Correcting Language Model Outputs by Editing Salient Layers

Kshitij Mishra*+
Indian Institute of Technology
mishra.kshitij07@gmail.com

Tamer Soliman*
Amazon Generative AI Center
tsoliman@amazon.com

Anil Ramakrishna
Amazon AGI Foundations
aniramak@amazon.com

Anoop Kumar
Amazon AGI Foundations
anooamzn@amazon.com

Aram Galstyan
Amazon AGI Foundations
argalsty@amazon.com

Abstract

Large language models can accumulate incorrect or outdated knowledge as the real world evolves. Compared to typical solutions such as retraining, retrieval augmented generation, model editing offers an effective yet low cost solution to address this issue. However, existing model editing algorithms employ manual selection of edit layers, which requires prior domain knowledge or expensive architecture-specific empirical layer selection methods, such as causal tracing. In this work, we propose SaLEM (Salient Layers Editing Model), an efficient solution for data driven layer selection for the model editing task. Our solution utilizes layer-wise saliency maps for layer selection, and matches the accuracy of prior approaches but with only 1/3 of their edits, enabling efficient updates to the parametric knowledge in large language models.

*Equal contribution; + Work done as Amazon intern

1 Introduction

Large Language models (LLMs) are well known for their capacity to store extensive factual knowledge, which enables them to perform well in tasks such as question answering (De Cao et al., 2021). However, facts can change after model training, which can introduce inaccuracies in model predictions and degrade downstream task performance. Updating such factual knowledge is typically done by fine-tuning the model with corrected answers but this is an expensive approach and is prone to model overfitting (Mitchell et al., 2022). Model editing (Sinitsin et al., 2020; Mitchell et al., 2022) techniques, such as MEND (Mitchell et al., 2022) and ROME (Meng et al., 2022) offer a practical and an effective alternative approach to address this problem, where we selectively edit a small subset of model parameters to update the factual knowledge.

An important prerequisite to do model editing is to identify the network layers most likely to store the corresponding facts to be edited. For instance, in MEND (Mitchell et al., 2022), this selection is done manually. While grounded in intuition, this approach depends on the model developer's domain knowledge and faces the risk of introducing superfluous edits. On the other hand, in ROME, (Meng et al., 2022), the authors employ causal tracing, which attempts to locate facts in an autoregressive neural network model by identifying hidden states which have the strongest causal effect on predictions of given facts. While effective, it is unclear if this technique generalizes beyond decoder model architectures. More importantly, layer selection through causal tracing is extremely costly, requiring full two autoregressive passes through the entire network for each layer *and* token in the input sequence.

In this work, we develop a new approach for automated layer selection for model editing called SaLEM (Salient Layers Editing Model). SaLEM leverages gradient values for given dataset with respect to the parameters of the LLM to be edited to create layer *saliency profiles* (Levin et al., 2022) and outputs the most salient layer to be edited. The salient layer selection method is an inexpensive, effective, and architecture-neutral approach for this task. We then thread the salient layer selection approach with MEND to apply edits using decomposed gradients with respect to the selected layers. Extensive experimental analysis established the effectiveness of SaLEM. Our main contributions in this work are as follows:

1. We propose SaLEM (Figure 1), a simple yet efficient and architecture-neutral approach for precise editing of erroneous knowledge in language models.

2. We conduct extensive empirical analysis on several benchmark datasets, demonstrating

the effectiveness of SaLEM in terms of editing accuracy but with substantially fewer number of training steps.

2 Related Work

Model Editors: The need to update and adapt knowledge representations of language models has traditionally been served through fine tuning (Kenton and Toutanova, 2019). Various model editing strategies have been explored, including modified fine-tuning methods that enforce locality of edits (Zhu et al., 2020) or minimize L2-norm parameter updates for reliable edits (Sotoudeh and Thakur, 2021), updating model beliefs based on learned optimizers (Hase et al., 2021). However, parameter space constraints may not always translate effectively into function space for neural networks (De Cao et al., 2021). To address this, fine-tuning can be incorporated with a KL-divergence (Kullback and Leibler, 1951) constraint, but this may not yield generalizable edits.

Editable Neural Networks (ENN) (Sinitsin et al., 2020) and Knowledge-Editor (KE) (De Cao et al., 2021) use meta-learning techniques (Finn et al., 2017; Ha et al., 2017) to effectively edit base models, offering alternative paths for desirable edit capabilities. But (Sinitsin et al., 2020) requires costly specialized training of the original network, while (De Cao et al., 2021) lacks tractability.

