ATTRIBUTION-GUIDED PRUNING FOR CIRCUIT DISCOVERY AND TARGETED CORRECTION IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The parameters of Language Models (LMs) encode knowledge acquired during training and jointly support many behaviors at inference. Identifying the components associated with a particular behavior provides insight into its implementation, while modifying them offers targeted control but can affect general capabilities. Circuit discovery and model correction address these respective goals. We connect them through attribution-guided pruning, which uses attribution methods to score neurons and individual weights. For circuit discovery, we retain highly attributed components. For correction, we compare component attribution on behavior-eliciting prompts and diverse reference text, then prune components more strongly associated with the target behavior. We evaluate Layer-wise Relevance Propagation (LRP) and one-step gradient saliency, with Wanda as a pruning baseline where applicable. On Indirect Object Identification, LRP yields the sparsest structured circuits across the three tested checkpoints and remains competitive for individual-weight pruning. In held-out evaluations across nine LM checkpoints from 125M to 8B parameters, attribution-guided pruning reduces the mean toxicity score by 43.2-77.6% on prompts whose unpruned continuations exhibit the target behavior, defined in this study by high toxicity, insult, and obscenity scores. Across three models, it improves response uniqueness by 39.8-86.6% on repetition-eliciting prompts. All selected corrections retain at least 90% of their unpruned HellaSwag and ARC-Challenge normalized accuracy and keep WikiText2 perplexity within $1.20\times$ baseline. Both LRP and gradient saliency support targeted correction, with gradients yielding the best selected operating point for several models. The results show that the same parameter-level pruning framework can yield sparse task circuits and suppress selected generation behaviors.

1 INTRODUCTION

Since the introduction of the Transformer architecture Vaswani et al. (2017), Language Models have become increasingly capable across a wide range of tasks. Their internal mechanisms, however, remain difficult to understand and control, limiting transparency and raising concerns about reliability in sensitive applications Bender et al. (2021); Liu et al. (2023). Mechanistic interpretability addresses this problem by identifying circuits: subsets of model computations or components associated with a behavior Conmy et al. (2023); Ferrando & Voita (2024); Marks et al. (2024). Existing circuit-discovery methods commonly identify nodes or edges in an activation graph using interventions or approximations to activation patching Conmy et al. (2023); Syed et al. (2023); Ferrando & Voita (2024); Jafari et al. (2025). These methods can aggregate evidence over a dataset and subsequently ablate the selected components. Our work instead changes the native object that receives a score: we directly score (attribute) learned parameter groups and individual weights. Like activation-space circuits, these scores are data-dependent. The extracted circuit in this work is represented as a fixed parameter mask that can be used without recomputing attribution. This formulation directly connects circuit discovery to structured and unstructured pruning.

The framework accepts any attribution score defined over the candidate components. We evaluate it with LRP Bach et al. (2015) and one-step gradient saliency Simonyan et al. (2014), and compare Wanda Sun et al. (2023), an activation-aware pruning score, where applicable. LRP redistributes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

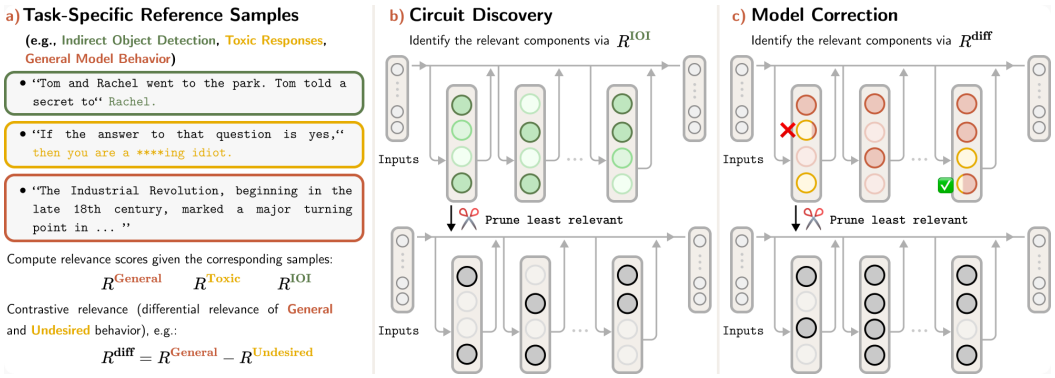

Figure 1: Attribution-guided circuit discovery and correction. **(a)** Component scores are averaged over behavior-specific inputs. **(b)** Pruning low scores extracts a sparse parameter circuit. **(c)** Contrastive scores identify parameters associated more strongly with an undesired behavior than with broad language-modeling behavior represented by a diverse reference dataset.

the model output across internal components using relevance-conservation rules Bach et al. (2015); Montavon et al. (2019); Achtibat et al. (2024), whereas gradient saliency measures the output's first-order sensitivity to each component. Using behavior-specific reference samples, we apply both methods to neurons and individual weights; LRP additionally propagates relevance through attention operations. The weight-level LRP decomposition is detailed in Appendix B.1.

