Backward Lens: Projecting Language Model Gradients into the Vocabulary Space

Shahar Katz[1,2] **Yonatan Belinkov**[1] **Mor Geva**[2] **Lior Wolf**[2]
[1]Faculty of Computer Science, Technion – Israel Institute of Technology
[2]Blavatnik School of Computer Science, Tel Aviv University
`{shaharkatz3@mail,morgeva@tauex,wolf@cs}.tau.ac.il, belinkov@technion.ac.il`

Abstract

Understanding how Transformer-based Language Models (LMs) learn and recall information is a key goal of the deep learning community. Recent interpretability methods project weights and hidden states obtained from the forward pass to the models' vocabularies, helping to uncover how information flows within LMs. In this work, we extend this methodology to LMs' backward pass and gradients. We first prove that a gradient matrix can be cast as a low-rank linear combination of its forward and backward passes' inputs. We then develop methods to project these gradients into vocabulary items and explore the mechanics of how new information is stored in the LMs' neurons. Our code is available at: `https://github.com/shacharKZ/BackwardLens`.

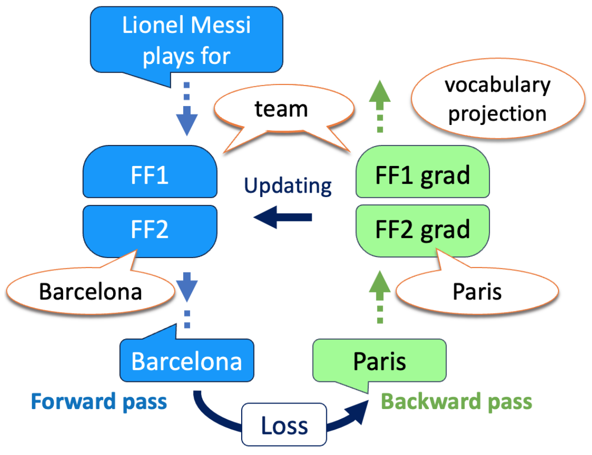


Figure 1: An illustration depicting the tokens promoted by a single LM's MLP layer and its gradient during the forward and backward pass when editing the model to answer "Paris" for the prompt "Lionel Messi plays for". The gradients (in green) of the first MLP matrix, FF_1, attempt to imprint into the model's weight (in blue) the information that FF_1 encountered during the forward pass. Utilizing a vocabulary projection method, we reveal that this information represents the token "team". The gradients of the second MLP matrix, FF_2, aim to shift the information encoded within FF_2 towards the embedding of the new target.

1 Introduction

Deep learning models consist of layers, which are parameterized by matrices that are trained using a method known as backpropagation. This process involves the creation of gradient matrices that are used to update the models' layers. Backpropagation has been playing a major role in interpreting deep learning models and multiple lines of study aggregate the gradients to provide explainability (Simonyan et al., 2014; Sanyal and Ren, 2021; Chefer et al., 2022; Sarti et al., 2023; Miglani et al., 2023).

Recent interpretability works have introduced methods to project the weights and intermediate activations of Transformer-based LMs (Vaswani et al., 2017) into the vocabulary space. The seminal "Logit Lens" method (nostalgebraist, 2020) has paved the way to explaining LMs' behavior during inference (Geva et al., 2022a; Dar et al., 2022; Ram et al., 2023), including directly interpreting individual neurons (Geva et al., 2021; Katz and Belinkov, 2023). Our work is the first, as far as we can ascertain, to project LM *gradients* to the vocabulary space. Furthermore, modern LMs contain thousands of neurons in each layer, while certain features are likely distributed across multiple neurons (Elhage et al., 2022; Cunningham et al., 2023). These issues are handled in our examination of the gradient matrices by performing a decomposition of provably low-rank matrices.

Despite the popularity of LMs, our understanding of their behavior remains incomplete (Bender et al., 2021; Dwivedi et al., 2023), particularly regarding how LMs acquire new knowledge during training and the mechanisms by which they store and recall it (Dai et al., 2022; Geva et al., 2021, 2023; Meng et al., 2023). We identify a mechanism we refer to as "imprint and shift", which captures how information is stored in the feed-forward

(MLP) module of the transformer layer. This module has two fully connected layers, FF_1 and FF_2. The "imprint" refers to the first layer, to or from which the learning process adds or subtracts copies of the intermediate inputs encountered during the forward pass. The "shift" refers to the second matrix, where the weights are shifted by the embedding of the target token; see Figure 1.

In summary, our contributions are: (i) Analyzing the rank of gradients. (ii) Interpreting gradients by inspecting relatively small spanning sets. In particular, we examine the Vector-Jacobian Product (VJP) obtained during the backward pass, which is the equivalent of the hidden states of the forward pass. (iii) Investigating the embedding of these sets by projecting them into tokens. (iv) Revealing a two-phase mechanism by which models learn to store knowledge in their MLPs, which we termed "imprint and shift". (v) Exploring a novel editing method based solely on a single forward pass.

2 Related Work

Developing methods to explain LMs is central to the interpretability community (Belinkov and Glass, 2019; Srivastava et al., 2023). Initially inspired by interpretability efforts in vision models (Sundararajan et al., 2017; Samek et al., 2017; Zhang and Zhu, 2018; Indolia et al., 2018; Olah et al., 2020), LMs have also benefited from the ability to operate in the language domain. This includes leveraging projections of vectors into readable concepts (nostalgebraist, 2020; Simhi and Markovitch, 2023) or clustering them into the idioms they promote (Cunningham et al., 2023; Tamkin et al., 2023; Tigges et al., 2023; Bricken et al., 2023).

Reverse engineering the gradient's role in shifting model behavior has been a primary method to comprehend the mechanics of deep learning models. Recent work (Ilharco et al., 2022; Gueta et al., 2023; Tian et al., 2023) demonstrates that clustering the weights that models learn during training or fine-tuning reveals patterns that connect the tasks and their training data. Other approaches employing Saliency Maps (Simonyan et al., 2014) explore the relationship between a gradient matrix and parts of its corresponding forward pass input, where we have observed that gradients are spans, linear combinations, of those inputs. Our work differs from previous studies in two key aspects: (1) we interpret gradients based on the components that generate them, primarily the VJPs, rather than analyzing entire gradient matrices, and (2) we elucidate the information stored in the gradients and project them into tokens, instead of solely examining their impact on the model's predictions or their connection to the training data. To the best of our knowledge, we are the first to explore both projecting gradients into tokens and examining them through VJPs, offering a perspective that has yet to be explored.

Our experiment regarding LM editing adds to the line of work that utilizes interpretability for knowledge editing. The closest idea to our implementation was introduced by Dai et al. (2022), who identified activated neurons for specific idioms in encoder LMs and altered them by injecting the embedded target. We show that gradients work in a very similar way. Other state-of-the-art model editing methods include Mitchell et al. (2021) and Meng et al. (2022, 2023).

Our work analyzes the gradient's rank. It was previously known that gradients are low-rank (Mitchell et al., 2021), but the utilization of this characteristic for interpretability study or predicting the rank of an edited prompt have remained unexplored. Optimizers and adaptors, such as LoRA (Hu et al., 2022), were created to constrain the rank of gradients, making fine-tuning faster. In contrast, our work demonstrates that the ranks of gradients can be predicted based on the edited prompt length.

3 Background

We first provide background on transformers, focusing on the components that play a role in our analysis and omitting other concepts, such as Layer Norms and positional embedding, which are explored in full by Vaswani et al. (2017) and Radford et al. (2019). We then provide the necessary background on the backward pass; a more comprehensive description is given by Clark (2017) and Bishop (2006). Finally, we discuss the building blocks of the Logit Lens method.

3.1 Transformer LMs

Generative Pre-trained Transformer (GPT), is an auto-regressive family of architectures containing multiple transformer blocks. Given a prompt sequence of n tokens, GPT predicts a single token. The architecture maintains an embedding dimension d throughout all layers. First, the input tokens are embedded using an embedding matrix E into vectors $X = [x_1, \cdots, x_n] \in \mathbb{R}^{n \times d}$. This is mirrored at the final stage, in which a decoding matrix

D projects the output of the last transformer block into a score for each token within the vocabulary.

Each transformer block comprises an attention layer (Attn) and a Multi-Layer Perceptron (MLP) layer, interconnected by a residual stream. The attention mechanism transfer vectors (information) from each of the preceding inputs to the current forward pass. In our study, we do not delve into this module and refer the reader to Radford et al. (2018) for more details.

The MLP layer (also known as FFN, Feed-Forward Network) consists of two fully connected matrices FF_1, $FF_2^T \in \mathbb{R}^{d \times d_m}$, with an activation function f between them: $\text{MLP}(X) = f(XFF_1)FF_2$.

Hence, the calculation that the l-th transformer block performs on its input hidden state, X^l, is given by $X^{l+1} = X^l + \text{Attn}(X^l) + \text{MLP}(\text{Attn}(X^l) + X^l)$.

3.2 Backpropagation

Backpropagation (Rumelhart et al., 1986; Le Cun, 1988) is an application of the chain rule to compute derivatives and update weights in the optimization of deep learning network-based models. The process begins with the model executing a forward pass, generating a prediction $\hat{y}$, which is subsequently compared to a desired target by quantifying the disparity through a loss score L. Following this, a backward pass is initiated, iterating through the model's layers and computing the layers' gradients in the reverse order of the forward pass.

For a given layer of the model that during the forward pass computed $z = xW$, where $x \in \mathbb{R}^{d_1}$, $z \in \mathbb{R}^{d_2}$ are its intermediate input and output, we compute its gradient matrix using the chain rule:

$$\frac{\partial L}{\partial W} = \frac{\partial z}{\partial W}\frac{\partial L}{\partial z} \in \mathbb{R}^{d_2 \times d_1} \tag{1}$$

We can directly compute $\frac{\partial z}{\partial W} = \frac{\partial xW}{\partial W} = x^\top$. The other derivative $\delta = \frac{\partial L}{\partial z} \in \mathbb{R}^{d_2}$ is known as the Vector-Jacobian Product (VJP) of z. It can be thought of as the hidden state of the backward pass and is the error factor that later layers project back.

In LMs, the output of the model is an unnormalized vector, $\hat{y} \in \mathbb{R}^{|\text{vocabulary}|}$, representing a score for each of the model's tokens. We denote the target token by an index $t \in [|\text{vocabulary}|]$. Typically the Negative Log-Likelihood (NLL) loss is used:

$$\hat{p} = \text{Softmax}(\hat{y}) \in \mathbb{R}^{|\text{vocabulary}|} \tag{2}$$

$$L = \text{NLL}(\hat{p}, t) = -\log(\hat{p}[t]) \in \mathbb{R} \tag{3}$$

where $\hat{p}$ represents the normalized probabilities of $\hat{y}$ and $[t]$ is its t-th value (the target token's probability). For the last layer's output $z = \hat{y}$, calculating its δ (VJP) can be done directly by ($k \in [|\text{vocabulary}|]$):

$$\delta[k] = \begin{cases} \hat{p}[k] - 1 \leq 0 & \text{if } k = t \\ \hat{p}[k] \geq 0 & \text{otherwise} \end{cases} \tag{4}$$

For an earlier layer in the model l, we cannot compute the VJP of its output z^l directly (here ^{l} indicates the layer's index). Since we iterate the model in a reverse order, we can assume we already computed the VJP of layer $l + 1$. If the layers are sequential, the output of layer l is the input of $l + 1$, therefrom $z^l = x^{l+1}$. Utilizing the backward step, we can compute:

$$\delta^l = \frac{\partial L}{\partial z^l} = \frac{\partial L}{\partial x^{l+1}} = \delta^{l+1}(W^{l+1})^\top \tag{5}$$

To summarize, in deep learning models, the gradient of a loss function L with respect to a given layer W, is the outer product of the layer's forward pass input, x, and its output z's VJP, δ:

$$\frac{\partial L}{\partial W} = \frac{\partial z}{\partial W}\frac{\partial L}{\partial z} = x^\top \cdot \delta \in \mathbb{R}^{d_2 \times d_1} \tag{6}$$

3.3 Vocabulary Projection Methods

nostalgebraist (2020) discovered that we can transform hidden states from LMs forward passes into vocabulary probabilities, thereby reflecting their intermediate predictions. Termed as **Logit Lens (LL)**, this method projects a vector x in the size of the embedding space d by applying it with the LM's decoding, the process that transforms the last transformer block's output into a prediction:

$$\text{LL}(x) = \text{Softmax}(ln_f(x)D) \in \mathbb{R}^{|\text{vocabulary}|} \tag{7}$$

where ln_f is the model's last Layer Norm before the decoding matrix D.

The projection captures the gradual building of LMs output (Millidge and Black, 2022; Haviv et al., 2023), and projections from later layers are more interpretable than earlier ones. Efforts such as Din et al. (2023) and Belrose et al. (2023) try to solve this gap by incorporating learned transformations into LL. However, to emphasize our main discoveries, we have not included such enhancements, which primarily aim to shortcut the models' computations and require dedicated training procedures.

An artificial neuron performs a weighted sum of its inputs, and appears as a column or a row of the model's matrices taken along a direction that has a dimensionality d. Static neurons can also be projected into tokens using LL: Geva et al. (2021), Geva et al. (2022b) observe that neurons of the first MLP matrix FF_1 determine the extent to which each neuron in FF_2 contributes to the intermediate prediction. Dar et al. (2022), Geva et al. (2023) employ the same approach to investigate the attention matrices. Elhage et al. (2021), Katz and Belinkov (2023) demonstrate how these neurons can elucidate model behavior, Wang et al. (2023), Millidge and Black (2022), Todd et al. (2023) use it to explore circuits and in-context learning.

Despite the growing interest in this approach, we are only aware of works that have applied it to the static weights of models or the hidden states of the forward pass. In contrast, our work is focused on the backward pass of LMs.

4 Backward Lens

In this section, we detail the methods we developed to analyze gradients based on our understanding of how each gradient matrix is formed. In the following sections, we consider the editing of a single sequence (prompt) to answer a single token (target) using NLL loss and without employing batching.

4.1 The Rank of the Gradient Matrices

Hu et al. (2022) and Mitchell et al. (2021) have observed the low-rank of MLP layers' gradients with a single input. However, they did not explain this phenomenon in the context of a matrix with a sequence of inputs, nor did they predict this rank. Building on the observations of Kiani et al. (2022), who bounded the rank of gradients in convolutional and vanilla recurrent networks, we define the following lemma for GPT:

Lemma 4.1. *Given a sequence of inputs of length n, a parametric matrix W and a loss function L, the gradient $\frac{\partial L}{\partial W}$ produced by a backward pass is a matrix with a rank of n or lower.*

Proof. According to Equation 6, the gradient of a matrix is $\frac{\partial L}{\partial W} = x^\top \cdot \delta$. Assuming x, δ are non-zero vectors, the rank of the gradient matrix is 1, given its interpretation as a span, a linear combinations, of a single column vector x, or equivalently, as a span of a single row δ. In the case when x or δ is a zero vector, the rank of the gradient matrix is 0.

In LMs, an input prompt comprises a sequence of n tokens, each of which introduces an intermediate input (x_i) at every layer. In this case, the gradient matrix is the sum of each x_i, δ_i's product:

$$\frac{\partial L}{\partial W} = \sum_{i=1}^{n} \frac{\partial z_i}{\partial W} \frac{\partial L}{\partial z_i} = \sum_{i=1}^{n} x_i^\top \cdot \delta_i \tag{8}$$

The maximum rank of the summed gradient matrix is n given each x_i, or δ_i, is linearly independent, since we sum n distinct rank-1 matrices. This rank would be lower than n if there are linear dependencies between x_i or between δ_i, with 0 being the minimum possible rank. For instance, if a linear dependency exists between only two δ_i's, the rank of the gradient matrix would be $n-1$. $\square$

Of particular interest is the case of the last layer of the transformer. In this case, the rank of the gradient is one, see Appendix A.

4.2 Gradient Matrices as Spans of Vectors

In our analysis we focus on the MLP layers, due to recent interest in identifying and editing the knowledge stored in these layers' neurons (Geva et al., 2022b, 2021; Dai et al., 2022; Mitchell et al., 2021; Meng et al., 2022). Consider the matrices of the MLP modules, FF_1 and FF_2, each having d_m neurons, which are d-sized vectors. Exploring all the modules' dimensions is prohibitive, see Appendix C. However, Equation 8 reveals that every gradient matrix is a sum of n outer products $x_i^\top \cdot \delta_i$. This view allows us to examine every gradient matrix as a sum of n pairs of vectors. Therefore, assuming we edit a sequence of inputs with under a few hundred tokens, our analysis would focus on only $n \ll d < d_m$ vectors in $\mathbb{R}^d$.

Every matrix formed by $x_i^\top \cdot \delta_i$ can be interpreted in two ways simultaneously: (1) as a span (linear combination) of x_i and (2) as a span of δ_i. Figure 2 illustrate the two viewpoints. We utilize this duality and examine gradients as the spans of n vectors, x_i or δ_i, guided by the observation that for each module there is only one span with vectors that are in the size of the module's neurons.