MEND (Mitchell et al., 2022) was proposed as a resource-efficient approach for training large language models by leveraging rank-1 gradients in a novel parameter update scheme. Unlike traditional gradient-based meta-learning algorithms (Finn et al., 2017; Lee and Choi, 2018; Park and Oliva, 2019; Flennerhag et al., 2020), MEND introduces adaptability post-hoc to a pre-trained model, enabling effective model adaptation without high computational costs. MEND, however, lacks a data-driven method for identifying most effective layers to edit, and instead applies edits to statically predetermined layers.

To address this gap, Meng et al. (2022) employed causal mediation analysis (Pearl, 2022; Vig et al., 2020) to trace hidden state activations within GPT (Radford et al., 2019). This helped identify and update parameters within the forward mid-layers MLPs that are decisive for last subject token in factual associations. Causal tracing, however, is an expensive parameter discovery mechanism requiring two full autoregressive passes through the model for each token, and has been recently shown not to always offer insights on the optimal MLP layer to edit (Hase et al., 2023) .

Interpreting LLMs: In search for a less expensive parameter discovery mechanism, we turned into the interpretability and attribution literature. Some previous work focused on measuring knowledge stored in pre-trained models using cloze queries (Petroni et al., 2019; Jiang et al., 2020), checking factual consistency (Elazar et al., 2021), examining knowledge neurons (Dai et al., 2022), or identifying causal input features . (Sundararajan et al., 2017). But most relevant to our purposes was the work of (Levin et al., 2022), where weights responsible for output are discovered by creating parameter saliency profiles, which are then used to obtain layer-saliency profiles utilizing gradient information of all parameters.

Our proposed model, SaLEM, builds on MEND with three crucial distinctions: (i) Empirical determination of the most salient layer, hence, eliminating the need for human expertise; (ii) Selectively targeting and editing the most salient layer only to minimize computational costs; (iii) Focusing on editing the outputs of mispredicted samples to enhance correctness and adaptivity.

3 SaLEM: Approach

3.1 Preliminaries

Consider a base model represented as $f_{\theta_W}(X) = Y$, where X denotes the input, θ_W represents trained parameters, and Y denotes the model output. Given a set of incorrectly predicted examples X_{fail}, the aim of model editing is to modify $f_{\theta_W}()$ to $f_{\theta_{\tilde{W}}}()$, thereby correcting the wrongly predicted outputs Y_{fail} to accurate answers. In other words, we want to map old learned parameters θ to new parameters $\theta_{\tilde{W}}$.

An important consideration while editing the model is to ensure that the correct edits also generalize to related inputs X_{adapt} which are semantically equivalent to X_{fail}, while keeping the model predictions unchanged for the correctly predicted examples X_{pass} (Sinitsin et al., 2020; De Cao et al., 2021; Mitchell et al., 2022; Meng et al., 2022). Therefore, a model editor is trained using an edit dataset D_{edit}, which includes the edit examples (X_{fail}, Y_{fail}), generalizability samples (X_{adapt}, Y_{fail}) and the locality samples (X_{pass}, Y_{pass}). The model editor, denoted as E, can be defined as:

$$E_\phi(D_{edit}, \theta_W) = \theta_{\tilde{W}} \quad (1)$$

To address the challenge of making efficient edits without computationally expensive and overfitting global parameter changes, we next introduce our approach which identifies the most salient network parameters responsible for the erroneous predictions, and performing edits solely on these selected parameters.

3.2 Saliency based layer selection

For a given base model $f_{\theta_W}()$, we begin by calculating layer wise saliency profiles with respect to the editing dataset D_{edit}. We utilize gradient information from the loss function as a measure of parameter saliency, aggregating at various levels:

1. **Parameter Saliency:** We compute parameter-wise saliency profiles, as introduced in (Levin et al., 2022), by calculating the gradients of the loss on the editing data D_{edit} with respect to the trained parameters θ_W for a given example (X, Y) from D_{edit}:

$$s_i(X, Y) = |\nabla_{\theta_W} L_{\theta_W}(X, Y)| \quad (2)$$

   A higher norm of the gradient signifies a greater impact of the respective parameter in making mistakes in D_{edit}.

2. **Column Saliency:** We compute column-wise saliency profiles by averaging parameter-wise saliency values across all elements of a column $\mathbf{p}$ in each layer's parameters of the network:

$$s_{\mathbf{p}}(X, Y) = \frac{1}{|\mathbf{p}|} \sum_{i=0}^{i=|\mathbf{p}|} s_i(X, Y) \quad (3)$$

   Here, $|\mathbf{p}|$ indicates number of parameters in layer $\mathbf{p}$ and $s_{\mathbf{p}}(X, Y)$ quantifies the saliency of given column $\mathbf{p}$ in a layer, with a higher value indicating a more significant impact on erroneous predictions.

