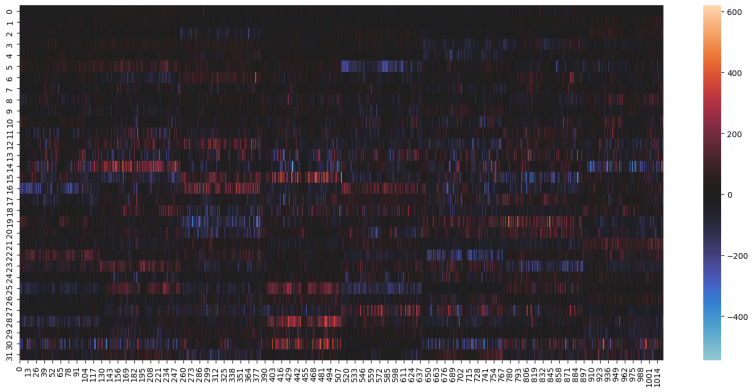


Figure 7: The combined feature importance mapping ξ for Llama 3.1 8B for linear regression as first task and Manhattan distance as second task for $\iota^{\texttt{k}}_i$ ($0 \leq i \leq 27$). The y-axis is the depth i and the x-axis are the features. Red indicates that a feature is more important for linear regression and blue indicates that a feature is more important for Manhattan distance.

While better feature attribution methods and importance combination calculations could improve the clarity of our results, we suspect that using a reversible transformation of the features, would result in even better results. Recent research employs this kind of approach (Wu et al., 2024). However, such methods rely on the mechanism under inspection being the major contributor to predictions. We are not aware of a method that does not rely on the evaluated mechanism being the major contributor to predictions. It is likely that a mechanism tailored to our setting needs to be developed.

D.3 Minor Mechanism

This analysis also suffers from the noise caused by problems mentioned in App. D.2. Therefore, this step must be repeated with an improved version of the step in App. D.2.

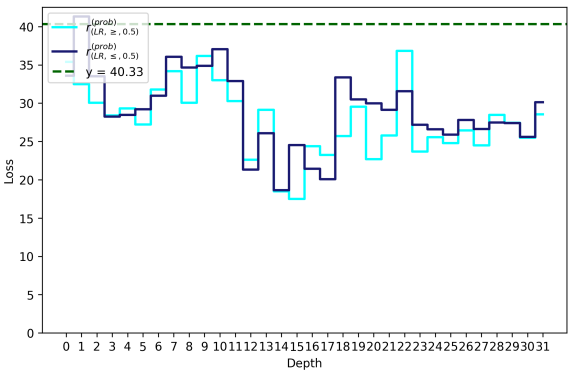
D.4 Patching

This step suffers from the same problems as App. D.3. However, given the results, it is additionally possible that LLMs utilise a major contributor to the prediction, which is ether similar or the same for both problems. The analysis of this new hypothesis may shed additional light on the workings of LLMs given mathematical problems.

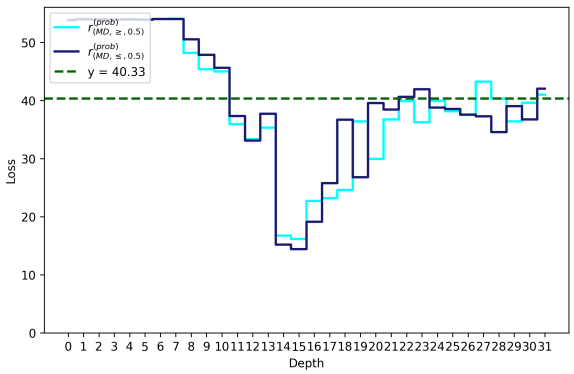
D.5 Additional Suggestions

Our experiments were conducted on Llama 3.2 3B and Llama 3.1 8B, which at the time of the experiments were the most popular free available methods with a manageable parameter size. There is no guarantee our hypotheses will generalise to smaller and larger models. Therefore, we recommend extending the experiments to other LLMs once the previously discussed problems are resolved.

It further would be interesting to experiment with the Linear Interpolation of Algorithms (LIA)

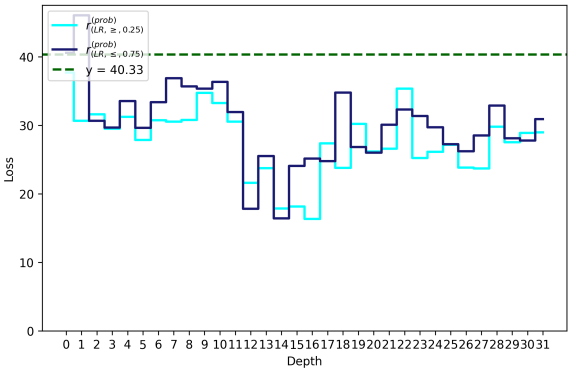


(a) $r^{\texttt{prob}}_{(\texttt{LR},\geq,50)}$ and $r^{\texttt{prob}}_{(\texttt{LR},\leq,50)}$

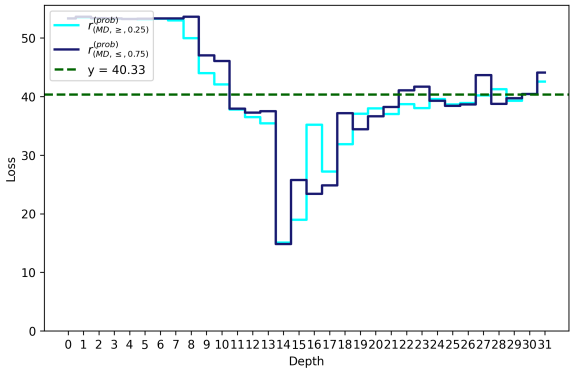


(b) $r^{\texttt{prob}}_{(\texttt{MD},\geq,50)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,50)}$

Figure 8: $r^{\texttt{prob}}_{(\texttt{LR},\geq,50)}, r^{\texttt{prob}}_{(\texttt{LR},\leq,50)}, r^{\texttt{prob}}_{(\texttt{MD},\geq,50)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,50)}$ for Llama 3.1 8B for the layers $\iota^{\texttt{k}}_i$ $(0 \leq i \leq 27)$. 44.33 is the value a model would take if it ether predicts randomly using an uniform distribution over range $[0 - 22]$ or the optimal constant prediction (11).