The gradients of FF_1 δ_i is not of size d and thus are not suitable for methods that examine vectors of the hidden state size. However, x_i are d-sized vectors and were previously explored using LL (Geva et al., 2022b; Dar et al., 2022). Therefore, for FF_1 we chose to observe the gradient matrix as a span of x_i, considering δ_i as the factor determining the extent of the update each x_i will introduce (the upper option in Figure 2). Explicitly, we refer to x_i as FF_1's **spanning set** since the j-th neuron of the

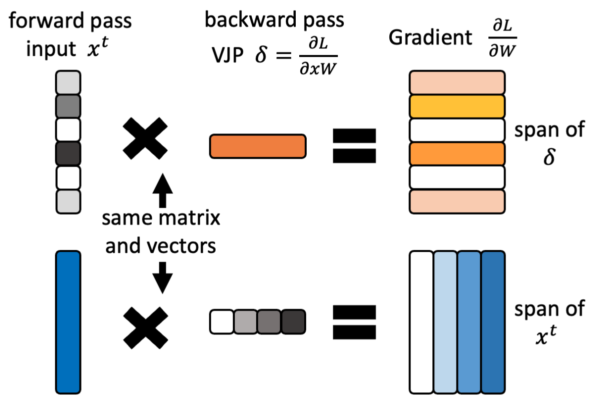


Figure 2: The calculation of gradient matrix by the outer product of $x^\top \cdot \delta$. Each row consists of the same values, but above we describe the matrix as a span (linear combinations) of δ, while below as a span of $x^\top$. The displayed vectors are presented transposed to emphasize the spanning effect.

gradient matrix is equal to a linear combination of x_i:

$$\frac{\partial L}{\partial FF_1}[j] = \sum_{i=1}^{n} x_i^\top \cdot \delta_i[j]\,, \tag{9}$$

where $\delta_i[j]$ is the j-th element of the vector δ_i.

The gradients of FF_2 The sizes of FF_2's x_i, δ_i are switched from those of FF_1, hence we chose δ_i as FF_2's gradient spanning set. Thus, its gradient's j-th neuron is viewed as a combination of δ_i:

$$\frac{\partial L}{\partial FF_2}[j] = \sum_{i=1}^{n} \delta_i \cdot x_i[j]\,, \tag{10}$$

where $x_i[j]$ is the j-th element of the vector x_i. In Section 5 we provide a theoretical explanation of why the choice of the VJP δ_i is not only a technical one, due to dimensionality considerations.

Logit Lens Since each spanning set comprises d-sized vectors, we can project them into tokens using LL. In the subsequent sections, we offer a theoretical explanation for why we expect to discover token embeddings in the vectors of these spanning sets and empirical examinations of their LL projections.

5 Understanding the Backward Pass

The VJPs, δ_i, are the hidden states of the backward pass and the vectors that constitute the gradient matrices (Section 3.2). In this section, we try to shed light on what information is encoded in the VJP. Additionally, we aim to explain how Equation 9 and Equation 10 cause the model to change its internal knowledge. For simplicity, we ignore Dropouts and Layer Norms.

5.1 The VJPs of the Top Layer

In this section, we analyze the initial VJPs that are created during the editing of a single prompt, as we describe in Section 3.2. The last matrix of an LM, which is the final model parameter used before calculating the loss score, is the decoding matrix $D \in \mathbb{R}^{d\times|\text{vocabulary}|}$. During the forward pass, this matrix calculates the output vectors $x_i D = \hat{y}_i$. When editing a prompt, we only use the final prediction of the last prompt's token $x_n D = \hat{y}_n$ to calculate the loss score.

We calculated D's VJP of the last token δ_n. According to Equation 5, the backward pass' VJP to the layer that preceded D is given by the following backward step:

$$\frac{\partial L}{\partial x_n} = \frac{\partial L}{\partial \hat{y}} D^\top = \delta_n D^\top \in \mathbb{R}^d \tag{11}$$

This result can be simplified as a weighted sum of D's columns:

$$\delta_n D^\top = \sum_{k=1}^{|\text{vocabulary}|} \delta_n[k] D^\top[k]\,, \tag{12}$$

where $D^\top[k] \in \mathbb{R}^d$ is the k-th column of D. Since D is the decoding (un-embedding) matrix, the k column of D holds the embedding of the model's k-th token. From the equation, $\delta_n[k] \in \mathbb{R}$ controls the magnitude by which we add the embedding of the k-th token into $\frac{\partial L}{\partial x_n}$. Plugging Equation 4 and using the notation $\hat{p} = \text{Softmax}(\hat{y}_n), t$ from Equation 2:

Lemma 5.1. *The VJP $\delta_n D^\top$ passed at the beginning of a backward pass is a vector in $\mathbb{R}^d$ that is a sum of weighted token embeddings. It is dominated by the embedding of the target token, $D^\top[t]$, multiplied by a negative coefficient $\delta_n[t] = \hat{p}[t] - 1$. The embedding of all other tokens $k \neq t$ are scaled by a positive coefficient $\hat{p}[k] D^\top[k]$.*

The VJP $\delta_n D^\top$ is the initial vector to be passed in the backward pass, if we simplify the effect of Layer Norms, which are discussed in Appendix B. In particular, this is the only VJP to span the last MLP layer's gradient.

The LL of δ_n is provided as $\text{Softmax}(ln_f(\delta_n D^\top))$. Except for ln_f, this behaves similarly to $\text{Softmax}(\delta_n D^\top)$. As the lemma shows, $\delta_n D^\top[t]$ is negative, while $\delta_n D^\top[k]$

is positive for all tokens $k \neq t$. Therefore, we can expect the target embedding to have the lowest probability in the softmax. In practice, since related tokens have more similar embeddings and similar entries in y_n, this effect is expected to be even more pronounced.

5.2 Storing Knowledge in LMs

In Section 4.2 we observed that each neuron in the MLP's gradients is a sum of vectors in $\mathbb{R}^d$ from the forward and backward passes, x_i and δ_i respectively. Based on this observation, we aim to understand how LM editing with a single prompt and a single backward pass changes the internal knowledge of a model. Explicitly, we study the implications of updating a weight matrix with its gradients: $W \leftarrow W + \eta \frac{\partial L}{\partial W}^\top$,where $\eta \in \mathbb{R}$ is a negative learning rate. In particular, while $\frac{\partial L}{\partial W} = \sum_{i=1}^{n} x_i^\top \cdot \delta_i$ we are interesting on the direct effect of updating a layer by the gradient of a single token from the edited prompt, $x_i^\top \cdot \delta_i$.

Lemma 5.2. *When updating an MLP layer of an LM using backpropagation only with the VJPs of the i-th token, δ_i, and rerunning the layer with the same inputs x_i from the forward pass of the prompt we used for the editing, the following occurs: (i) The inputs, x_i, are added or subtracted from the neurons of FF_1, thereby adjusting how much the activations of each corresponding neuron in FF_2 would increase or decrease. (ii) The VJPs δ_i are subtracted from the neurons of FF_2, amplifying in FF_2's output the presence of the VJPs after they are multiplied with negative coefficients.*

See Appendix D for the proof.

Since the change in FF_1 uses the given inputs x_i to amplify future activation, we term this mechanism "imprint". The modification of FF_2 is termed as the "shift", since it represents a process of altering the output of the layer. In summary, the "imprint and shift" mechanism depicts the MLP's learning process during a single backward pass as having two phases: Given the layer's original input and the new target, the process imprints a similar input through the update of FF_1 and subsequently shifts the output of FF_2 towards the new target. Figure 1 illustrates this process.

Updating with the full gradient: In practice, the gradient matrix is the sum of individual contributions, each given by the outer product of a token, $x_i^\top \cdot \delta_i$. However, as we will demonstrate in Section 6, only a few tokens dominate the full update.

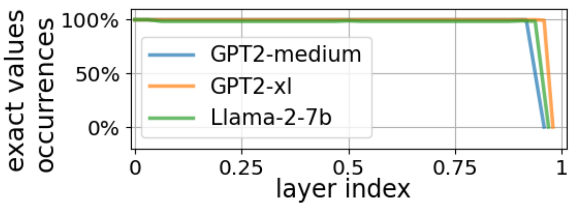


Figure 3: The percentage of occurrences where the rank of FF_1's gradient equals the length of the prompt used for editing. To show different models in the same plot, we normalize the layer indices. Except for the last layer, all layers and models exhibit the above equality more than 98.5% of the time.

Therefore, for the simplicity of our explanation, we can regard the complete update as consisting of just a few individual steps of "imprint and shift".

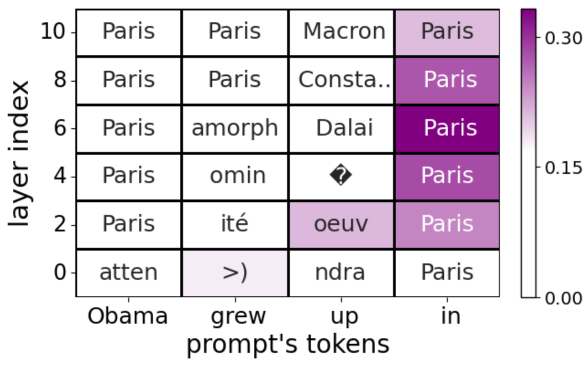
LL ranking refers to the index assigned to the vocabulary's tokens when ordered by the probability scores generated by LL (Equation 7). Updating FF_1 involves adding or subtracting x_i from weights, focusing on the most probable tokens from the LL ranking. Conversely, for FF_2, updating entails subtracting δ_i, effectively adding $-1 \cdot \delta_i$. This subtraction reverses the LL rankings, turning previously least probable tokens into most probable ones. Thus, when utilizing LL with FF_2's δ_i, attention should be given to the least probable tokens from the projections.

6 Experiments

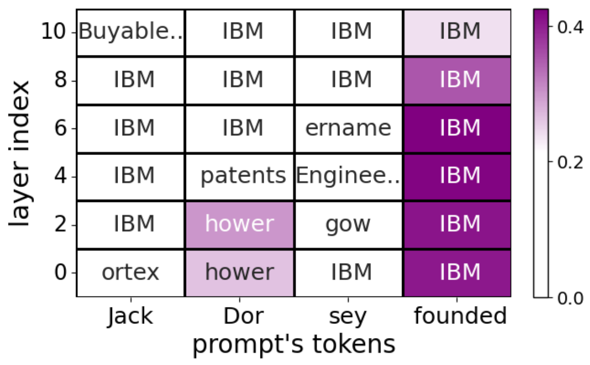
We conduct a series of experiments to support the results of Sec. 4 and 5, as well as to briefly demonstrate their application to LM analysis.

We employ GPT2 (Radford et al., 2019) and Llama2-7B (Touvron et al., 2023) in our experiments. We randomly sampled 100 prompts and their corresponding editing targets from the CounterFact dataset (Meng et al., 2022). For each model and prompt, we conducted a single backpropagation using SGD and without scaling optimizers, such as Adam (Kingma, 2014), and no batching.

The rank of the gradients To examine Lemma 4.1, we measure the rank of each layer gradient matrix. As depicted in Figure 3, for every prompt with the length of n tokens, the model's gradient matrices are almost always exactly rank n. The only exceptions are the last MLP layers, which have a rank of 1, as predicted in Section 4. Although unnoticeable from the figure, once in a few dozen examples, there is a drop of one or two



(a) Obama - Paris



(b) Jack Dorsey - IBM

Figure 4: The gradient of GPT2-small FF_2 when (a) editing the model to answer "Paris" for the prompt "Obama grew up in", and (b) when editing the model to answer "IBM" for "Jack Dorsey founded". Each cell shows the Logit Lens projection of the gradient's VJP (δ_i) for a token input and a layer. Non-English characters are replaced with a question mark, and long tokens are truncated with "..". According to Section 5.2, instead of showing the most probable token in each cell, we display the least probable one. The color indicates the norm of the VJP, with white cells indicating that almost no editing is done in practice.

in the rank of the gradients, indicating linear dependency in x_i or δ_i, see Section 4.1. This is not a result of a repeated token, since the positional encoding would still lead to a different x_i.

Logit Lens of Gradients Next, we present examples of our gradients' interpretation through LL in Figure 4 and in Appendix E. In each cell of these plots, the LL projections of the chosen spanning set (FF_1's x_i and FF_2's δ_i) are presented for a specific layer and a token from the prompt that was used for the editing.

Prior studies that projected the forward pass examine the LL projections of hidden states, highlighting the gradual change in the projected tokens between layers (nostalgebraist, 2020; Haviv et al., 2023). Similarly, Figure 4a presents a gradual change in the backward pass' VJP. Across most layers, LL reveals that the gradients represent the embedding of "Paris". Other projections have semantics that are related to "Paris" such as "Macron", the family name of the President of France. The norm of the VJP is indicted by color, and, in the top layers, the only meaningful updates are for the token "Paris". Some of the edits in the lower layers are harder to explain, similarly to the situation in those layers for the vanilla LL of the forward pass. Another example, presented in Figure 4b, shows similar patterns: when attempting to edit the model to suggest that Jack Dorsey, known as one of Twitter's founders, founded IBM, we observe that most gradients correspond to the embedding of "IBM".

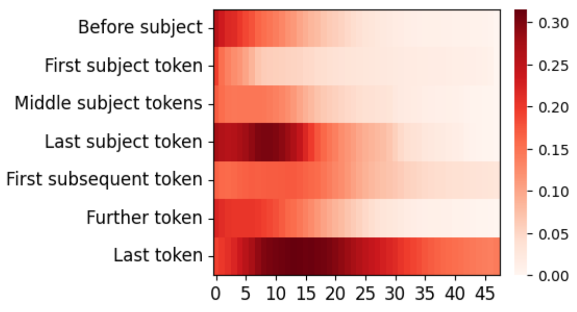


Figure 5: The norm of GPT2-xl's FF_2's VJPs (δ_i) as a function of the layers' index and segments of the edited prompts. White color represents close-to-zero updates with almost no effect on the model's weights.

Impact of Different Segments of the Prompt We observe that while all the prompt's tokens contribute to the gradient construction (Equation 8), the majority of these contributions are done by VJPs, δ_i, with a close-to-zero norm. Furthermore, upon examining the LL of every individual neuron from the gradient matrix (Appendix C), we found that all the projected tokens are correlated with only 1-2 vectors we can identify from the spanning sets presented in Section 4.2.

To discern the relative importance of tokens and layers in the gradient reconstruction, we divide each prompt's tokens into segments and plot their δ_i mean norm. This experiment is done with GPT2-xl, due to its extensive use in prior work on interpretability research.

Results are depicted for FF_2 in Figure 5, see Appendix F.1 for FF_1. Evidently, predominant updates occur in two main areas: (1) by the subject's tokens in the initial layers, and (2) by the last prompt's token around the second quarter of the layers. The majority of other tokens exhibit a norm close to zero throughout the layers, indicating that they have almost no effect on the updating. We hypothesize that the changes to the last subject token may involve editing the information transferred by the subject's token through attention, as demonstrated by Geva et al. (2023).

A complementary view is provided by considering the LL rank of the target token for each VJP δ_i (labeled by the segment of token i of the input). Figure 6 illustrates that the VJP of the last token from the edited prompts, δ_n, consistently ranks the target token among the least probable ones. The VJPs of other tokens from the edited prompt, δ_i, exhibit comparable behavior, generally ranking the target token as improbable.

The result reveals that along the first and last layers, some of the δ_i show degradation in their ranking of the target token, which we attribute to their low norms as reflected by Figure 5. We demonstrate in Appendix H that normalizing δ_i before LL magnifies the presence of the target token. Specifically, the drop at the model's last layer is due to the fact that apart from the last prompt's token, all the others have a zero vector δ_i at that layer (Appendix A).

Please note that the degradation of this rank in the first few layers might be related to the gap in LL interpretability for the earlier layers discussed in Section 3.3. In Appendix F.2 we provide a similar analysis for FF_1's gradients.

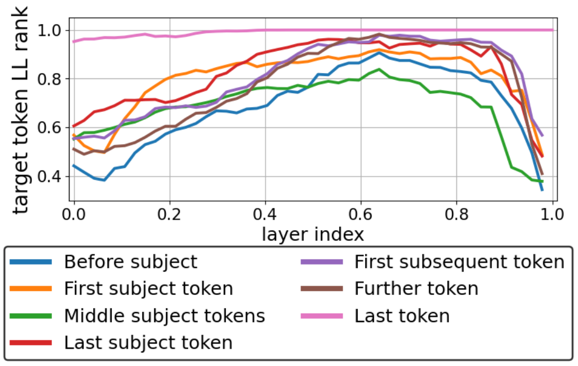


Figure 6: The Logit Lens rank of the target token for GPT2-xl FF_2's VJPs, δ_i. Most gradients tend to rank the target token as one of the least probable tokens, with the last token consistently ranking it as such. We found that the degradation of the first and last layers can be attributed to the proximity of certain δ_i norms to zero. In the initial layers, Logit Lens is less effective, thereby resulting in lower readability for the earlier layers.

7 Application: Editing Based on the "Shift" Mechanism

Prior work (Mitchell et al., 2021; Meng et al., 2022, 2023) introduced editing methods that change only FF_2's matrices. In Section 5.2 we identify the "shift" mechanism of editing FF_2. In Section 6 we observe that the dominant components in constructing the gradients are derived from the outer product of the last token's input and VJP, $x_n^\top \cdot \delta_n$, and that δ_n contains the embedding of the target token.