3. **Layer Saliency:** Finally, to identify the saliency values for a layer $\mathbf{l}$, we further calculate averages of column-wise saliency profiles for each column $\mathbf{p}$ in the layer $\mathbf{l}$, and repeat this with each layer of the network:

$$s_{\mathbf{l}}(X, Y) = \frac{1}{|\mathbf{l}|} \sum_{\mathbf{p}=0}^{\mathbf{p}=|\mathbf{l}|} s_{\mathbf{p}}(X, Y) \quad (4)$$

4. **Select Edit Candidates:** Finally, we select top K layers with the highest saliency values as candidates for model editing:

$$EL = \underset{topK}{\arg\max}(s_{\mathbf{l}}(X, Y)) \quad (5)$$

3.3 Model Editing

Once we've identified the most salient layers, model editing is performed using the MEND framework (Mitchell et al., 2022). In this approach, we train a lightweight model editor network E to edit the weights of a specific layer $\mathbf{l}$. During testing, E transforms the fine-tuning gradient of the corresponding layer into a parameter update that aligns with three key properties: *correctness* (i.e., correcting erroneous outputs), *consistency* (i.e., maintaining correct outputs), and *adaptiveness* (i.e., adapting to semantically equivalent inputs).

The model editor E leverages the rank-1 fine-tuning gradient $\nabla_{W_l} L_{\theta_W}(X, Y)$ for the layer $\mathbf{l}$ as input and outputs the parameter edits for that layer, denoted as $\tilde{\nabla}_{W_\mathbf{l}}$. This is achieved by conditioning on single layer gradient values, reducing the computational complexity compared to editing all parameters. The overall loss to train E combines correctness loss $L_{corr} = -\log p_{\theta_{\tilde{W}}}(Y_e|X_e)$ and consistency loss $L_{cons} = KL\left(p_{\theta_{\tilde{W}}}(\cdot|X_e) \,\|\, p_{\theta_W}(\cdot|X_{pass})\right)$:

$$L_E = c_{fail} L_{corr}(\theta_{\tilde{W}}) + L_{cons}(\theta_W, \theta_{\tilde{W}}) \quad (6)$$

Here, $X_e = X_{fail} \cup X_{adapt}$. The loss defined in Equation 6 allows the model editor to adapt the parameters of the selected layer effectively while maintaining correctness, consistency, and adaptiveness.

4 Datasets

To evaluate our approach, we conducted experiments on a diverse set of datasets encompassing text classification with varying levels of accuracy, question-answering and generation tasks. We consider five text classification datasets: i) FEVER-FACTCHECKING (Thorne et al., 2018) - fact checking with respect to Wikipedia information, ii) MULTINLI (Williams et al., 2018) - sentence pairs annotated with textual entailment information, iii) DIALOGUENLI (Welleck et al., 2019) - sentence pairs consisting of a dialogue utterance and corresponding persona annotated with

EDITING MODELS →	FT		ENN		KE		MEND		SaLEM		
Datasets ↓	**EA ↑**	**DD ↓**	**EA ↑**	**DD ↓**	**EA ↑**	**DD ↓**	**EA ↑**	**DD ↓**	**EA ↑**	**DD ↓**	SL
MULTINLI	0.79	0.001	0.98	0.002	0.96	0.001	**0.99**	0.001	**0.99**	**0.0001**	10
DIALOGUENLI	0.90	0.001	**0.99**	0.0001	0.98	0.001	**0.99**	0.0001	**0.99**	**0.0001**	11
EMPATHETICDIALOGUES	0.53	0.026	**0.76**	0.017	0.69	0.214	**0.76**	0.016	**0.76**	**0.015**	10
PERSUASIONFORGOOD	0.66	0.16	**0.90**	0.009	0.87	0.011	**0.90**	0.008	**0.90**	**0.002**	10

Table 1: Results of **SaLEM** for **val** sets of natural language inference datasets *viz.* MULTINLI and DIALOGUENLI and classification datasets *viz.* EMPATHETICDIALOGUES and PERSUASIONFORGOOD. Each of the datasets base model is trained by fine-tuning BERT-large (Kenton and Toutanova, 2019).