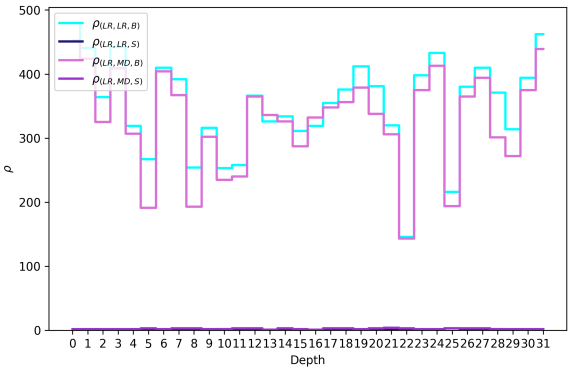
(a) $r^{\texttt{prob}}_{(\texttt{LR},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{LR},\leq,0.75)}$



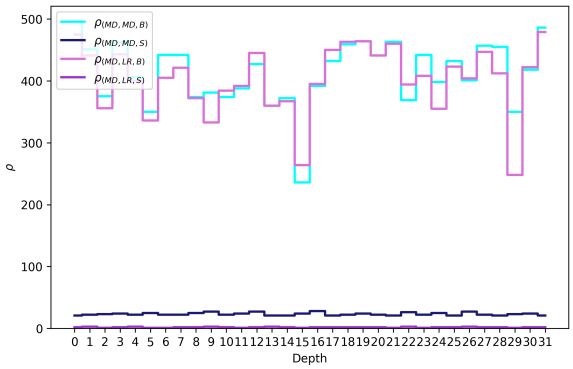
(b) $r^{\texttt{prob}}_{(\texttt{MD},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,0.75)}$

Figure 9: $r^{\texttt{prob}}_{(\texttt{LR},\geq,25)}, r^{\texttt{prob}}_{(\texttt{LR},\leq,0.75)}, r^{\texttt{prob}}_{(\texttt{MD},\geq,25)}$ and $r^{\texttt{prob}}_{(\texttt{MD},\leq,0.75)}$ for Llama 3.1 8B for the layers $\iota^{\texttt{k}}_i$ $(0 \leq i \leq 27)$. 44.33 is the value a model would take if it ether predicts randomly using an uniform distribution over range $[0 - 22]$ or the optimal constant prediction (11).

method proposed by Park et al. (2024). This would allow us to explore another possibility for investigating the influence of different mechanisms and their contributions. Such an approach could allow us to get additional insight, if mathematical mechanisms exist ($\mathcal{H}_1$) in LLMs and to which degree they do ($\mathcal{H}_3$). We discovered Park et al. (2024) at the end of our work and unfortunately were unable to incorporate their method into our experiments.



(a) $\rho_{(\texttt{LR},\texttt{LR},\texttt{B})}$, $\rho_{(\texttt{LR},\texttt{LR},\texttt{S})}$, $\rho_{(\texttt{LR},\texttt{MD},\texttt{B})}$, $\rho_{(\texttt{LR},\texttt{MD},\texttt{S})}$



(b) $\rho_{(\texttt{MD},\texttt{LR},\texttt{B})}$, $\rho_{(\texttt{MD},\texttt{LR},\texttt{S})}$, $\rho_{(\texttt{MD},\texttt{MD},\texttt{B})}$, $\rho_{(\texttt{MD},\texttt{MD},\texttt{S})}$

Figure 10: The results for our activation patching experiments for LLama 3.1 8B.

E Model Architectures

E.1 meta-llama/Llama-3.2-3B

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(128256, 3072)
    (layers): ModuleList(
      (0-27): 28 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): Linear(in_features=3072, out_features=3072, bias=False)
          (k_proj): Linear(in_features=3072, out_features=1024, bias=False)
          (v_proj): Linear(in_features=3072, out_features=1024, bias=False)
          (o_proj): Linear(in_features=3072, out_features=3072, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=3072, out_features=8192, bias=False)
          (up_proj): Linear(in_features=3072, out_features=8192, bias=False)
          (down_proj): Linear(in_features=8192, out_features=3072, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm((3072,), eps=1e-05)
        (post_attention_layernorm): LlamaRMSNorm((3072,), eps=1e-05)
      )
    )
    (norm): LlamaRMSNorm((3072,), eps=1e-05)
    (rotary_emb): LlamaRotaryEmbedding()
  )
  (lm_head): Linear(in_features=3072, out_features=128256, bias=False)
)
```

E.2 meta-llama/Llama-3.1-8B

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(128256, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (up_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (down_proj): Linear(in_features=14336, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
        (post_attention_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
```

```
        )
      )
      (norm): LlamaRMSNorm((4096,), eps=1e-05)
      (rotary_emb): LlamaRotaryEmbedding()
    )
    (lm_head): Linear(in_features=4096, out_features=128256, bias=False)
)
```