We hypothesize that we can edit LMs' internal knowledge by updating only a single FF_2 matrix with a single forward pass and an approximation of the VJP, thereby eliminating the need for the backward pass. Based on Lemma 5.1, the embedding of the editing target is annotated by $D^\top[t]$, where D is the decoding matrix and t is the index of the target token. Our experimental method works as follows: (i) We choose an MLP layer we wish to edit, $\hat{FF_2}$ (predefined as a hyper-parameter according to Appendix I). (ii) We run a single forward pass with the prompt whose output we want to edit. (iii) During the forward pass, we collect the last token input for the layer we want to edit, x_n. (iv) We collect the embedding of the target token $D^\top[t]$ and (v) update the MLP matrix by $\hat{FF_2} \leftarrow \hat{FF_2} + \eta \cdot x_n^\top \cdot D^\top[t]$, where η is the learning rate. We name this method "forward pass shifting". As motivation, please observe that $x_n(\hat{FF_2} + \eta \cdot x_n^\top \cdot D^\top[t])D^\top[t] - x_n\hat{FF_2}D^\top[t] = \eta \cdot \|x_n\|_2^2 \cdot \|D^\top[t]\|_2^2$.

We examined our method on 1000 samples from CounterFact (Table 1; the full results and additional implementation details are presented in Appendix I), and found that for single editing our approach is on par with the state-of-the-art methods MEND (Mitchell et al., 2021), ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), in editing a given prompt, but it falls short in comparison to ROME in generalization (editing paraphrases) and specificity. However, our method has much lower runtime complexity and does not employ a multi-step (iterative) execution. Overall, our results suggest we might be able to find "shortcuts" in the implementation of fine-tuning by injecting tokens directly into LMs' layers.

METHOD	EFFICACY ↑	PARAPHRASE ↑	NEIGHBORHOOD ↑	N-GRAM↑
ORIGINAL MODEL	0.4	0.4	11.21	626.94
FINETUNING (MLP 0)	96.4	7.46	10.12	618.81
FINETUNING (MLP 35)	100.0	46.1	5.59	618.50
MEND	71.4	17.6	7.73	623.94
ROME	99.4	71.9	10.91	622.78
MEMIT	79.4	40.7	10.98	627.18
FORWARD PASS SHIFT	99.4	41.6	6.02	622.45

Table 1: GPT2-xl single editing results for CounterFact. Efficacy represents the editing success rate (accuracy). Paraphrase denotes the accuracy of predicting the new target for phrases derived from the edited prompt. Neighborhood examine the model accuracy on prompts we wish the edit will not change but have similar domain to the edited one. N-gram measures generation fluency using weighted bi- and tri-gram entropies.

8 Conclusions

Other LL-type interpretability contributions shed light on LMs through the forward pass. Here, we show that gradients can be projected into the vocabulary space and utilize the low-rank nature of the gradient matrices to explore the backward pass in an interpretable way. As we show, the gradients are best captured by a spanning set that contains either the input to each layer, or its VJP. These two components, which are accessible during the forward and backward passes, are used to store information in the MLP layers, using a mechanism we call "imprint and shift". We provided experimental results to substantiate the results of our analysis, including an editing method that only requires a single forward pass, but is on par with the SOTA knowledge editing methods.

Future Work: The focus of this work is in understanding the mechanism behind the simplest form of LM editing. We hope our observations can serve further research on how knowledge is embedded into the model's parameters. Scaling our experiments to multiple editing tasks or full fine-tuning would result in losing the low-rank characteristic that we utilize in this work. However, our findings in Section 6 suggest that only a small set of VJPs and subspaces dominate the editing process. Therefore, future research could interpret complex editing scenarios by examining only the most dominant VJPs, or by decomposing the full gradients into low-rank components using techniques such as singular value decomposition (SVD).

Notably, this work did not explore the similarity between gradients of different sentences' editing. The question of how subspaces are shared between different edits and tasks remains an open question, which has been recently discussed in the context of superposition (Elhage et al., 2022). While most work has examined superposition phenomenon by looking at the forward pass of LMs, exploring how information is embedded into the same subspaces might shed light on how LMs work within their constrained embedding space.

In addition, in this work we utilize Logit-Lens (LL) projection, which is known to have reduced interpretability abilities in the earlier layers of models compared to later ones. Future research could build on our findings by interpreting VJPs using alternative projection methods, such as those proposed by Din et al. (2023), or by employing techniques like sparse autoencoders (Bricken et al., 2023).

9 Limitations

Our use of LL in projecting gradients has limitations when it comes to explaining the gradients of earlier layers. At this point, it remains unclear whether gradients operate in the same embedding space across all layers or if another transformation is required for projecting earlier layers. This question is currently being explored for the forward pass (see Section 3.3), suggesting additional learned transformations to the first layers. Given the lack of a wide consensus on this additional transformation, we have opted to employ only the original LL projection in our analysis. Furthermore, some recent contributions against LL argue that this method is more correlated with LMs' behaviors, rather than causally explaining them. Our work shows that at least in the later layers of LMs, token embeddings are directly placed into the weights of the LM, making LL projections well-justified.

Recently, alternative approaches have been proposed to explain LMs by intervening in the forward pass (Meng et al., 2022). When combined with token projection methods, this approach holds promise in providing insights into the "thinking" process of LMs (Ghandeharioun et al., 2024).

Our work ignores the additional scaling that is introduced by optimizers other than Stochastic Gradient Descent, such as Adam (Kingma, 2014). While the backward pass's VJPs remain unaffected when such optimizers are employed, they do alter the rank and weights of each gradient matrix, due to the additional scaling.

Our approach to explaining how knowledge is stored in LMs is grounded in single editing with a constant embedding. While our approach elucidates how models store various information, fine-tuning is typically conducted on multiple prompts and involves multiple steps (iterations). Additionally, training a model from scratch includes the training of its embeddings.

Our experimental approach to editing LMs with "forward pass shift" is presented as a case study rather than as a suggested alternative to existing methods. The results in Section 7, Appendix I might obfuscate "editing" and "output shifting", since only plotting the desired answers does not fully encapsulate the effect of the edit on similar prompts, which is a challenge faced by most editing benchmarks and datasets.

Our focus on the MLP layers excludes the attention layers. This decision is influenced by the growing consensus that MLPs are where LMs predominantly store information (Dai et al., 2022; Meng et al., 2022). We acknowledge the possibility that attention layers may also store information and that editing MLPs and attention simultaneously could have different effects on the model from those detailed in Section 5.2.

In the theoretical exploration, we omitted Layer Norms (Appendix B) and Dropouts to simplify the explanation, aligning with prior interpretability work that also overlooks these components due to their negligible impact compared to others (Elhage et al., 2021; Geva et al., 2023; Meng et al., 2023). While these components may influence gradients, our empirical results were conducted on full assumption-free LMs and show consistent patterns with the theoretical parts.

Lastly, this work was conducted on Decoder LMs with sequential architecture. It is important to note that other types of LMs might exhibit different behaviors in terms of their gradients. We chose this architecture as it is currently the leading architecture both in terms of downstream performance and in interpretability research (nostalgebraist, 2020; Brown et al., 2020; Touvron et al., 2023).

10 Ethics and Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. However, future research could use the methods we developed to edit LMs. We hope such cases would be for developing better and safer models, rather than promoting harmful content.

Acknowledgements

This work was supported by the Tel Aviv University Center for AI and Data Science (TAD), the ISRAEL SCIENCE FOUNDATION (grant No. 448/20), an Open Philanthropy alignment grant, and an Azrieli Foundation Early Career Faculty Fellowship. The contribution of the first author is part of a PhD thesis research conducted at Tel Aviv University.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *stat*, 1050:21.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

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

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Christopher Bishop. 2006. Pattern recognition and machine learning. *Springer google schola*, 2:531–537.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac

Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

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

Hila Chefer, Idan Schwartz, and Lior Wolf. 2022. Optimizing relevance maps of vision transformers improves robustness. *Advances in Neural Information Processing Systems*, 35:33618–33632.

Kevin Clark. 2017. Computing neural network gradients.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

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

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2022. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*.

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

Yogesh K Dwivedi, Nir Kshetri, Laurie Hughes, Emma Louise Slade, Anand Jeyaraj, Arpan Kumar Kar, Abdullah M Baabdullah, Alex Koohang, Vishnupriya Raghavan, Manju Ahuja, et al. 2023. "so what if chatgpt wrote it?" multidisciplinary perspectives on opportunities, challenges and implications of generative conversational ai for research, practice and policy. *International Journal of Information Management*, 71:102642.

N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. 2021. A mathematical framework for transformer circuits.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022a. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscope: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*.

Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. 2023. Knowledge is a region in weight space for fine-tuned language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1350–1370, Singapore. Association for Computational Linguistics.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. 2023. Understanding transformer memorization recall through idioms. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 248–264. Association for Computational Linguistics.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Sakshi Indolia, Anil Kumar Goswami, and Pooja Asopa. 2018. Conceptual understanding of convolutional neural network-a deep learning approach. *Procedia computer science*, 132:679–688.

Shahar Katz and Yonatan Belinkov. 2023. VISIT: Visualizing and interpreting the semantic information flow of transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14094–14113, Singapore. Association for Computational Linguistics.

Bobak Kiani, Randall Balestriero, Yann LeCun, and Seth Lloyd. 2022. projunn: efficient method for training deep networks with unitary matrices. *Advances in Neural Information Processing Systems*, 35:14448–14463.

DP Kingma. 2014. Adam: a method for stochastic optimization. In *Int Conf Learn Represent*.

Y Le Cun. 1988. A theoretical framework for backpropagation. In *Proceedings of the 1988 Connectionist Models Summer School*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Vivek Miglani, Aobo Yang, Aram Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. 2023. Using captum to explain generative language models. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 165–173.

Beren Millidge and Sid Black. 2022. The singular value decompositions of transformer weight matrices are highly interpretable.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. In *International Conference on Learning Representations*.

nostalgebraist. 2020. interpreting gpt: the logit lens.

Chris Olah, Nick Cammarata, Chelsea Voss, Ludwig Schubert, and Gabriel Goh. 2020. Naturally occurring equivariance in neural networks. *Distill*, 5(12):e00024–004.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296.

Soumya Sanyal and Xiang Ren. 2021. Discretized integrated gradients for explaining language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10285–10299.

Gabriele Sarti, Nils Feldhus, Ludwig Sickert, and Oskar van der Wal. 2023. Inseq: An interpretability toolkit for sequence generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 421–435, Toronto, Canada. Association for Computational Linguistics.

Adi Simhi and Shaul Markovitch. 2023. Interpreting embedding spaces by conceptualization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1704–1719, Singapore. Association for Computational Linguistics.

K Simonyan, A Vedaldi, and A Zisserman. 2014. Deep inside convolutional networks: visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Alex Tamkin, Mohammad Taufeeque, and Noah D Goodman. 2023. Codebook features: Sparse and discrete interpretability for neural networks. *arXiv preprint arXiv:2310.17230*.

Yuandong Tian, Yiping Wang, Beidi Chen, and Simon Du. 2023. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. *arXiv preprint arXiv:2305.16380*.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. 2023. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

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.

Quan-shi Zhang and Song-Chun Zhu. 2018. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39.

A The Rank of The Last Layer

In Section 4.1, we delve into the observation that each gradient matrix has a rank equal to the length of the edited prompt (annotated by n), except for the last layer's ones. In this section, we explain why the last layer's MLP matrices are always rank-1.

The backward pass, applied to the final loss score (L, Section 3.2), generates a computational graph that is reversed in direction compared to the forward pass Section 3.2. It begins with the loss score and the matrices of the last layer, proceeding in reverse order until reaching the matrices of the first layer. This computational graph encapsulates every hidden state and intermediate result that contributed to the final prediction, which is the output of the last layer for the last token in the prompt.

One might initially assume that, since the last prediction was formed by the input of the last token, only its hidden states would be involved in this computational graph. However, due to the attention mechanism, hidden states from previous forward passes can be recalled and utilized in subsequent forward passes, contributing to all the tokens that follow them in the prompt. The last hidden state to be recalled using the attention modules is called at the model's last layer's attention module, which precedes an MLP module in sequential architectures, such as GPT2 and Llama-7B. Hence, in every layer, MLP inputs in the reverse computational graph comprise all individual intermediate inputs x_i from the forward pass of each token in the prompt. However, at the last layer, the only input included is the one belonging to the last prompt token x_n. For this reason, also only the VJP of the last token, δ_n, is included in the reconstruction of the gradients of the last MLP layer, while the δ_i for all the other tokens from the prompt are not included (or more correctly, they are equal to the zero vectors).

When constructing the gradients using x_i and δ_i, the rank of each layer is equal up to the number of x_i, δ_i involved in the computational graph (assuming linear independence Section 4.1). This implies that all layer matrices are formed by $\frac{\partial L}{\partial W} = \sum_{i=1}^{n} x_i^\top \cdot \delta_i$ except for the last layer, which is constructed with $\frac{\partial L}{\partial W} = x_n^\top \cdot \delta_n$, which is rank-1.

In our study, especially in our figures and tables, we decided to include all the vectors of the last layer, including those from tokens which are not the prompt's last, which are thus equal to zero vectors. This approach is also the reason for the observed changes in the behavior of the gradients in some figures. For example, in Figure 6 we can see that all the graphs (except for the last token's) converge to the same value at the last layer. The reason for this is that they are all equal to the zero vector. In Figure 11 we see the LL projection of the VJPs from the model's last layer, which are equal to projecting the zero vector.

B Layer Norm's Backward Step

Layer Norms (Ba et al., 2016) are used in transformers to scale their activations, thereby speeding up the training process and making it more stable. Although they contain learnable weights, they are usually omitted in interpretability literature. For completeness, we demonstrate how to calculate the derivative of the Layer Norm component and examine its effect on the backward pass.

Given an input vector $x \in \mathbb{R}^d$, Layer Norm is defined as the following scaled dot product:

$$\text{LayerNorm}(x) = \alpha \odot \frac{x - \mu}{\sigma} + \beta \quad (13)$$

where μ and σ are the mean and standard deviation of the input x respectively. $\alpha, \beta \in \mathbb{R}^d$ are scaling and shifting (bias) parameters, which are learnable.

In a general fine-tuning process, as well as in our experiment in Section 6, Layer Norms are frozen (their parameters are not updated). However, during the backward pass, we do apply the backward step through the Layer Norm, necessitating the calculation of the derivative of the Layer Norm with respect to the forward pass inputs.

Given the VJP $\delta \in \mathbb{R}^d$ up to a given Layer Norm, the backward step of the Layer Norm that produces $\hat{\delta}$ is defined by:

$$\gamma_1 = (x - \mu) \odot \sigma \in \mathbb{R}^d \quad (14)$$

$$\gamma_2 = \delta \odot \alpha \in \mathbb{R}^d \quad (15)$$

$$\hat{\delta} = (\gamma_2 - \mu_{(\gamma_1)} - \gamma_1 \mu_{(\gamma_2 \odot \gamma_1)}) \odot \sigma \in \mathbb{R}^d \quad (16)$$

where $\mu_{(v)}$ is the mean of the vector v.

We view this backward step as a mirroring process to the normalization step performed during the forward pass. Furthermore, our results in Section 6, conducted on full transformer models, demonstrate that the effect of Layer Norms on the behavior (information embedded) that was predicted by the theoretical analysis Section 5 is negligible.

C Why Decomposed Gradient Analysis Makes Sense

In Section 3.3, Section 4.2 we establish our interpretation of gradients via spanning sets. This approach is based on the understanding that each neuron in the gradient matrix is formed by the linear combination of x_i (the forward pass's intermediate inputs) or δ_i (VJPs, the backward pass's hidden states). In this section, we aim to illustrate, through a singular example, why analyzing a gradient matrix through its spanning set is more informative and simpler compared to attempting to analyze the full gradient matrix.

We use GPT2-medium (24 layers and 330M parameters) for our examination. We examine the MLP gradients using the prompt "Lionel Messi plays for", to which the model responds with "Barcelona". We edit the model with a single backward pass to respond "Paris". In the case of this model, each MLP matrix comprises 4096 neurons. Consequently, to apply the Logit Lens (LL) projection to a particular gradient matrix, the process needs to be applied 4096 times.

We start by analyzing FF_2 from layer 14 . In Table 2 we present samples of gradient neurons' projections by LL. In Section 4.2, we elaborate on how each neuron is formed by multiplying an interpretable vector by a coefficient (δ_i and x_i), which in turn dictates its norm. We group these neurons based on their norms, unveiling shared projections among neurons within the same norm group. To underscore the proximity of certain neurons to the zero vector, we also include the projection of the zero vector. Expanding Table 2 to every individual neuron in the matrix is impractical, given the challenge of reading a table with 4096 rows. Instead, we use plots Figure 7: the initial plot presents the LL intersection, measuring the extent of overlap between the top 100 most probable tokens from two vectors, while the second shows the cosine similarity between the vectors.

We repeat the process after sorting the gradient neurons according to their norms Figure 8. The gradients with the higher norms, are almost identical to the last VJP (δ_n), with alignment extending up to the sign of the vectors. In Table 3 LL reveals that these VJPs project "Paris". Gradients with low norms may appear correlated with parts of the spanning set's vectors (FF_2's δ_i), yet they are more correlated to the zero vector, emphasizing that these neurons do not update the model's weights (induce minimal change).

In addressing color shifting, we see around index 500 from the right that this is where the activations change sign from positive to negative. The negative learning rate causes the positive activation to add "Paris" into those neurons, while the negative activation reduces "Paris". In both cases, the process causes the model to add "paris" in the same direction (Section 5.2).

We repeat this type of analysis with FF_1 . We remind that our interpretation for this layer's spanning set is its inputs x_i Section 4.2. Again, we see the alignment between individually analyzing the neurons of the gradients and the spanning set Table 4, Table 5, Figure 9.