Datasets →	ZSRe				WIKITEXT			
Generation Models →	T5-XL		BART		GPT-Neo 2.7B		Distil-GPT2	
EDITING MODELS ↓	**EA**	**DD ↓**	**EA ↑**	**DD ↓**	**EA ↑**	**DD ↓**	**EA ↑**	**DD ↓**
FT	0.57	**0.001**	0.96	**0.001**	0.55	0.200	0.28	0.991
ENN	-	-	**0.99**	**0.001**	-	-	**0.92**	**0.100**
KE	0.04	**0.001**	0.98	0.001	0.0	0.148	0.25	0.607
MEND	**0.88**	**0.001**	0.98	0.003	**0.81**	0.062	0.86	0.276
SaLEM	**0.88**	**0.001**	0.98	0.002	**0.81**	**0.054**	0.87	0.253

Table 2: Results of SaLEM on **val** sets of Question-Answering dataset ZSRe and generation dataset WIKITEXT. - denotes that ENN had not been run due to high computational requirements.

Model	**EA ↑**		**DD ↓**	
	Train	Val	Train	Val
FT	0.74	0.75	0.001	0.001
ENN	0.94	0.97	0.002	0.003
KE	0.90	0.94	0.003	0.004
MEND	**0.99**	**0.99**	0.001	0.001
SaLEM (3 layers)	**0.99**	**0.99**	**0.0001**	**0.0001**
SaLEM	**0.99**	**0.99**	**0.0001**	**0.0001**

Table 3: Results of SaLEM on FEVER-FACTCHECKING used by (Mitchell et al., 2022)

Generation Model →	**GPT2-XL**			
Editing Models ↓	**Efficacy ↑**		**Generalization ↑**	
	ES	EM	PS	PM
ROME	**1**	0.979	0.964	0.627
SaLEM	**1**	**0.986**	**0.967**	**0.649**

Table 4: Results of SaLEM on COUNTERFACT used by (Meng et al., 2022)

textual entailment information, iv) EMPATHETIC-DIALOGUES (Rashkin et al., 2019) - dialogue situations annotated with one of the 32 fine-grained emotions, and v) PERSUASIONFORGOOD (Wang et al., 2019) - a dialogues agent responses annotated with imbibed persuasion strategies. These datasets provide comprehensive evaluations on a diverse set of tasks. Statistics for each of these datasets are listed in Table 5 of the Appendix. For generation tasks such as Question-Answering and next token generation, we utilized ZSRE (Levy et al., 2017) and WIKITEXT (Merity, 2016) datasets. Details on how we used these datasets to train the editor networks are in Appendix A.

5 Experimental Results

We conduct detailed experiments, comparing SaLEM with four competitive baselines: i) FT (Fine-tuning), ii) ENN (Editable Neural Networks), iii) KE (Knowledge Editing) and iv) MEND, and report results on two key evaluation metrics: EA (Edit Accuracy) and DD (Drawdown).

5.1 Implementation Details

To optimize the performance of SaLEM, we used identity function as the initialization method (Mitchell et al., 2022), along with a residual connection (He et al., 2016) for enhanced learning. Additionally, a combination of partially random and partially zero initialization strategies is employed (Zhang et al., 2019). U_1 and U_2 are initialized with zeros, while V_1 and V_2 are initialized using the standard Xavier initialization (Glorot and Bengio, 2010). To address varying input magnitudes, u and δ_{l+1} are normalized to have zero mean and unit variance. This improves the conditioning, training speed, edit performance and efficiency of SaLEM.

For classification, we use BERT-large (Kenton and Toutanova, 2019) with 12 layers and 125M parameters, whereas for generation, we use Distil-GPT2 with 6 layers 82M parameters (Sanh et al., 2019) and GPT-Neo (Black et al.; Gao et al., 2020) with 24 layers and 2.7B parameters. Lastly, for Question-Answering task, we employ BART-large (Lewis et al., 2020) with 24 layers and 406M parameters and T5-XL (Raffel et al., 2020) with 24 layers and 3B parameters. Consistent with previous research work (Mitchell et al., 2022), all reported performance metrics are based on the validation set. The maximum number of training steps is set at 150000, but we terminate training if validation set does not decrease for 30000 steps to prevent overfitting. Following (Mitchell et al., 2022) we focus on editing MLP layers rather than editing the attention layers, as they yield better performance.

During training, we utilize a batch size of 10, employing gradient accumulation to effectively update model parameters. We employ the Adam optimizer (Kingma and Ba, 2015) to optimize parameters at each time step. Throughout our experiments, we maintain a consistent value of $c_{fail} = 0.1$ for all the conducted trials (Mitchell et al., 2022). This ensures that the optimization procedure focuses on editing the existing information while also allowing for sufficient non-edits to search for potentially better solutions.

5.2 Classification Results

We first present performance on the FEVER-FACTCHECKING dataset, as edit instances are sampled differently in this task. As evident in Table 3, the data driven layer selection approach of SaLEM in conjunction to MEND, meets the EA of vanilla MEND (which manually selects layers 10, 11, and 12 for editing) and achieves lower DD. Further, while vanilla MEND required 55,000 steps for this experiment, SaLEM completed it in only 45,000 steps, highlighting its computational efficiency advantage.