In conclusion, In this section, we demonstrate the converging results of analyzing individual gradient neurons via LL and the spanning sets interpretation. The aim of this analysis is to emphasize the efficacy of employing these spanning sets to simplify experiments involving a vast number of vectors (neurons) into a much smaller, representative subset. The advantage of this simplification is twofold: it conserves computational resources and reduces time expenditure.

Group	norm	LL top	LL bottom
Biggest by norm	1.212	Paris, Paris, Marse	ufact, Logged, otomy
	0.352	Paris, Paris, Marse	ufact, Spanish, Gerr
	0.297	Paris, Paris, Copenhagen	ceremonial, cade, uana
Medium by norm	0.033	ufact, Gerr, sheriff	Paris, Paris, ienne
	0.033	Logged, turtle, ceremon	Paris, Paris, France
	0.033	ufact, sec, recess	Paris, Paris, qus
Smallest by norm	0.001	,, the, and	VIDIA, advertisement, Dialogue
	0.001	,, the, and	VIDIA, advertisement, Magikarp
	0.001	,, the, and	VIDIA, advertisement, Companies
zero vector	0	,, the, and	VIDIA, advertisement, Companies

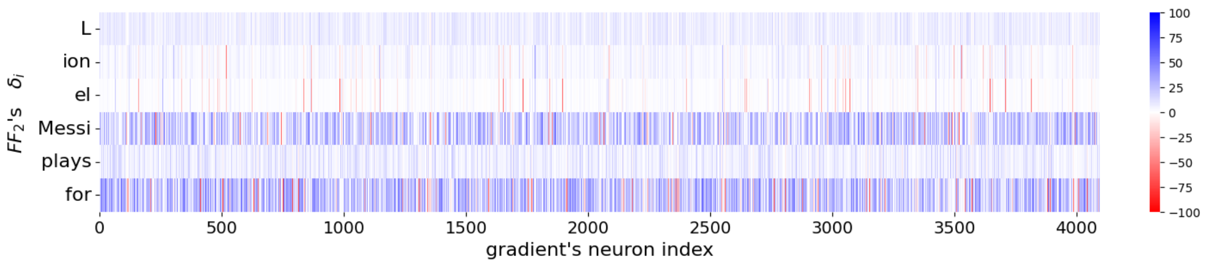
Table 2: Sample of gradient's neurons projection via Logit Lens (LL) from GPT2-medium, FF_2 matrix, layer 14. **LL TOP** stands for the most probable tokens via LL, while **BOTTOM** are the most improbable ones. In this example, we edit the prompt "Lionel Messi plays for" with the editing target "Paris". In the projected tokens we notice the predominance of "Paris", and also that gradient's neurons with a relatively low norm project the same tokens as the zero-vector.

prompt token	norm	LL top	LL bottom
L	0.029	Rutherford, Apost, PROG	Paris, Paris, ienne
ion	0.035	ramid, ngth, livest	France, ée, É
el	0.067	unlaw, owship, arantine	Libyan, Libya, France
Messi	0.165	ğ, relic, ejected	vu, igmat, tain
plays	0.026	surv, POV, NTS	Paris, hotels, Merit
for	0.165	ceremonial, ado,	Paris, Paris, Copenhagen

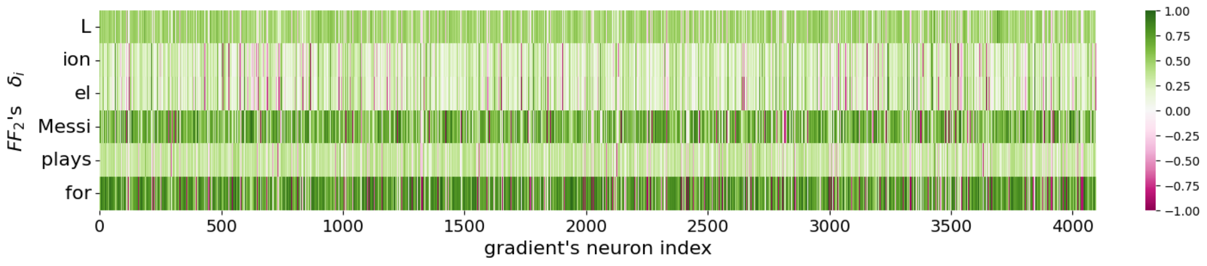
Table 3: The Logit Lens of the VJPs (δ_i) of GPT2-medium, FF_2 matrix, layer 14. Notice the dominance of "Paris" (the editing target) in the projected vocabulary and the norm ratio of the vectors.

Group	norm	LL top	LL bottom
Biggest by norm	0.438	Cruz, ization, ize	mathemat, trave, nodd
	0.422	Football, Jr, Team	theless, challeng, neighb
	0.369	psychiat, incent, theless	Jr, Junior, Sr
Medium by norm	0.02	the, a, hire	irez, inelli, intosh
	0.02	the, a, one	theless, Magikarp, irez
	0.02	perpend, coerc, incent	Junior, Jr, Football
Smallest by norm	0.002	,, the, and	Magikarp, VIDIA, advertisement
	0.002	,, the, and	advertisement, VIDIA, Magikarp
	0.001	,, the, and	VIDIA, advertisement, Companies
zero vector	0	,, the, and	VIDIA, advertisement, Companies

Table 4: Samples of gradient's neurons projection via Logit Lens (LL) from GPT2-medium, FF_1 matrix, layer 14.

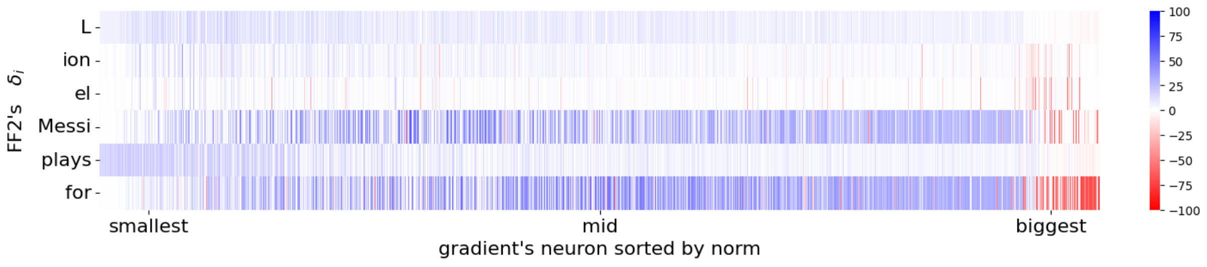


(a) Logit Lens intersection

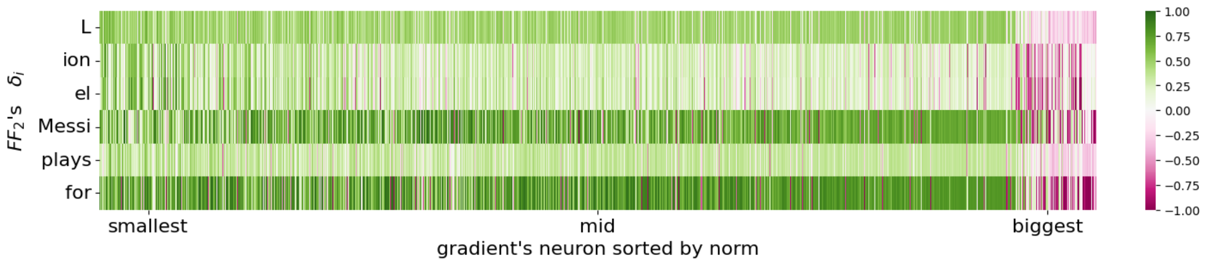


(b) Cosine Similarity

Figure 7: VJPs (δ_i, the spanning set of FF_2) and single gradient's neurons comparison for layer 14's FF_2. In Figure 7a a positive score is the amount of shared tokens between the top 100 most probable tokens after projecting each vector. A negative score is presented if multiplying one of the vectors by -1 produces a higher amount of shared tokens (than the score presented with a negative sign). The reason we apply this -1 multiplication is to address cases where two vectors have high correlation up to their sign (similarly to two vectors that overlap each other after one of them is multiplied by -1, making their cosine similarity negative).



(a) Logit Lens intersection

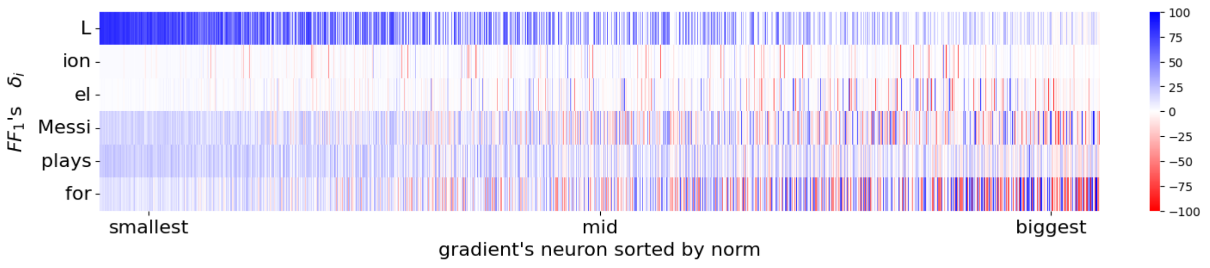


(b) Cosine Similarity

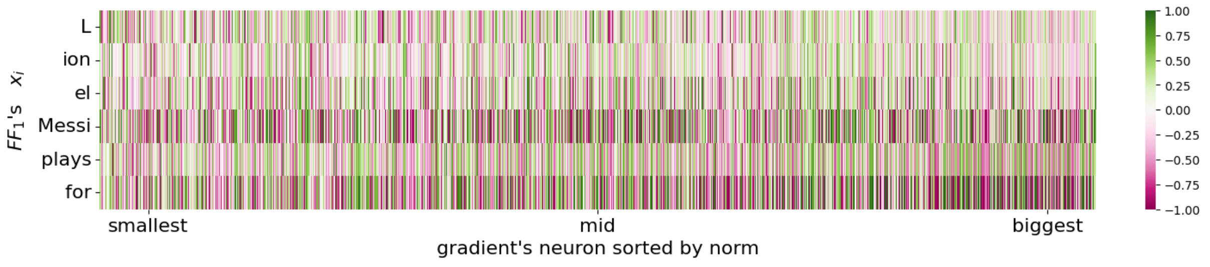
Figure 8: VJPs (δ_i) and single gradient's neurons (sorted by norm) comparison for layer 14's FF_2. It is noteworthy that high-norm neurons, corresponding to those activated with positive activations during the forward pass, inject the VJP of "for" with a flipped sign. Medium-sized norm neurons also exhibit correlation with the representative vector of "for". Smallest by-norm neurons show minimal correlation; we refer to their proximity to the zero vector.

PROMPT TOKEN	NORM	LL TOP	LL BOTTOM
L	9.499	,,, the	FontSize, 7601, Magikarp
ion	7.808	fall, fish, wood	nodd, incorpor, accompan
el	7.728	Cruz, McC, Esp	perpend, mathemat, shenan
Messi	6.944	Jr, Junior, Sr	theless, psychiat, incent
plays	6.715	football, golf, alongside	hedon, ilts, uries
for	6.596	Team, team, a	irez, newsp, Magikarp

Table 5: The Logit Lens of the inputs (x_i) of GPT2-medium, FF_1 matrix, layer 14. According to our observation Section 5.2, those are also the embeddings the gradients inject into the model's weights.



(a) Logit Lens intersection



(b) Cosine Similarity

Figure 9: The inputs x_i (which are FF_1's spanning set) and single gradient's neurons (sorted by norm) comparison for layer 14's FF_1.

D Proof of Lemma 5.2

Lemma 5.2. *When updating an MLP layer of an LM using backpropagation only with the VJPs of the i-th token, δ_i, and rerunning the layer with the same inputs x_i from the forward pass of the prompt we used for the editing, the following occurs: (i) The inputs, x_i, are added to or subtracted from the neurons of FF_1, thereby adjusting how much the activations of each corresponding neuron in FF_2 increase or decrease. (ii) The VJPs δ_i are subtracted from the neurons of FF_2, amplifying in FF_2's output the presence of the VJPs after they are multiplied with negative coefficients.*

Proof. In Section 4.2, we revealed that FF_1 weights are updated by injections (adding or subtracting vectors) of its x_i. This update is done according to the coefficients of its δ_i, after multiplying them with the learning rate. If we rerun the same layer with the same input after the update, we obtain the following output per each neuron j:

$$\begin{aligned} & x_i \cdot (FF_1[j] + \eta \cdot x_i^\top \cdot \delta_i[j]) = \\ & x_i \cdot FF_1[j] + x_i \cdot \eta \cdot x_i^\top \cdot \delta_i[j] = \\ & x_i \cdot FF_1[j] + \|x_i\|_2^2 \cdot \eta \cdot \delta_i[j] \in \mathbb{R} \end{aligned} \tag{17}$$

$x_i \cdot FF_1[j]$ is the pre-edit output of this neuron. The second component, $\|x_i\|_2^2 \cdot \eta \cdot \delta_i[j]$, is derived from the update and can be positive or negative, hence it controls the increment or decrement of this output compared to the pre-edit one. In models with monotonic (or semi-monotonic) activation functions, such as ReLU (GeLU is positive monotonic only from around -0.75), the activation of the corresponding neuron in FF_2 will be changed directly by this addition to that output.

In Section 4.2, we show how FF_2's δ_i form the gradient matrix. Consider the result of updating only FF_2 and rerunning the same layer:

$$\begin{aligned} & x_i[j] \cdot (FF_2[j] + \eta \cdot \delta_i \cdot x_i[j]) = \\ & x_i[j] \cdot FF_2[j] + \eta \cdot (x_i[j])^2 \cdot \delta_i \in \mathbb{R}^d \end{aligned} \tag{18}$$

$x_i[j] \cdot FF_2[j]$ represents the original output of this neuron. Since $(x_i[j])^2$ is always non-negative and η is negative, the update is a subtraction of δ_i from each neuron.

□

Figure 10 illustrates "imprint and shift" mechanism.

E Additional Logit Lens of Gradients Examples

The work that presented Logit Lens (LL) (nostalgebraist, 2020) established this method by constructing tables of the projected results of different prompts, illustrating the tokens each individual forward pass represents at each layer of the model. In this section, we present a similar approach, focusing on the backward pass rather than the forward pass. In the following figures, we provide examples of the LL projections of gradients when they are interpreted as combinations of the intermediate inputs x_i or VJPs δ_i according to Section 4.2.

In addition to illustrating the information stored in the gradient matrices, the following tables also describe the information moving through LMs during the forward and backward pass. Our interpretation of FF_1 using x_i reflects the gradual buildup in the prediction of the forward pass, as x_i represents the intermediate inputs for each MLP layer. However, FF_2's δ_i, VJPs, serve as the backpropagation counterpart to the forward pass x_i. We can conceptualize them as the input of the MLP when executing the backward pass, or as the error propagated from later layers to earlier ones. In conclusion, Figure 11, 12, 13, 14, 15 depict the information stored within the gradients, simultaneously also comparing the information revealed by LL from the forward pass (x_i, known from prior studies) with that from the backward pass (δ_i), which is part of our innovative contribution.

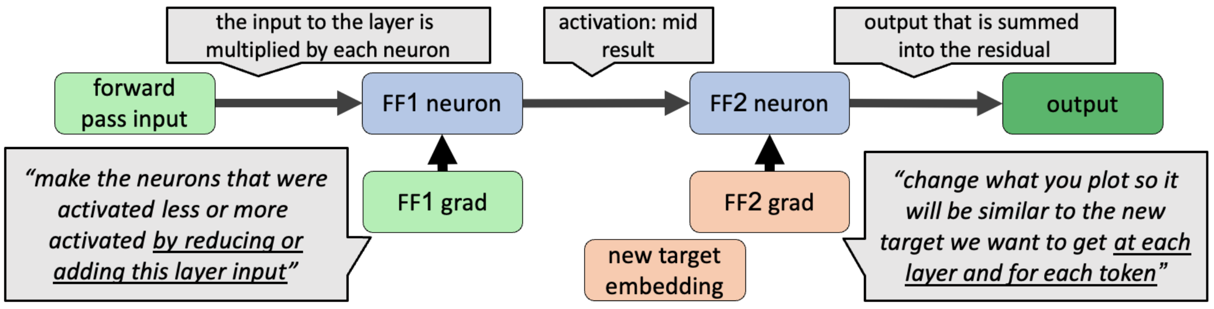
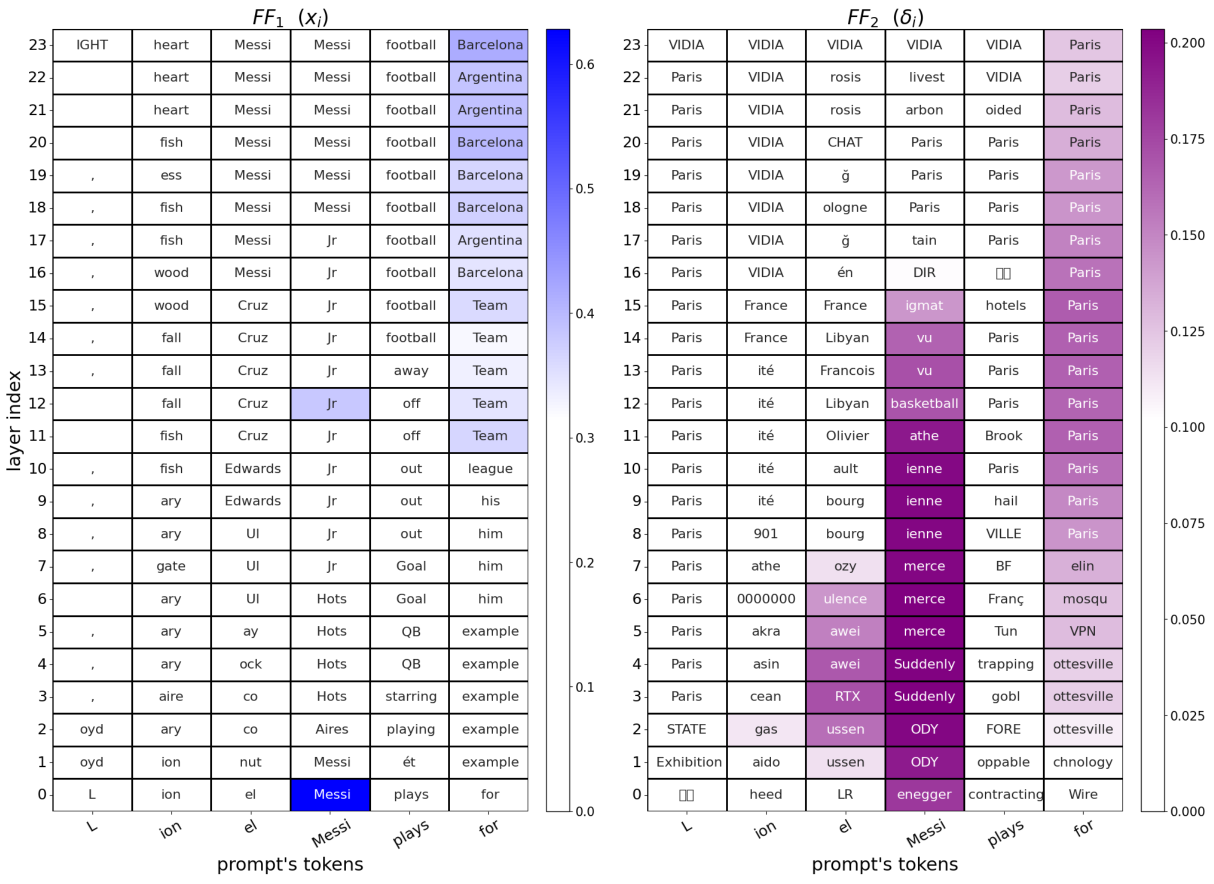

the input to the layer is multiplied by each neuron
activation: mid result
output that is summed into the residual
forward pass input
FF1 neuron
FF2 neuron
output
FF1 grad
FF2 grad
new target embedding
"make the neurons that were activated less or more activated by reducing or adding this layer input"
"change what you plot so it will be similar to the new target we want to get at each layer and for each token"

Figure 10: The **Imprint and Shift** mechanism of backpropagation. "grad" represents a single neuron from a gradient matrix. The color of FF_1 grad is the same as the forward-pass input, while FF_2 is the same as the new target embedding, suggesting that they are similar to each other.



FF_1 (x_i)

layer index	L	ion	el	Messi	plays	for
23	IGHT	heart	Messi	Messi	football	Barcelona
22		heart	Messi	Messi	football	Argentina
21		heart	Messi	Messi	football	Argentina
20		fish	Messi	Messi	football	Barcelona
19	,	ess	Messi	Messi	football	Barcelona
18	,	fish	Messi	Messi	football	Barcelona
17	,	fish	Messi	Jr	football	Argentina
16	,	wood	Messi	Jr	football	Barcelona
15	,	wood	Cruz	Jr	football	Team
14	,	fall	Cruz	Jr	football	Team
13	,	fall	Cruz	Jr	away	Team
12		fall	Cruz	Jr	off	Team
11		fish	Cruz	Jr	off	Team
10	,	fish	Edwards	Jr	out	league
9	,	ary	Edwards	Jr	out	his
8	,	ary	Ul	Jr	out	him
7	,	gate	Ul	Jr	Goal	him
6		ary	Ul	Hots	Goal	him
5	,	ary	ay	Hots	QB	example
4	,	ary	ock	Hots	QB	example
3	,	aire	co	Hots	starring	example
2	oyd	ary	co	Aires	playing	example
1	oyd	ion	nut	Messi	ét	example
0	L	ion	el	Messi	plays	for

FF_2 (δ_i)

layer index	L	ion	el	Messi	plays	for
23	VIDIA	VIDIA	VIDIA	VIDIA	VIDIA	Paris
22	Paris	VIDIA	rosis	livest	VIDIA	Paris
21	Paris	VIDIA	rosis	arbon	oided	Paris
20	Paris	VIDIA	CHAT	Paris	Paris	Paris
19	Paris	VIDIA	ğ	Paris	Paris	Paris
18	Paris	VIDIA	ologne	Paris	Paris	Paris
17	Paris	VIDIA	ğ	tain	Paris	Paris
16	Paris	VIDIA	én	DIR	□□	Paris
15	Paris	France	France	igmat	hotels	Paris
14	Paris	France	Libyan	vu	Paris	Paris
13	Paris	ité	Francois	vu	Paris	Paris
12	Paris	ité	Libyan	basketball	Paris	Paris
11	Paris	ité	Olivier	athe	Brook	Paris
10	Paris	ité	ault	ienne	Paris	Paris
9	Paris	ité	bourg	ienne	hail	Paris
8	Paris	901	bourg	ienne	VILLE	Paris
7	Paris	athe	ozy	merce	BF	elin
6	Paris	0000000	ulence	merce	Franç	mosqu
5	Paris	akra	awei	merce	Tun	VPN
4	Paris	asin	awei	Suddenly	trapping	ottesville
3	Paris	cean	RTX	Suddenly	gobl	ottesville
2	STATE	gas	ussen	ODY	FORE	ottesville
1	Exhibition	aido	ussen	ODY	oppable	chnology
0	□□	heed	LR	enegger	contracting	Wire

Figure 11: GPT2-medium MLP gradients via our spanning set interpretation. Each cell illustrates the Logit Lens of the gradient's spanning set according to a layer and a token from the edited prompt. According to Section 5.2 observation, for FF_1 we present the projections of the most probable tokens for the intermediate inputs x_i. For FF_2 we show most improbable tokens for the VJPs δ_i. The color indicates the norm of the gradient's δ_i. In this example, the prompt is "Lionel Messi plays for", to which the model responds with "Barcelona". The new target is "Paris". Notably, the target token "Paris" is evident through the majority of FF_2's VJPs, which are the vectors that are injected into FF_2's weights. On the other hand, FF_1 reflects not only the tokens that the gradients try to inject into FF_1, but also the model's intermediate predictions at each MLP layer during the forward pass. It is worth noting that the presence of tokens with a less clear meaning, such as "VIDIA" in FF_2 results from the projection of vectors with almost zero norm, as exampled in (Table 2). This implies that these subcomponents of the gradient exert negligible influence when updating the model.

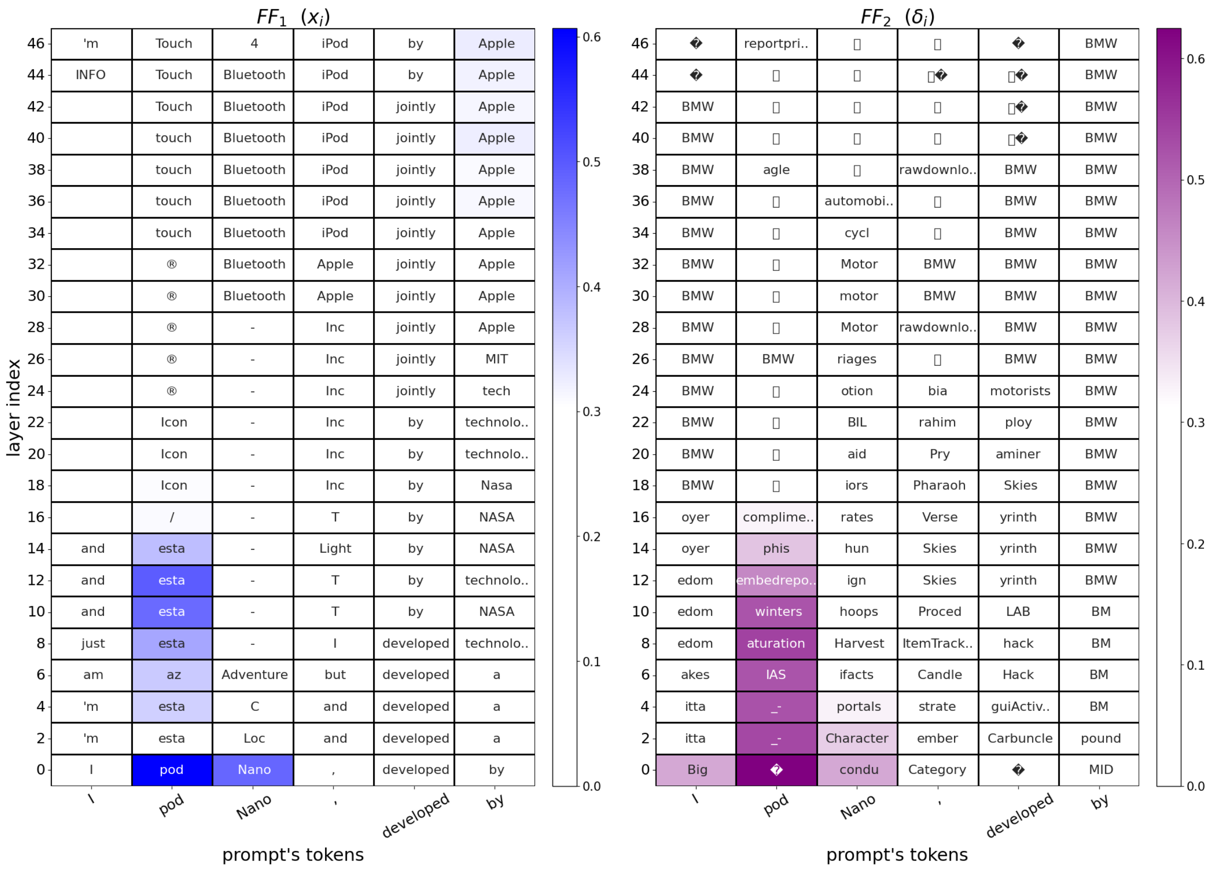
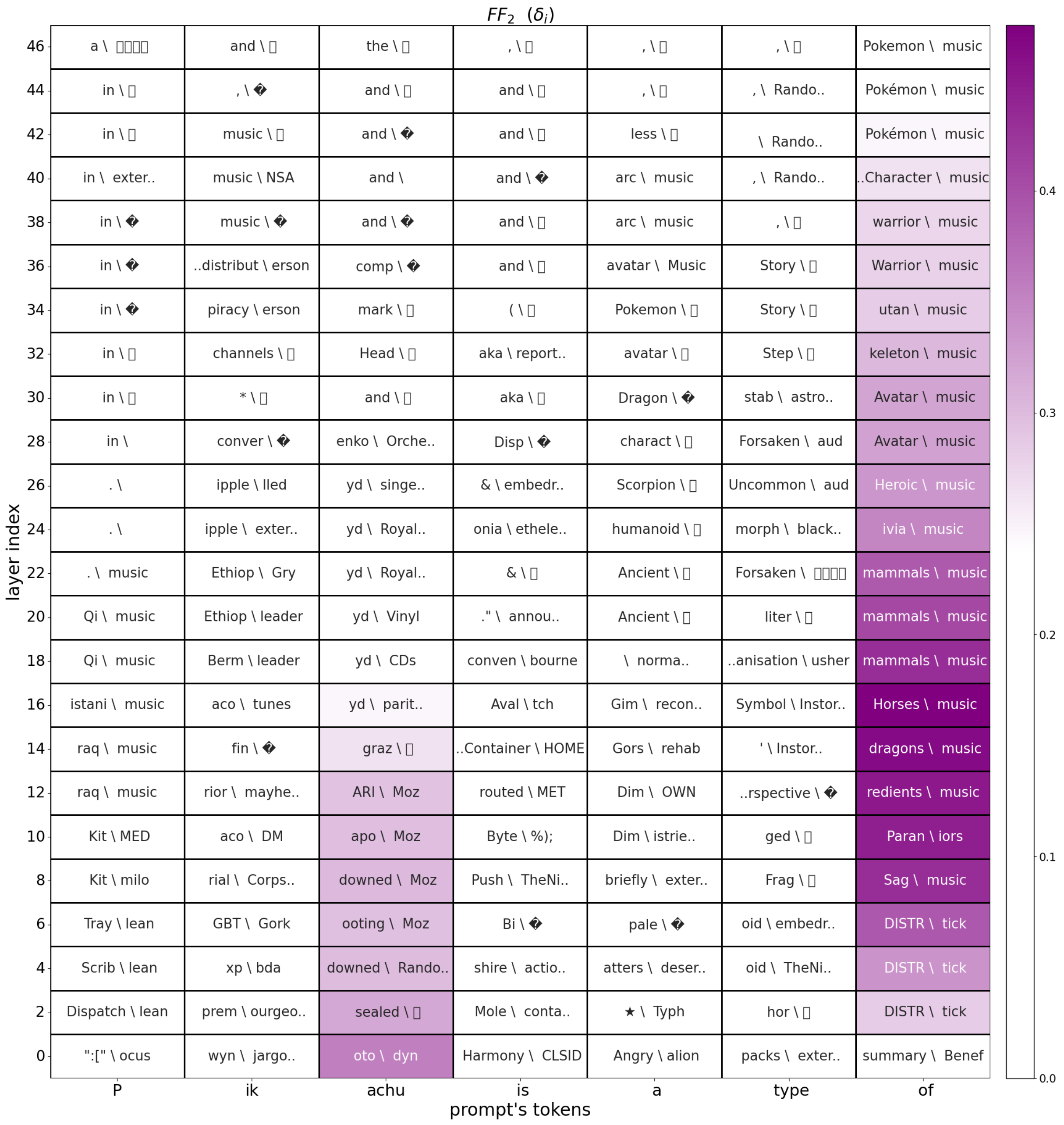
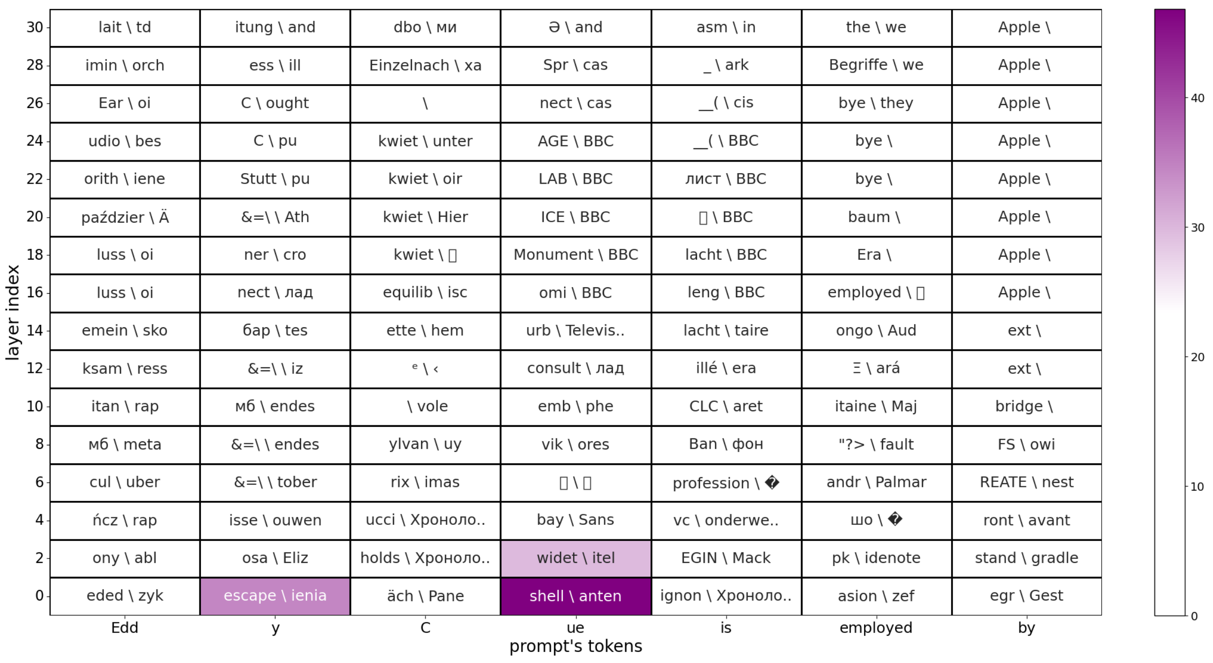


Figure 12: GPT2-xl MLP gradients via Logit Lens for "Ipod Nano, developed by" which we edit from "Apple" to "BMW". Due lack of space, we show only every second layer. The color scheme indicates that the primary focus of editing lies on the subject token "Pod". We hypothesize that the word "Ipod" is highly related to "Apple", so by targeting "pod", the gradients edit the relation between Ipod and the company that created it. Question marks and empty boxes are non-English tokens.