We next evaluated SaLEM on the other datasets mentioned above, and show results in Table 1, highlighting the selected layer under column SL in the table. In terms of EA, it outperforms FT and KE and meets ENN and MEND across all four datasets. ENN and MEND, while competitive, come with specific limitations: ENN requires maintaining a duplicate base model, leading to increased memory demands while MEND depends on manual layer selection process. SaLEM further excels in DD value, potentially due to its 1/3 edits compared to MEND. The reduced number of edits allows SaLEM to minimize updates to the base model as compared to MEND, hence resulting into better DD score. he varying layer selections for different datasets underscore SaLEM's adaptability to diverse dataset characteristics. Its advantage lies in its ability to gain insights into the network's inner workings, identifying relevant parameters contributing to incorrect predictions, thus achieving efficient and targeted editing by focusing on a single layer.

5.3 Generation Results

In Table 2, we present the results for the Question-Answering datasets ZSRE and WIKITEXT. Notably, SaLEM outperforms the baselines FT, and KE across both datasets. Further, SaLEM matches MEND's performance in terms of successful edits. It is also seen that ENN outperforms all other models for both ZSRE and DISTIL-GPT2. Due to high computational requirements ENN is not evaluated for T-5-XL and GPT-Neo. Similarly, SaLEM outperforms FT, KE and meets MEND's results with T-5-XL and GPT-Neo. Specifically, FT struggles to generalize to different rephrasings of the edit input, resulting in reduced edit success. The KL-constrained baseline shows reduced DD for T5-XL, and GPT-Neo, but it comes at the expense of edit success. KE proves to be ineffective at this scale, generally failing to provide successful edits.

5.4 Autoregressive Models

To showcase the effectiveness of SaLEM's parameter selection mechanism with large autoregressive model (like GPT2-XL), we compared it with ROME (Meng et al., 2022) which uses causal tracing to select salient layer. In Table 4, we can see that SaLEM performs better than ROME (Meng et al., 2022) on the COUNTERFACT dataset developed in (Meng et al., 2022). In addition, SaLEM is computationally efficient since it needs only a single pass compared to ROME's multiple passes. Our experiments show the promising performance of SaLEM in both encoder-decoder and decoder only autoregressive architectures, while it is unclear how well ROME performs in encoder-decoder models. We provide additional experiments and results in Appendix C.

6 Conclusion

Facts stored in LLMs routinely get outdated, and model editing offers an elegant solution to selectively update these facts without compromising the integrity of the model. However, existing algorithms suffer from shortcomings such as relying on domain knowledge or using computationally expensive mechanism for layer selection. To address these shortcomings, here we propose SaLEM, an effective and computationally efficient solution for layer selection which utilizes parameter saliency maps aggregated at various levels. Our experimental results demonstrate that by identifying the salient layer, SaLEM matches the *edit success* of MEND and ROME with considerably better computational efficiency. Further, detailed evaluation of SaLEM across various NLP tasks, including natural language inference, classification, question-answering, and generation, demonstrate its robust performance.

Limitations

For low base classifier accuracies, SaLEM can be further improved. As we focused to edit only failed examples, we restricted our dataset size while training the edit models of SaLEM. SaLEM can be improved by enriching the editing dataset with better failed samples and their semantic and counterfactual equivalents. We also need a better weight update mechanism to inform the editor about the extent of updates for borderline instances, such that consistency of edited model can be maintained. This drives towards our future work. Further, though SaLEM is computationally efficient, in its current form it expects the entire LLM to be in memory before edits and hence requires considerable GPU memory when working with large LLMs. It maybe possible to perform the edits without loading the full model into memory, we defer this exploration for future work.

Ethics Statement

Algorithms designed for model editing offer a potential solution to address the issue of undesirable model behaviors by allowing developers to modify and rectify these behaviors as they are identified. However, it is important to acknowledge that a model editor could also be misused, potentially amplifying the very behaviors we aim to eliminate. For examples, a large language model can be edited to generate toxic sentences for given input. This dual use presents a risk inherent in development of these large language models. For all experiments, we used only publicly available datasets and adhered to their policies. On acceptance, we will make our editing datasets publicly available.

References

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

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.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. 2020. Meta-learning with warped gradient descent. In *International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

David Ha, Andrew M Dai, and Quoc V Le. 2017. Hypernetworks. In *International Conference on Learning Representations*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2021. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*.

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

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. 2015. Likert scale: Explored and explained. *British journal of applied science & technology*, 7(4):396.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936. PMLR.