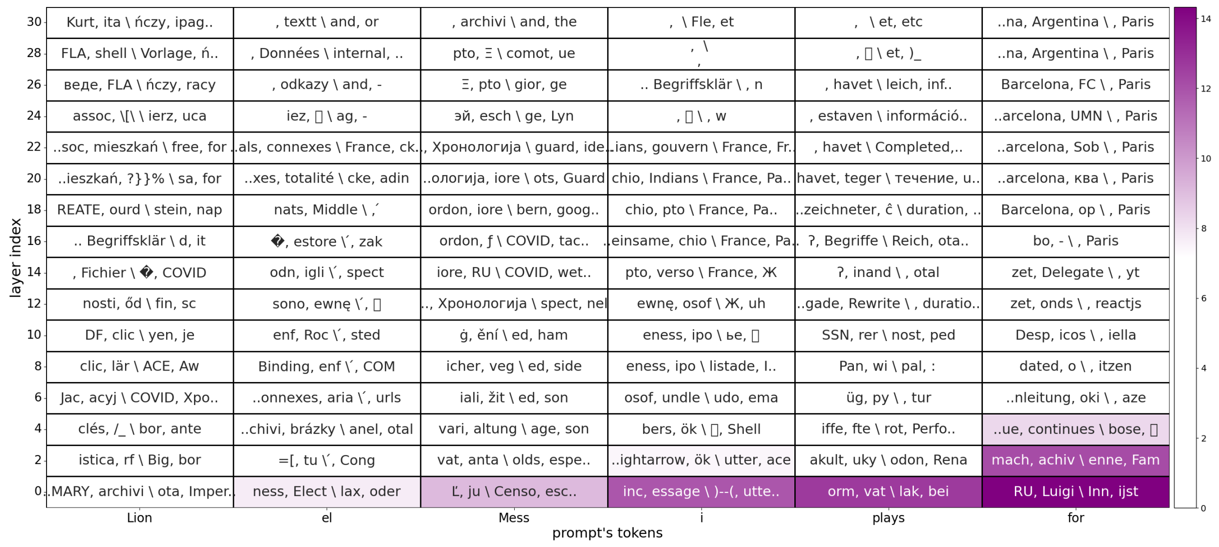
layer index	P	ik	achu	is	a	type	of
46	a \ □□□□	and \ □	the \ □	, \ □	, \ □	, \ □	Pokemon \ music
44	in \ □	, \ �	and \ □	and \ □	, \ □	, \ Rando..	Pokémon \ music
42	in \ □	music \ □	and \ �	and \ □	less \ □	\ Rando..	Pokémon \ music
40	in \ exter..	music \ NSA	and \	and \ �	arc \ music	, \ Rando..	..Character \ music
38	in \ �	music \ �	and \ �	and \ □	arc \ music	, \ □	warrior \ music
36	in \ �	..distribut \ erson	comp \ �	and \ □	avatar \ Music	Story \ □	Warrior \ music
34	in \ �	piracy \ erson	mark \ □	(\ □	Pokemon \ □	Story \ □	utan \ music
32	in \ □	channels \ □	Head \ □	aka \ report..	avatar \ □	Step \ □	keleton \ music
30	in \ □	* \ □	and \ □	aka \ □	Dragon \ �	stab \ astro..	Avatar \ music
28	in \	conver \ �	enko \ Orche..	Disp \ �	charact \ □	Forsaken \ aud	Avatar \ music
26	. \	ipple \ lled	yd \ singe..	& \ embedr..	Scorpion \ □	Uncommon \ aud	Heroic \ music
24	. \	ipple \ exter..	yd \ Royal..	onia \ ethele..	humanoid \ □	morph \ black..	ivia \ music
22	. \ music	Ethiop \ Gry	yd \ Royal..	& \ □	Ancient \ □	Forsaken \ □□□□	mammals \ music
20	Qi \ music	Ethiop \ leader	yd \ Vinyl	." \ annou..	Ancient \ □	liter \ □	mammals \ music
18	Qi \ music	Berm \ leader	yd \ CDs	conven \ bourne	\ norma..	..anisation \ usher	mammals \ music
16	istani \ music	aco \ tunes	yd \ parit..	Aval \ tch	Gim \ recon..	Symbol \ Instor..	Horses \ music
14	raq \ music	fin \ �	graz \ □	..Container \ HOME	Gors \ rehab	' \ Instor..	dragons \ music
12	raq \ music	rior \ mayhe..	ARI \ Moz	routed \ MET	Dim \ OWN	..rspective \ �	redients \ music
10	Kit \ MED	aco \ DM	apo \ Moz	Byte \ %);	Dim \ istrie..	ged \ □	Paran \ iors
8	Kit \ milo	rial \ Corps..	downed \ Moz	Push \ TheNi..	briefly \ exter..	Frag \ □	Sag \ music
6	Tray \ lean	GBT \ Gork	ooting \ Moz	Bi \ �	pale \ �	oid \ embedr..	DISTR \ tick
4	Scrib \ lean	xp \ bda	downed \ Rando..	shire \ actio..	atters \ deser..	oid \ TheNi..	DISTR \ tick
2	Dispatch \ lean	prem \ ourgeo..	sealed \ □	Mole \ conta..	★ \ Typh	hor \ □	DISTR \ tick
0	":[" \ ocus	wyn \ jargo..	oto \ dyn	Harmony \ CLSID	Angry \ alion	packs \ exter..	summary \ Benef

Figure 13: GPT2-xl FF_2 gradients via Logit Lens. The editing tries to change the prompt "Pikachu is a type of" to answer "music" instead of "Pokémon". Every cell shows the most $<$ probable \improbable $>$ tokens for its δ_i. The fact that we see FF_2's last token's project "Pokémon / music" means the edit tries to subtract from the model's weights the embedding of "Pokémon", while adding the embedding of "music" (same mechanism with "mammals / music"). Regarding the presence of many non-English tokens (question marks and empty boxes), in Appendix H, we present the same table with "Normalized Logit Lens", revealing the embedding of the target token "music".



layer index	Edd	y	C	ue	is	employed	by
30	lait \ td	itung \ and	dbo \ ми	ə \ and	asm \ in	the \ we	Apple \
28	imin \ orch	ess \ ill	Einzelnach \ ха	Spr \ cas	_ \ ark	Begriffe \ we	Apple \
26	Ear \ oi	C \ ought	\	nect \ cas	__(\ cis	bye \ they	Apple \
24	udio \ bes	C \ pu	kwiet \ unter	AGE \ BBC	__(\ BBC	bye \	Apple \
22	orith \ iene	Stutt \ pu	kwiet \ oir	LAB \ BBC	лист \ BBC	bye \	Apple \
20	paździer \ Ä	&=\ \ Ath	kwiet \ Hier	ICE \ BBC	□ \ BBC	baum \	Apple \
18	luss \ oi	ner \ cro	kwiet \ □	Monument \ BBC	lacht \ BBC	Era \	Apple \
16	luss \ oi	nect \ лад	equilib \ isc	omi \ BBC	leng \ BBC	employed \ □	Apple \
14	emein \ sko	бар \ tes	ette \ hem	urb \ Televis..	lacht \ taire	ongo \ Aud	ext \
12	ksam \ ress	&=\ \ iz	ᵉ \ ‹	consult \ лад	illé \ era	Ξ \ ará	ext \
10	itan \ rap	мб \ endes	\ vole	emb \ phe	CLC \ aret	itaine \ Maj	bridge \
8	мб \ meta	&=\ \ endes	ylvan \ uy	vik \ ores	Ban \ фон	"?> \ fault	FS \ owi
6	cul \ uber	&=\ \ tober	rix \ imas	□ \ □	profession \ �	andr \ Palmar	REATE \ nest
4	ńcz \ rap	isse \ ouwen	ucci \ Хроноло..	bay \ Sans	vc \ onderwe..	шо \ �	ront \ avant
2	ony \ abl	osa \ Eliz	holds \ Хроноло..	widet \ itel	EGIN \ Mack	pk \ idenote	stand \ gradle
0	eded \ zyk	escape \ ienia	äch \ Pane	shell \ anten	ignon \ Хроноло..	asion \ zef	egr \ Gest

Figure 14: Llama2-7B FF_2 gradients via Logit Lens. The editing tries to change the prompt "Eddy Cue is employed by" to answer "BBC" instead of "Apple" (template was taken from the CounterFact dataset). Every cell shows the most $<$ probable \improbable $>$ tokens for its δ_i. 3 main patterns can be observed from the table: (1) The part that has the largest VJPs by norm, which we assume is where most of the editing is done, is in the last subject token ("cu") and during the first few layers of the model. (2) We do not see the target token "BBC" in the most improbable projection of the last token; instead, the most improbable one is the empty token "". When we examined these projections we found "BBC" to be only the third most improbable token. However, in other VJPs we notice this is indeed the most improbable projection. (3) The original answer of the model, "Apple", frequently emerges in the projections of the last token. As outlined in Section 5.2, given that we update the model with a negative learning rate, when the VJPs Logit Lens projection ranks a token as highly probable, subtracting the gradients from the model's weights equals to subtracting the embedding of this probable token from them . Consequently, this pattern illustrates the mechanism of "shift" within the context of "imprint and shift": We decrease the probability associated with "Apple" in pursuit of increasing the probability of "BBC".



layer index	Lion	el	Mess	i	plays	for
30	Kurt, ita \ ńczy, ipag..	, textt \ and, or	, archivi \ and, the	, \ Fle, et	, \ et, etc	..na, Argentina \ , Paris
28	FLA, shell \ Vorlage, ń..	, Données \ internal, ..	pto, Ξ \ comot, ue	, \ ,	, □ \ et,)_	..na, Argentina \ , Paris
26	веде, FLA \ ńczy, racy	, odkazy \ and, -	Ξ, pto \ gior, ge	.. Begriffsklär \ , n	, havet \ leich, inf..	Barcelona, FC \ , Paris
24	assoc, \[\ \ ierz, uca	iez, □ \ ag, -	эй, esch \ ge, Lyn	, □ \ , w	, estaven \ információ..	..arcelona, UMN \ , Paris
22	..soc, mieszkań \ free, for	..als, connexes \ France, ck..	.., Хронологија \ guard, ide..	..ians, gouvern \ France, Fr..	, havet \ Completed,..	..arcelona, Sob \ , Paris
20	..ieszkań, ?}}% \ sa, for	..xes, totalité \ cke, adin	..ологија, iore \ ots, Guard	chio, Indians \ France, Pa..	havet, teger \ течение, u..	..arcelona, ква \ , Paris
18	REATE, ourd \ stein, nap	nats, Middle \ ,´	ordon, iore \ bern, goog..	chio, pto \ France, Pa..	..zeichneter, ĉ \ duration, ..	Barcelona, op \ , Paris
16	.. Begriffsklär \ d, it	�, estore \´, zak	ordon, ʃ \ COVID, tac..	..einsame, chio \ France, Pa..	ʔ, Begriffe \ Reich, ota..	bo, - \ , Paris
14	, Fichier \ �, COVID	odn, igli \´, spect	iore, RU \ COVID, wet..	pto, verso \ France, Ж	ʔ, inand \ , otal	zet, Delegate \ , yt
12	nosti, őd \ fin, sc	sono, ewnę \´, □	.., Хронологија \ spect, nel	ewnę, osof \ Ж, uh	..gade, Rewrite \ , duratio..	zet, onds \ , reactjs
10	DF, clic \ yen, je	enf, Roc \´, sted	ġ, ění \ ed, ham	eness, ipo \ ье, □	SSN, rer \ nost, ped	Desp, icos \ , iella
8	clic, lär \ ACE, Aw	Binding, enf \´, COM	icher, veg \ ed, side	eness, ipo \ listade, I..	Pan, wi \ pal, :	dated, o \ , itzen
6	Jac, acyj \ COVID, Xpo..	..onnexes, aria \´, urls	iali, žit \ ed, son	osof, undle \ udo, ema	üg, py \ , tur	..nleitung, oki \ , aze
4	clés, /_ \ bor, ante	..chivi, brázky \ anel, otal	vari, altung \ age, son	bers, ök \ □, Shell	iffe, fte \ rot, Perfo..	..ue, continues \ bose, □
2	istica, rf \ Big, bor	=[, tu \´, Cong	vat, anta \ olds, espe..	..ightarrow, ök \ utter, ace	akult, uky \ odon, Rena	mach, achiv \ enne, Fam
0	..MARY, archivi \ ota, Imper..	ness, Elect \ lax, oder	Ľ, ju \ Censo, esc..	inc, essage \ )--(, utte..	orm, vat \ lak, bei	RU, Luigi \ Inn, ijst

Figure 15: Llama2-7B FF_2 gradients via Logit Lens. The editing tries to change the prompt "Lionel Messi plays for" to answer "Paris" rather than "Barcelona". Every cell shows the **2 most probable and improbable tokens** of the VJPs δ_i in the format of $<$ probable \improbable $>$. The target token, "Paris" is the second most improbable token in the projections of the last prompt's token ("for"), only second to the empty token (""). The projection's most probable tokens of the last prompt's token are "Barcelona" (the final prediction of the model for this prompt), with some projections also revealing "Argentina" (which was the second most probable prediction of the model). In addition, the projections of the last subject token from the prompt ("i") show that among the most improbable tokens there are appearances of "France", which can be associated to its capital, "Paris". Together, these projections illustrate how backpropogation editing tries to add into the model's FF_2's neurons (weights) the embedding of "Paris" (and related concepts, e.g. "France") while removing the embeddings of "Barcelona" and "Argentina".

F Additional Empirical Results

F.1 Impact of Different Segments of the Prompt in Every Layer

In Section 6, we elucidate how comparing the norm of each VJP (δ_i) involved in the construction of a gradient matrix can reflect which layers are updated more than others. Similar comparison is conducted for the tokens from the edited prompt, uncovering that certain layers and segmentation from the prompt do not contribute significantly to the updating process.

We extend this analysis from Section 6 to demonstrate the ability to identify the main editing matrices in every type of module in LMs Figure 16.

We will solely discuss the results related to the MLP layers, as we did not examine the attention modules in our study. Both the FF_1 module (*mlp.c_fc*) and FF_2 module (*mlp.c_proj*) demonstrate that the primary editing occurs around the first quarter of layers by the last subject token, and around the middle of the layers by the last token. The majority of the other layers and tokens exhibit VJP norms close to zero, indicating that they scarcely contribute to updating the model's weights. The same pattern is also evident for the VJPs between transformer layers (*transformer.h*), as, if we disregard Dropouts and Layer Norms, the VJPs of FF_2 are the same as the one between the transformer layers.

We also provide similar figures where, instead of plotting the norm of the VJPs, we compare the norms of the intermediate inputs to every layer (x_i) (Figure 17). The inclusion of those figures is solely to emphasize that there is no correlation between the norm of x_i and δ_i.

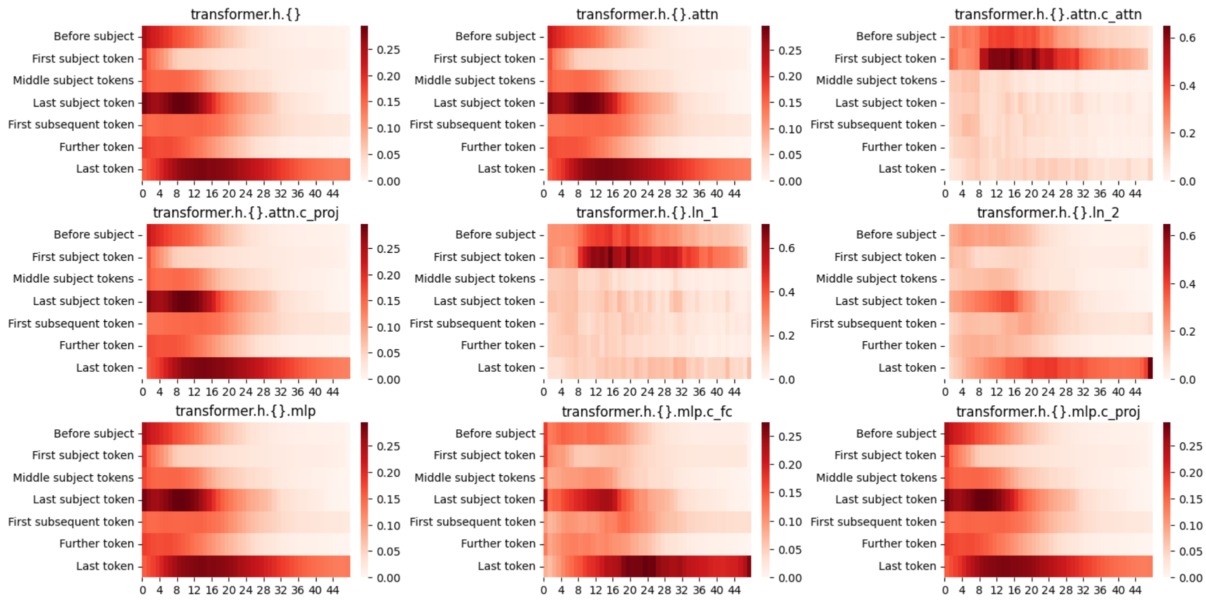


Figure 16: δ_i norm for each of GPT2-xl sub-module (names are according to the HuggingFace implementation). This is an extension to Section 6, showing how our approach is easily adapted to any submodule of the model. The layer of FF_1 is represented by *mlp.c_fc* and FF_2 by *mlp.c_proj* .