Roman Levin, Manli Shu, Eitan Borgnia, Furong Huang, Micah Goldblum, and Tom Goldstein. 2022. Where do models go wrong? parameter-space saliency maps for explainability. *Advances in Neural Information Processing Systems*, 35:15602–15615.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Stephen Merity. 2016. The wikitext long term dependency language modeling dataset. *Salesforce Metamind*, 9.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale.

OpenAI. 2023. Chatgpt (mar 14 version) [large language model].

Eunbyung Park and Junier B Oliva. 2019. Meta-curvature. *Advances in Neural Information Processing Systems*, 32.

Judea Pearl. 2022. Direct and indirect effects. In *Probabilistic and causal inference: The works of Judea Pearl*, pages 373–392.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC^{2}Workshop*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. *arXiv preprint arXiv:2004.00345*.

Matthew Sotoudeh and Aditya V Thakur. 2021. Provable repair of deep neural networks. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 588–603.

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

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: A large-scale dataset for fact extraction and verification. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2018*, pages 809–819. Association for Computational Linguistics (ACL).

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. 2019. Fixup initialization: Residual learning without normalization.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Datasets Creation

To train the editor networks, we require correct, consistent and adaptive instances. Hence, to create such datasets, samples to be corrected (i.e., X_{fail}) are obtained from test datasets where the base model $f_{\theta_W}()$ failed. Similarly, X_{pass} corresponds to accurately predicted instances in the test dataset. The adaptive samples X_{adapt} are obtained through rephrases of X_{fail}. We get five rephrased samples for each of the instance in X_{fail} in three phases as follows:

1. **Paraphrasing:** We initially tried to generate paraphrases using different openly accessible LLMs like GPT-Neo (Black et al.; Gao et al., 2020), GPT-J (Wang and Komatsuzaki, 2021; Wang, 2021), using which we obtained seven rephrases of 20-30 samples from each of the five datasets. The generated responses were found to be qualitatively bad for GPT-Neo, while GPT-J lacked in fluency and diversity of generated outputs. Hence, we employed Chat-GPT (OpenAI, 2023) to generate the final paraphrases for 50% of samples of each of the five datasets and then trained three different versions of the BART model (Lewis et al., 2020) to generate 3-2-2 paraphrases respectively. We leverage three BART models in order to counteract any information loss due to finite memory.

2. **Automatic Filtration:** The generated paraphrasers from BART are quantitatively evaluated in terms of BERTScore F1 (BS_{F1}) (Zhang et al., 2020) to check the quality of paraphrases, and those with $BS_{F1} < 0.4$ are discarded. After this, if the number of rephrases were found to be less than five for a given instance in X_{fail}, we repeated the previous step by generating rephrases using Chat-GPT (OpenAI, 2023).

3. **Manual Filtration:** We randomly sampled 50% of all rephrased samples from previous steps, and evaluated them in terms of fluency, adequacy and semantic-coherence on an integer likert scale (Likert, 1932; Joshi et al., 2015) of 1, 2, and 3 [1]. Evaluations were conducted by authors of the paper. Candidates with fluency=1, adequacy=1 and semantic-coherence=1 are sampled and rephrased again

[1] 1,2, and 3 denotes low, neutral and high quality rephrase.