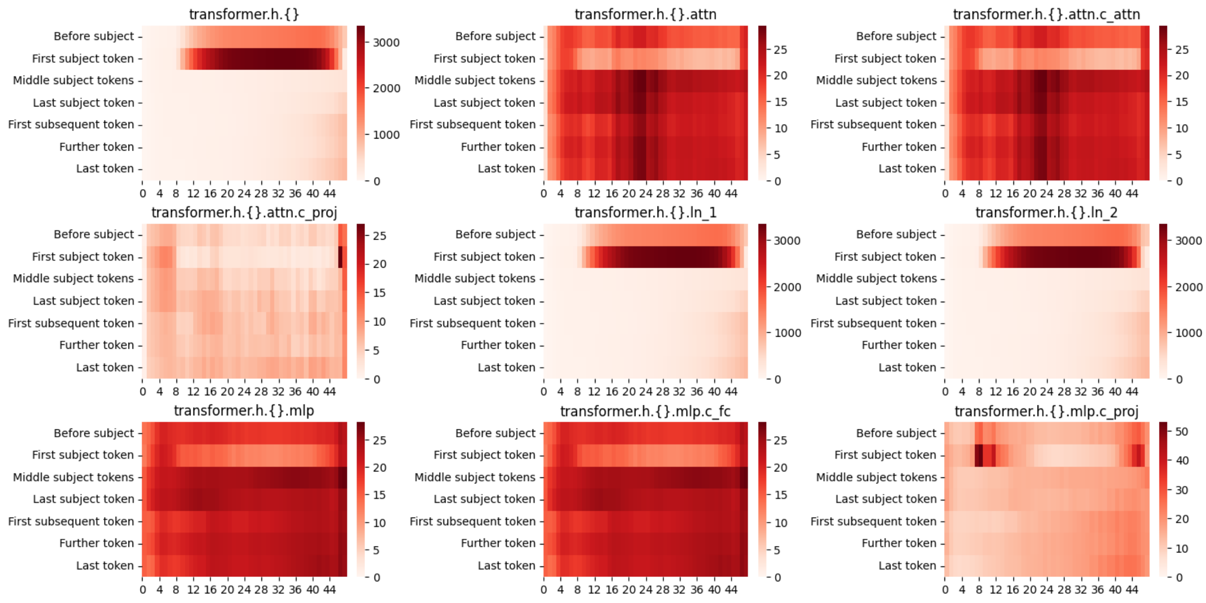


Figure 17: x_i norm for GPT2-xl. Provided to show no correlation with Figure 16.

F.2 The Ranks of FF_1 and the Models' Original Answer

In Section 6 we examine how FF_2's VJPs rank the target tokens, the tokens we try to learn during backpropagation. One possible interpretation of our analysis is to examine the backward pass according to the gradual change in the embeddings it tries to inject (add or subtract) into the model. Previous works conduct similar analysis with the hidden states from the forward pass Haviv et al. (2023); Geva et al. (2022b). Those works examined the gradual build in LMs' forward pass prediction, from the perspective of the last token in the edited prompts. In this section, we expand upon these examinations by measuring the LL rank for the original answers outputted by the forward pass (before editing), and by observing these LL ranks from the perspective of FF_1's spanning set.

The presented results are based on 100 distinct edits using a single backward pass per edit. We employ GPT2 and Llama2-7B, and the edited prompts and targets were taken from the CounterFact dataset.

FF_2 : In Section 6 we presented the ranking of the target token for GPT2-xl. Subsequently, we present the rank of the last prompt's token VJP (annotated with δ_n in Lemma 5.1) also for GPT2-medium and Llama-7B. In Figure 18 we can see that all models' LL projections rank the target token as one of the most improbable tokens along most of the layers, with some degradation in the first few layers. We associate this drop to the gap of LL in projection vectors from earlier layers.

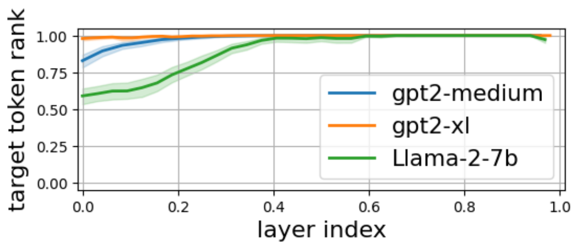


Figure 18: The Logit Lens rank of the target token in VJP of the last token from the edited prompt, δ_n. In order to show different models in the same figure, we normalize the layer indices and the rank of the token in the vocabularies (which is 50K for GPT2 and 32K for Llama2). All models' δ_n assign the target token as one of the most improbable tokens along most of their layers.

We repeat the same experiment and measure the rank of the original token predicted by the model. According to Equation 4 and Lemma 5.1, the embedding of a token in the gradient should be discernible if it is the target token or if its probability in the model's final output is relatively high.

The results depicted in Figure 19 illustrate that the LL rank of the final answer is relatively low in the model's last layers, although not at the lowest possible level.

In Section 4.2 and Section 5.2, we delved into how the injection of the gradient vector with a high LL rank of the target token implies that the update aims to enhance the target probability in the output. Similarly, the observed pattern regarding the LL rank of the model's final answer suggests that the updates attempt to diminish the probabilities of the final answers in the model's output, but in a much smoother manner than those of the target token.

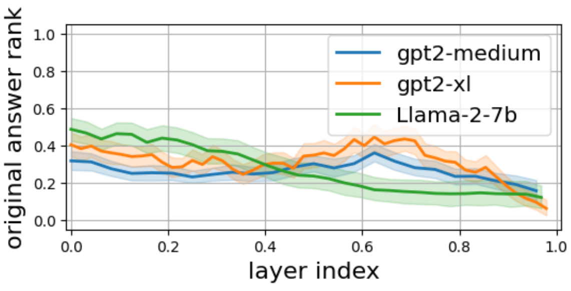
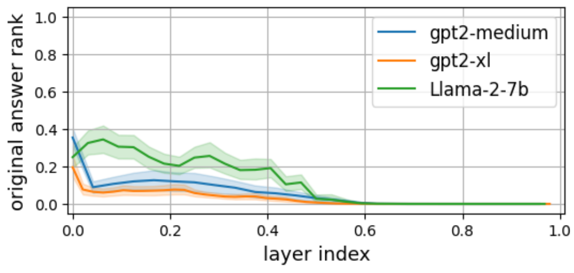


Figure 19: The Logit Lens ranks for the models' actual answers according to FF_2 gradients of the last prompts' token (layer indices and ranks are normalized). In the later layers, the rank of the original answers suggests that the model subtracts the embedding of those tokens from the model's weights.

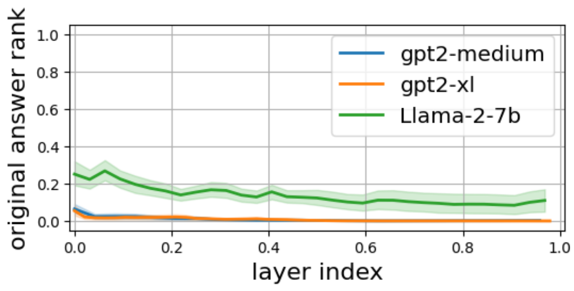
FF_1 : Since this layer's δ_i is not projectable via LL, we use its inputs x_i as its spanning set. Analyzing the ranks of x_i is almost identical to the analysis of Haviv et al. (2023). We share our analysis in Figure 20, mainly to emphasize that gradients write into FF_1's weights the inputs x_i from the forward pass. The gradual build in the models' predictions becomes more apparent when we filter out examples where the model answers CounterFact's prompts incorrectly, accounting for approximately 84% of the instances with GPT2-medium/xl and Llama.[12] In Figure 20, we also included the same analysis with 50 correctly answered prompts.

[1]Most of the time, they predict tokens like "a", "is", and "the", rather than factual notions in the context of the prompts.

[2]Llama utilizes two matrices that employ FF_1 as part of its implementation of SwiGLU as its MLP activation function. Notably, the input x_i remains consistent for both matrices.



(a) Only correctly answered prompts



(b) Correctly and incorrectly answered prompts

Figure 20: FF_1's x_i ranks for the model's actual answers. The main difference between the two figures is that when the models answer incorrectly, they mostly answer with function words, such as "a" and "the", which seems to have a constant rank from the earlier layers, while answering the actual factual answers changes the answers' rank gradually in the first half of the layers. The meaning of these graphs is that during fine-tuning, the embeddings injected into the models' weights are those of x_i.

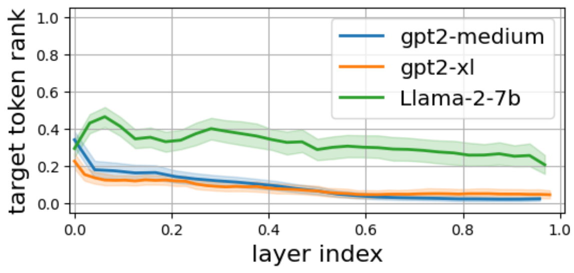


Figure 21: FF_1's x_i ranks for the edits' targets, reflecting the intermediate predictions the models give to them at each layer.

Throughout our study, this is the only result we found to be affected by the distinction whether the models answer the original prompt correctly or incorrectly.

Similarly, when we plot the ranks of the target token for each layer, we only observe the forward passes' intermediate probability for the target token (Figure 21).

G The Gradual Change of the VJPs Across Layers

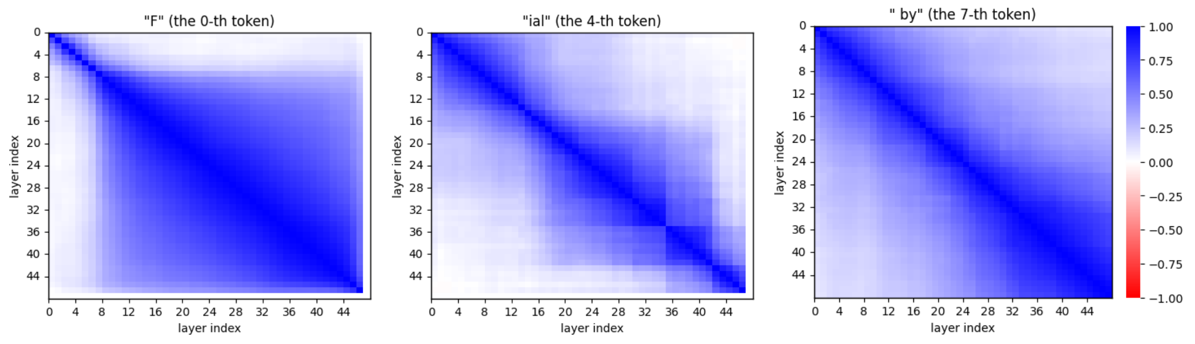
The results in Section F.2 reveal the presence of the embedded target within the MLP's FF_2 VJPs. To further extend this examination, we measure the similarity of the VJPs across different layers. Specifically, for two layers i and j, we calculate the cosine similarity between their VJPs for given token t, denoted as: $cosine(\delta_t^i, \delta_t^j)$. In particular, we are interested in the FF_2's VJPs since during the backward pass these VJPs are read by each FF_2 matrix from the residual stream. This allows us to explore how the gradient information propagates and evolves across layers.

We use the setup from Section F.2, with GPT2-xl, and visualize the results using a heat-map for each token. For the t token, the (i, j) cell of its map correspond to $cosine(\delta_t^i, \delta_t^j)$. An example for a single prompt is shown in Figure 22.

The VJP of the residual stream exhibits a cosine similarity score of approximately 0.4 over a sequence of 3-5 blocks for all tokens across various prompts.

Specifically, for the last token of each prompt, we calculate the averaged cosine similarity across 100 distinct edits. As shown in Figure 23, the cosine similarity score exceeds 0.7 across 10 blocks, indicating a strong alignment of VJPs between layers. Even when comparing the first layer (layer 0) to the last (layer 47), we observe an average cosine similarity above 0.08, which is relatively high score compared to random vectors in an embedding space of size 1600.

These results illustrate how the information from the initial VJP, originating in the model's final layer, permeates earlier layers and how gradually each layer modifies the residual VJP.



(a) first subject token (b) last subject token (c) the prompt last token

Figure 22: The cosine similarity between the residual VJPs (FF_2s' VJPs) of GPT-2 xl is examined. The provided maps are generated from editing the prompt "Ferrari Mondial, created by" with the target "Nintendo".

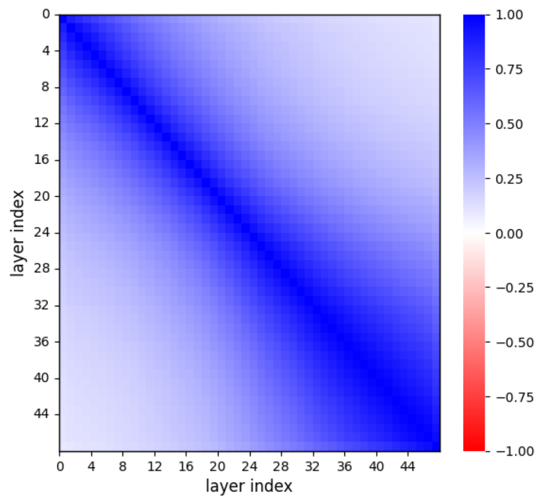


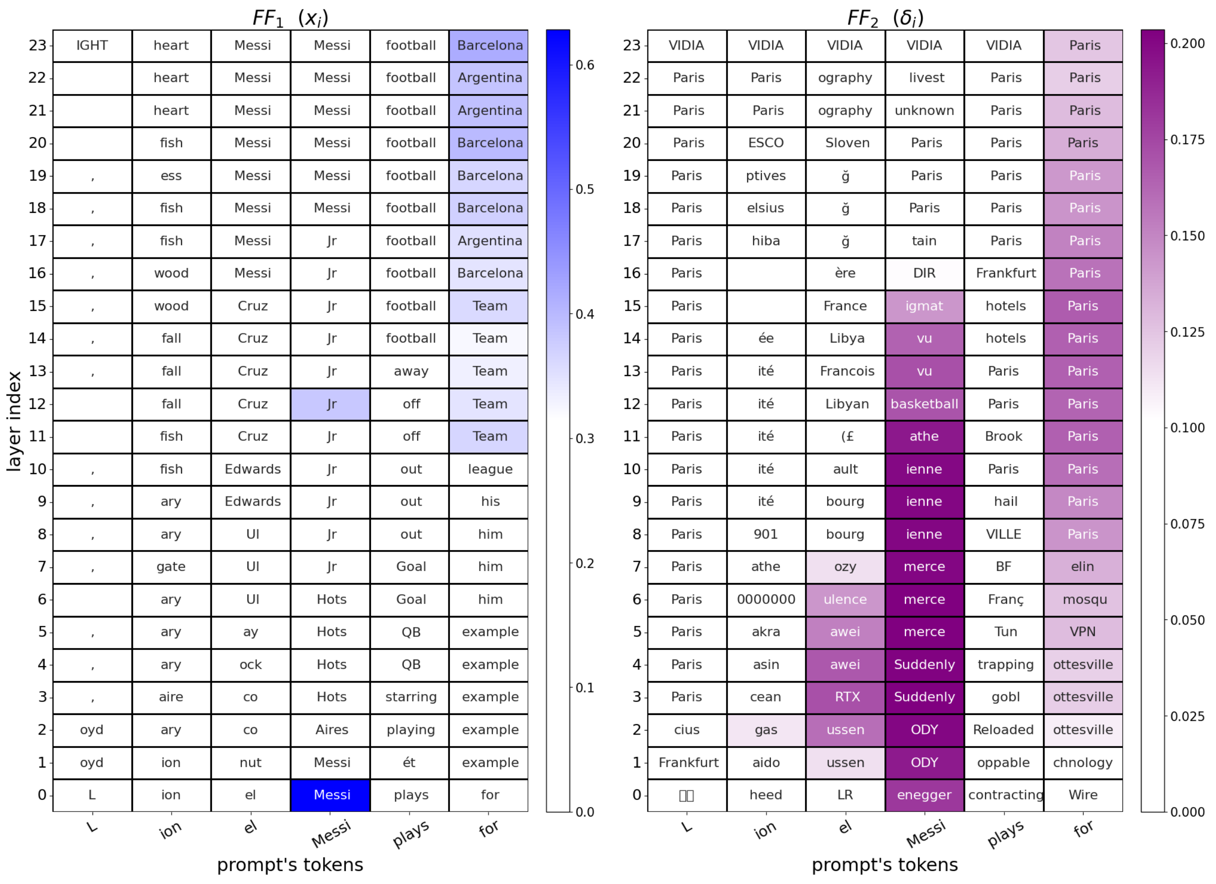
Figure 23: The cosine similarity between the residual VJPs of all GPT2-xl layers for the last token of a given edit, averaged across 100 distinct edits, shows a consistent alignment across layers.

H Normalized Logit Lens

We acknowledge the sensitivity of the Logit Lens to low-norm vectors. With vectors with close to zero norms, the tokens LL project tokens that resemble the projection of the zero vector. In Section 6, we discussed that certain VJPs, δ_i, exhibit low norms. We hypothesize that the norms of the VJPs reflect which parts of speech and layers the backward pass tries to edit more. We were interested in observing which tokens could be projected from gradients if we isolate the influence of these low norms. Our investigation reveals that normalizing the projected vectors before applying the Logit Lens can be an appropriate solution to the variance in VJPs' norms. We named this method "Normalized Logit Lens".

A place we consider using this normalization is in creating the tables of Appendix E. In Figure 24 and Figure 25 we share two examples that replicate the setup from Appendix E except we are using Normalized Logit Lens.

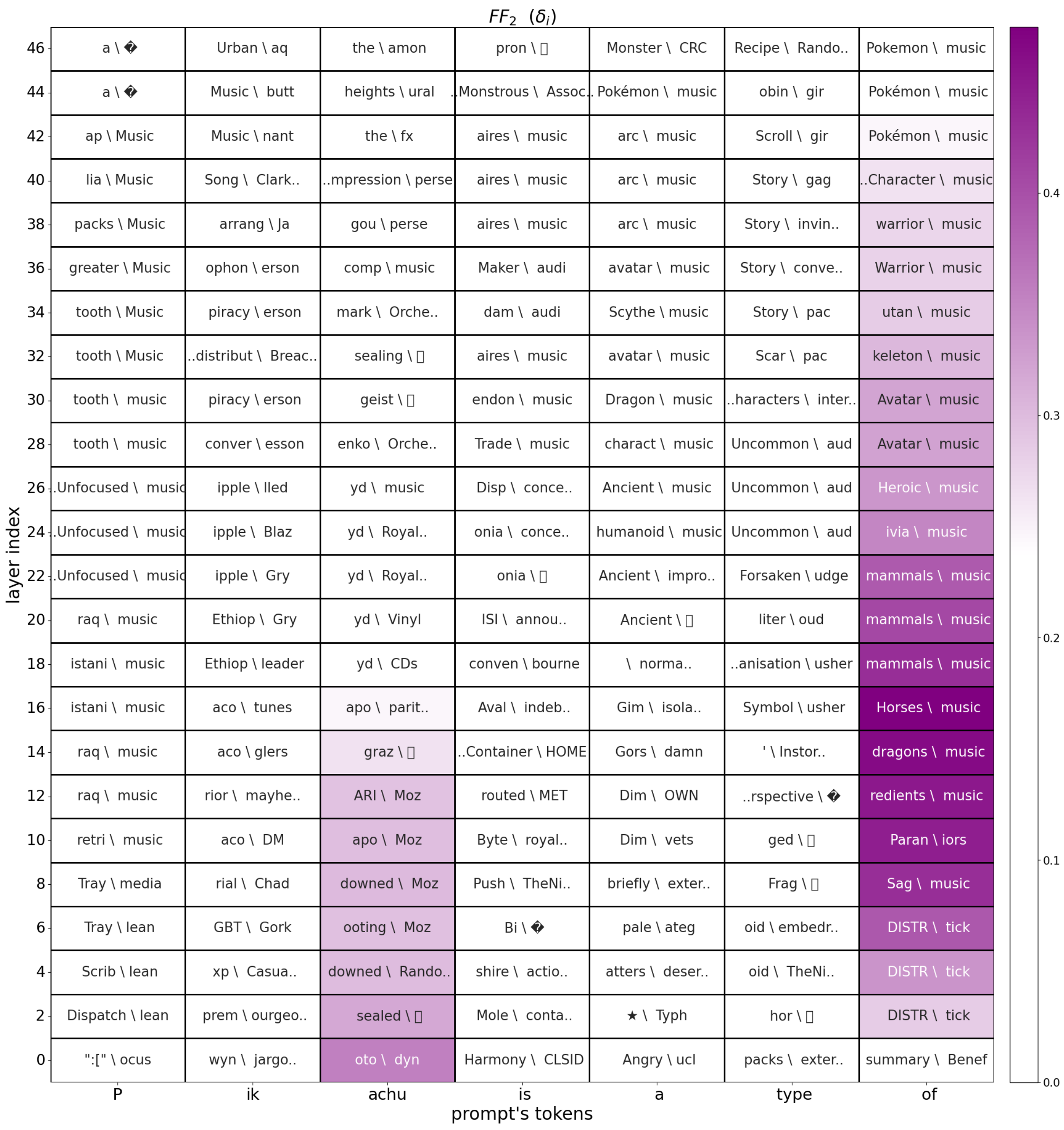
We provide this section to highlight the pattern we already mentioned in Section 6: that most of FF_2 gradients ranks the target token as one the most improbable token. We also want to suggest further work to consider using Normalized Logit Lens to examine low-norm vectors.



layer index	L	ion	el	Messi	plays	for
23	IGHT	heart	Messi	Messi	football	Barcelona
22		heart	Messi	Messi	football	Argentina
21		heart	Messi	Messi	football	Argentina
20		fish	Messi	Messi	football	Barcelona
19	,	ess	Messi	Messi	football	Barcelona
18	,	fish	Messi	Messi	football	Barcelona
17	,	fish	Messi	Jr	football	Argentina
16	,	wood	Messi	Jr	football	Barcelona
15	,	wood	Cruz	Jr	football	Team
14	,	fall	Cruz	Jr	football	Team
13	,	fall	Cruz	Jr	away	Team
12		fall	Cruz	Jr	off	Team
11		fish	Cruz	Jr	off	Team
10	,	fish	Edwards	Jr	out	league
9	,	ary	Edwards	Jr	out	his
8	,	ary	Ul	Jr	out	him
7	,	gate	Ul	Jr	Goal	him
6		ary	Ul	Hots	Goal	him
5	,	ary	ay	Hots	QB	example
4	,	ary	ock	Hots	QB	example
3	,	aire	co	Hots	starring	example
2	oyd	ary	co	Aires	playing	example
1	oyd	ion	nut	Messi	ét	example
0	L	ion	el	Messi	plays	for

layer index	L	ion	el	Messi	plays	for
23	VIDIA	VIDIA	VIDIA	VIDIA	VIDIA	Paris
22	Paris	Paris	ography	livest	Paris	Paris
21	Paris	Paris	ography	unknown	Paris	Paris
20	Paris	ESCO	Sloven	Paris	Paris	Paris
19	Paris	ptives	ğ	Paris	Paris	Paris
18	Paris	elsius	ğ	Paris	Paris	Paris
17	Paris	hiba	ğ	tain	Paris	Paris
16	Paris		ère	DIR	Frankfurt	Paris
15	Paris		France	igmat	hotels	Paris
14	Paris	ée	Libya	vu	hotels	Paris
13	Paris	ité	Francois	vu	Paris	Paris
12	Paris	ité	Libyan	basketball	Paris	Paris
11	Paris	ité	(£	athe	Brook	Paris
10	Paris	ité	ault	ienne	Paris	Paris
9	Paris	ité	bourg	ienne	hail	Paris
8	Paris	901	bourg	ienne	VILLE	Paris
7	Paris	athe	ozy	merce	BF	elin
6	Paris	0000000	ulence	merce	Franç	mosqu
5	Paris	akra	awei	merce	Tun	VPN
4	Paris	asin	awei	Suddenly	trapping	ottesville
3	Paris	cean	RTX	Suddenly	gobl	ottesville
2	cius	gas	ussen	ODY	Reloaded	ottesville
1	Frankfurt	aido	ussen	ODY	oppable	chnology
0	□□	heed	LR	enegger	contracting	Wire

Figure 24: GPT2-medium MLP gradients projections with **Normalized Logit Lens**. Compared to the naive Logit Lens Figure 11, we see more cells in FF_2 that project concepts similar to the target token "Paris" and less unclear tokens, such as "VIDIA" (notice the last layers or the column of the token"ion"). The coloring is according to the original norm of each representative vector (before normalization).



layer index	P	ik	achu	is	a	type	of
46	a \ �	Urban \ aq	the \ amon	pron \ ▯	Monster \ CRC	Recipe \ Rando..	Pokemon \ music
44	a \ �	Music \ butt	heights \ ural	..Monstrous \ Assoc..	Pokémon \ music	obin \ gir	Pokémon \ music
42	ap \ Music	Music \ nant	the \ fx	aires \ music	arc \ music	Scroll \ gir	Pokémon \ music
40	lia \ Music	Song \ Clark..	..mpression \ perse	aires \ music	arc \ music	Story \ gag	..Character \ music
38	packs \ Music	arrang \ Ja	gou \ perse	aires \ music	arc \ music	Story \ invin..	warrior \ music
36	greater \ Music	ophon \ erson	comp \ music	Maker \ audi	avatar \ music	Story \ conve..	Warrior \ music
34	tooth \ Music	piracy \ erson	mark \ Orche..	dam \ audi	Scythe \ music	Story \ pac	utan \ music
32	tooth \ Music	..distribut \ Breac..	sealing \ ▯	aires \ music	avatar \ music	Scar \ pac	keleton \ music
30	tooth \ music	piracy \ erson	geist \ ▯	endon \ music	Dragon \ music	..haracters \ inter..	Avatar \ music
28	tooth \ music	conver \ esson	enko \ Orche..	Trade \ music	charact \ music	Uncommon \ aud	Avatar \ music
26	..Unfocused \ music	ipple \ lled	yd \ music	Disp \ conce..	Ancient \ music	Uncommon \ aud	Heroic \ music
24	..Unfocused \ music	ipple \ Blaz	yd \ Royal..	onia \ conce..	humanoid \ music	Uncommon \ aud	ivia \ music
22	..Unfocused \ music	ipple \ Gry	yd \ Royal..	onia \ ▯	Ancient \ impro..	Forsaken \ udge	mammals \ music
20	raq \ music	Ethiop \ Gry	yd \ Vinyl	ISI \ annou..	Ancient \ ▯	liter \ oud	mammals \ music
18	istani \ music	Ethiop \ leader	yd \ CDs	conven \ bourne	\ norma..	..anisation \ usher	mammals \ music
16	istani \ music	aco \ tunes	apo \ parit..	Aval \ indeb..	Gim \ isola..	Symbol \ usher	Horses \ music
14	raq \ music	aco \ glers	graz \ ▯	..Container \ HOME	Gors \ damn	' \ Instor..	dragons \ music
12	raq \ music	rior \ mayhe..	ARI \ Moz	routed \ MET	Dim \ OWN	..rspective \ �	redients \ music
10	retri \ music	aco \ DM	apo \ Moz	Byte \ royal..	Dim \ vets	ged \ ▯	Paran \ iors
8	Tray \ media	rial \ Chad	downed \ Moz	Push \ TheNi..	briefly \ exter..	Frag \ ▯	Sag \ music
6	Tray \ lean	GBT \ Gork	ooting \ Moz	Bi \ �	pale \ ateg	oid \ embedr..	DISTR \ tick
4	Scrib \ lean	xp \ Casua..	downed \ Rando..	shire \ actio..	atters \ deser..	oid \ TheNi..	DISTR \ tick
2	Dispatch \ lean	prem \ ourgeo..	sealed \ ▯	Mole \ conta..	★ \ Typh	hor \ ▯	DISTR \ tick
0	":[" \ ocus	wyn \ jargo..	oto \ dyn	Harmony \ CLSID	Angry \ ucl	packs \ exter..	summary \ Benef

Figure 25: GPT2-xl FF_2 gradients via our spanning set interpretation and **normalized Logit Lens**. Every cell shows the most $<$ probable \improbable $>$ tokens for its δ_i. Compared to Figure 13, we observe fewer non-English tokens in the projections and more tokens associated with the target token "music".

I Editing Based on the "Shift" Mechanism

In Section 7 we introduce "forward pass shifting", a method for LM knowledge editing through the approximation of the gradient matrix. In this section, we provide additional results and implementation details. We want to remind that our intention is not to propose a new alternative editing method, but rather to explore the concept of injecting knowledge into the model in an approximate manner to how backpropagation operates.

I.1 Comparison with Naive Backpropagation

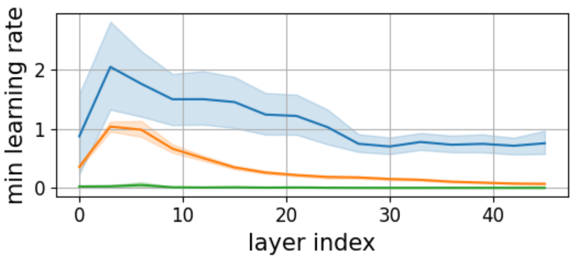
We check our method's ability to perform a single edit at a time of a given prompt, such that after editing, the most probable answer will be a selected target token. Our baselines for comparison are performing the same single Backpropagation editing with SGD and Adam. We use 50 samples from CounterFact (Meng et al., 2022) for the edited prompts and target. For each editing method and sample, we measure the method's sensitivity to different learning rates by checking a range of possible values and finding the minimum one to achieve a successful edit. We also monitor the model's general degradation after its update, using perplexity on maximum 256-tokens-long samples from WikiText (Merity et al., 2016). The findings are presented in Figure 26. Evidently, our approach's minimum learning rate shows less variation compared to SGD. Furthermore, its stability to editing, as indicated by perplexity, surpasses that of Adam and SGD in earlier layers and is identical in the latter ones.

I.2 Benchmark Implementation Details

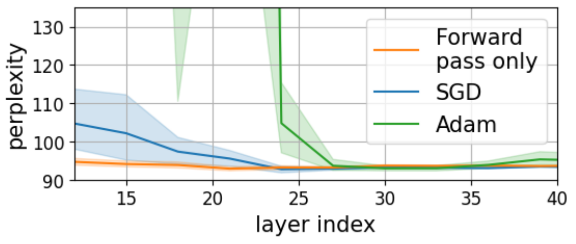
Based on the results from Section I.1, we identified layers 30–40 as the best potential editing layers. As for the learning rate, we examined values ranging from 0.14 to 0.26. The presented results use layer 35 with a learning rate of 0.24, which yielded the best results in the following benchmark.

The construction of the approximated gradient matrix is accomplished using the embedding of the target token. This embedding is obtained from the decoding matrix. If the target token consists of more than one token according to the model's vocabulary, we select the prefix (the first token) to construct the target token.

Regarding the other editing methods and CounterFact benchmark implementation, we followed



(a) Minimum learning rate for achieving a successful edit



(b) Perplexity at minimum learning rate editing

Figure 26: Editing only a single FF_2 matrix of GPT2-xl on CounterFact. By measuring a range of possible learning rate values we found the minimum that achieves a successful edit and measured the change in the model's general knowledge through perplexity on WikiText.

the implementations provided by Meng et al. (2023) to create the results ourselves. The post-editing metrics included in this benchmark, which we present, are as follows: (1) "Efficacy" - the accuracy (percentage of times) with which the model predicts the targeted token as the most probable one given the editing prompt. (2) "Paraphrase" - the accuracy of the model to answer paraphrases of the edited prompt (also known as "Generalization"). (3) "Neighborhood" - the accuracy of the model on prompts from similar domains as the edited prompt, which we do not wish to change (also known as "Specificity"). (4) "N-gram entropy" measures the weighted average of bi- and tri-gram entropies, reflecting fluency level. In addition, we included in the benchmark the perplexity score of the 100 first sentences from WikiText (version wikitext-2-raw-v1, test split[3]) that are at least 30 characters long. Moreover, each prompt was truncated to its first 256 tokens. The score of this metric reflects how much the model's original ability to generate text has changed.

The comparison was conducted on 1000 samples from CounterFact, and we benchmarked against state-of-the-art methods, such as ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and MEND (Mitchell et al., 2021).

[3]https://huggingface.co/datasets/wikitext

METHOD	EFFICACY ↑	PARAPHRASE ↑	NEIGHBORHOOD ↑
ORIGINAL MODEL	0.4± 6.31	0.4± 4.45	11.21± 19.77
FINE-TUNING (MLP 0)	96.37± 18.7	7.46± 19.44	10.12± 18.02
FINE-TUNING (MLP 35)	100.0± 0.0	46.1± 40.25	5.59± 13.19
MEND	71.4± 45.19	17.6± 31.47	7.73± 16.27
ROME	99.4± 7.72	71.9± 37.22	10.91± 19.53
MEMIT	79.4± 40.44	40.7± 41.64	10.98± 19.44
FORWARD PASS SHIFT	99.4± 7.72	41.55± 41.67	6.02± 14.0

Table 6: Single editing results of GPT2-xl on CounterFact benchmark

METHOD	N-GRAM ENTROPY ↑	WIKITEXT PERPLEXITY ↓
ORIGINAL MODEL	626.94± 11.56	93.56± 0.0
FINE-TUNING (MLP 0)	618.81± 52.74	199.84± 1392.66
FINE-TUNING (MLP 35)	618.5± 28.57	103.42± 11.69
MEND	623.94± 23.14	94.96± 0.66
ROME	622.78± 20.54	137.38± 786.35
MEMIT	627.18± 12.29	93.6± 0.11
FORWARD PASS SHIFT	622.45± 21.46	93.66± 1.08

Table 7: Examining the general ability of GPT2-xl to generate text after it has been applied with a single edit from CounterFact.

I.3 Results

In Table 6, we present the editing results regarding the model's ability to modify internal knowledge. In Table 7, we analyze the effect of editing on the model's ability to generate generic text.

If we filter out editing methods that cause drastic degradation to the model's ability to generate text (where n-gram entropy falls below 620), our method achieves the best editing results on the edited prompt (Accuracy), tying only with ROME. With regards to editing the paraphrased prompts, the "forward pass shifting" method achieves comparable results, but falls behind ROME. However, one of our method's limitations lies in its performance with neighborhood prompts, where the edited model altered prompts we do not anticipate to be changed.

I.4 Discussion

"Forward pass shift" demonstrates successful knowledge editing compared to methods that employ much more complex implementations. Additionally, our method shows minimal impact on the model's ability to generate text, addressing one of the main challenges of fine-tuning in precise knowledge editing.

From our understanding, "forward pass shift" is the simplest editing method in terms of algorithm complexity, as it only requires a single forward pass. The low score of this method in terms of neighborhood editing may be attributed to mistakenly editing activation patterns of FF_2 that are also shared with similar prompts. Future studies could explore the capability of "forward pass shift" to handle multiple edits at once, or to incorporate multiple iterations in its implementation (multiple forward passes).