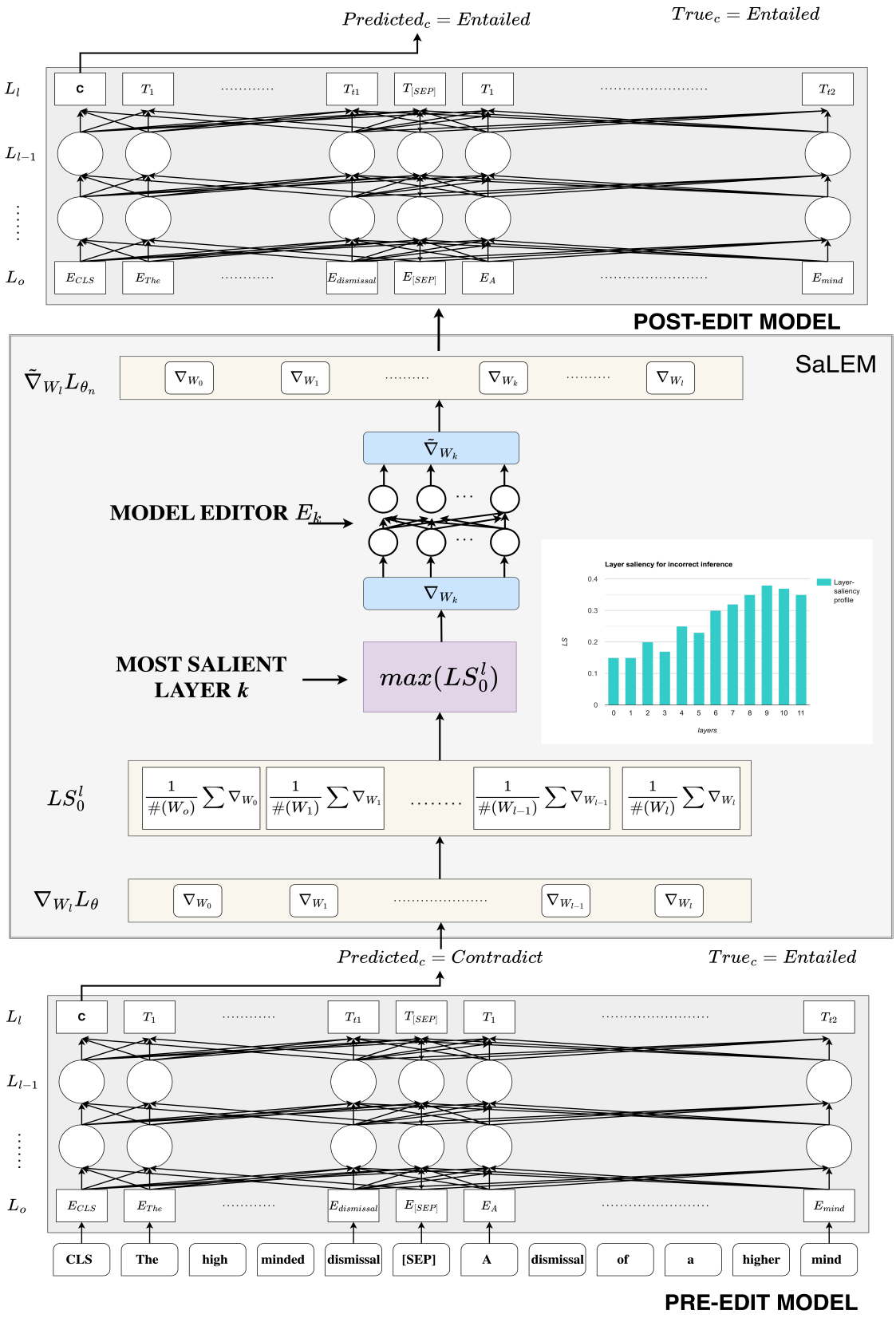

Figure 1: SaLEM Architecture: Identifying the most critical layer for erroneous entailment and training an editing network using low-rank gradient decomposition.

by the authors of the paper. Only 6% samples were found to be low quality rephrases. After editing these low quality rephrases, we end up with our X_{adapt} samples for each of the X_{fail} instances in all of four datasets *viz.*. MULTINLI, DIALOGUENLI, EMPATHETIC-DIALOGUES and PERSUASIONFORGOOD.

Finally, for generation tasks such as Question-Answering and next token generation, we utilized ZSRE (Levy et al., 2017) and WIKITEXT (Merity, 2016) datasets respectively. FEVER-FACTCHECKING, ZSRE and WIKITEXT are used same as (Mitchell et al., 2022). The edit instances for editing datasets *viz.* MULTINLI, DIALOGUENLI, EMPATHETICDIALOGUES and PERSUASIONFORGOOD datasets are incorrectly predicted instances in 3-fold cross-validation of respective classifiers. Whereas FEVER-FACTCHECKING differs from these datasets in the sense that edit instances binary labels are obtained by sampling from a Bernoulli distribution with a probability value of 0.5. The new flipped labels are treated as labels to be edited.

B Experiments

We conduct experiments to (i) to assess the effectiveness of SaLEM with respect to various competitive baselines: Fine-tuned (FT), Editable Neural Networks (ENN) (Sinitsin et al., 2020), Knowledge Editor (KE) (De Cao et al., 2021), and Model Editor Networks with Gradient Decomposition (MEND) (Mitchell et al., 2022), and (ii) perform extensive empirical analysis to showcase the importance of selecting layers empirically using SaLEM.

B.1 Baselines

1. **FT:** The fine-tuned base-model on edit dataset D_{edit}.

2. **ENN:** Discover a set of model parameters that achieves high performance on a given *'base task'* such as classification or machine translation, simultaneously, aim to enable efficient editing of the model's predictions for a specific set of *'edit examples'* through gradient descent, while ensuring that the model's behavior remains unchanged for unrelated inputs.

3. **KE:** An RNN that conditions explicitly on the input, incorrect output, and new desired label. outputs a mask m_i, offset b_i, and a scaling factor α to the gradient $\tilde{\nabla}_{W_i}$ for i^{th} weight matrix in a language model.

4. **MEND:** A collection of small auxiliary editing networks that use a single desired input-output pair to make fast, local edits to a pre-trained model's behavior. It learns to transform the gradient obtained by standard fine-tuning, using a low-rank decomposition of the gradient to make the parameterization of this transformation tractable.

B.2 Evaluation Metrics

We evaluate the correctness, consistency and adaptiveness of a model editor through the use of two key metrics: Edit Accuracy (**EA**), and Drawdown (**DD**) (Mitchell et al., 2022). Edit Accuracy (**EA**), serves as a measure of the effectiveness of our model editor. It quantifies the success rate of editing by evaluating the extent to which the edited model aligns with the desired modifications or enhancements. It can be formulated as:

$$EA = \mathbb{E}_{x_e, y_e} \mathbb{1}\{\text{argmax} p_\theta(y|x_e) = y_e\} \quad (7)$$

To assess the consistency aspect of the edits, we employ the Drawdown metric (**DD**). **DD** is computed by measuring the performance degradation of the edited model on the remaining dataset, when compared to the base model. The specific form of **DD** calculation depends on the problem being addressed. For tasks involving generative LLMs, **DD** is determined by the increase in perplexity of the edited model. On the other hand, for tasks involving classification, **DD** is computed as the decrease in accuracy. Considering both Edit Accuracy (**EA**), and Drawdown (**DD**), we gain insights into the correctness of the model editor's modifications as well as their impact on the adaptiveness capabilities of the edited model. These metrics provide a comprehensive evaluation framework for assessing the performance and effectiveness of our model editor. To evaluate all model editors, we adopt the train:val::90:10 split across all datasets. All editors are evaluated on **val** datasets and trained on **train** datasets.

C Additional Results

C.1 Layerwise Ablations

To highlight the importance of selecting the most salient layer, we conducted experiments with

Datasets	# of instances	Model Accuracy	# Edit instances	# Adaptive instances
MULTINLI	412349	0.823	76204	381020
DIALOGUENLI	343110	0.955	16951	84750
EMPATHETICDIALOGUES	19194	0.576	8080	40400
PERSUASIONFORGOOD	6018	0.706	1865	9327

Table 5: Dataset Statistics of MULTINLI, DIALOGUENLI, EMPATHETICDIALOGUES, and PERSUASIONFORGOOD.

Model	EA	DD	Steps
MEND (0,1,2)	0.89	0.005	55000
MEND (1,2,3)	0.95	0.009	1135000
MEND (2,3,4)	0.96	0.008	110000
MEND (3,4,5)	0.95	0.008	70000
MEND (4,5,6)	0.93	0.007	65000
MEND (5,6,7)	0.94	0.010	75000
MEND (6,7,8)	0.94	0.012	70000
MEND (7,8,9)	0.96	0.011	85000
MEND (2,5,9)	0.97	0.09	90000
MEND (1,2,4)	0.94	0.08	80000
MEND (8,9,10)	**0.99**	**0.0001**	**50000**
MEND (9,10,11)	**0.99**	0.001	55000

Table 6: Results of MEND on FEVER-FACTCHECKING with different set of layers. MEND (a, b, c) denotes MEND with a^{th}, b^{th}, and c^{th} layers.

MEND by editing different sets of layers in in Table 6 of Appendix. From the table, it is evident that when using the sets $\{8, 9, 10\}$ and $\{9, 10, 11\}$, MEND achieves the same performance w.r.t. **EA**, which is significantly better than the other variants such as MEND (0,1,2), MEND (1,2,3), MEND (2,3,4), MEND (4,5,6), MEND (5,6,7), MEND (6,7,8), and MEND (7,8,9). It is worth noting that MEND performs less effectively in the shallower layers of BERT-large compared to the deeper layers. This observation suggests that deeper layers play a more significant role in making decisions. Further, in terms of **EA** and **DD**, it is also seen that MEND (8,9,10), and MEND (9,10,11) outperforms MEND (2,5,9), and MEND (1,2,4) selecting three layers randomly. This supports our argument that we do need a mechanism to select the most salient layer/s need to be edited.

C.2 Error Analysis

It can be seen in Table 1 that **SaLEM** for base models with low accuracy, the editing accuracy is low compared to high accuracy base models. It could be due to the absence of reliable edit samples to train the editor which can clearly discriminate between different classes. For generation tasks (in Table 2) with Distil-GPT2, SaLEM achieves lower **DD** as compared to ENN, reflecting that SaLEM performs edits even for consistent examples. These instances could be borderline instances, which SaLEM may perceive as edit instances.