Parameter attribution can also guide a direct behavioral intervention. We study behavior-specific correction: given a reference set that reliably elicits a predefined behavior, can that behavior be suppressed on held-out eliciting inputs by removing a small number of components while preserving broader model utility? Attribution on the target samples alone is insufficient because highly attributed components may also support general language modeling[1]. We therefore contrast attribution averaged over behavior-specific samples with attribution averaged over long and diverse reference text, and prune components associated more strongly with the target behavior. This produces a persistent parameter intervention without fine-tuning (Fig. 1). Perplexity on WikiText2 and zero-shot evaluations are later used to assess whether general model utility is preserved.

We propose attribution-guided pruning for two primary applications: 1) Circuit discovery, where attribution identifies neurons and individual weights that support a target behavior, and pruning extracts a sparse parameter subnetwork that can be evaluated directly; and 2) Model correction, where contrastive attribution identifies and prunes components associated more strongly with an undesired behavior than with general language modeling, while preserving measured model utility. We empirically study how the attribution method affects circuit sparsity, necessity, and sufficiency, as well as the trade-off between behavioral correction and utility preservation. Although not our primary focus, the framework can also be applied to model compression by computing attribution on broad language-modeling data and pruning the lowest-scoring components.

2 Related Work

Attribution methods Attribution methods explain neural network predictions by assigning importance scores to inputs or internal components. Common approaches include gradient-based saliency methods Simonyan et al. (2014); Smilkov et al. (2017), Integrated Gradients Sundararajan et al. (2017), and DeepLIFT Shrikumar et al. (2017) which estimate how changes in inputs affect model outputs. LRP Bach et al. (2015); Montavon et al. (2019) instead redistributes the prediction scores backward through the network via relevance conservation, producing importance scores for intermediate neurons and parameters. Prior work reports advantages of LRP-based pruning over one-step gradients in vision models Hatefi et al. (2024); Yeom et al. (2021), while transformer-specific propagation rules enable decomposition through attention operations Achtibat et al. (2024); Ali et al.

[1]We use "general language modeling" and "broad language modeling" interchangeably for the general-purpose behavior approximated by diverse reference text.

(2022). These results motivate our evaluation of LRP but do not imply universal superiority over gradient attribution in language models. In this work, we use and compare both LRP and one-step gradient saliency as component-scoring methods.

Pruning and compression Pruning reduces model size by removing parameters with limited contribution to performance. Recent methods for Large Language Models (LLMs) rely on heuristics such as weight magnitude or activation-based importance. For example, Wanda Sun et al. (2023) prunes weights based on their magnitudes scaled by activation norms. Attribution-based pruning has also been explored, where attribution scores are used to identify important components Yeom et al. (2021); Hatefi et al. (2024). In this work, we extend attribution-guided pruning to LLMs. The pruning framework operates on component scores; we instantiate it with LRP and gradient saliency and compare it with Wanda. While our focus is circuit discovery and model correction, we additionally show competitive compression performance (Appendix D).

Circuit discovery Circuit discovery aims to identify internal components, such as attention heads and neurons from Multilayer Perceptron (MLP), that drive specific model behaviors. Most existing approaches identify nodes or edges in an activation-level computation graph, using intervention effects aggregated over task-specific examples. Methods such as Sparse Auto Encoders (SAEs) Marks et al. (2024) learn interpretable activation features but require additional training. Transcoders further decompose MLP computations into sparse feature circuits Dunefsky et al. (2024). Activation patching approaches such as Automated Circuit DisCovery (ACDC) Conmy et al. (2023) instead estimate causal importance via systematic interventions. ACDC evaluates candidate edge removals through repeated activation patching, which requires a model evaluation for each intervention. More efficient variants, including Information Flow Routes (IFR) Ferrando & Voita (2024) and Edge Attribution Patching (EAP) Syed et al. (2023), approximate these interventions using gradient-based or attribution-based signals. Relevance Patching (RelP) Jafari et al. (2025) integrates LRP coefficients into the attribution-patching signal. These methods can aggregate evidence across inputs and can subsequently ablate their selected components. In contrast, our native scored object is a parameter group or individual scalar weight: we identify circuits as dataset-conditioned subsets of model parameters. We use attribution scores (i.e., LRP or gradients) to rank parameters by relevance and extract circuits via pruning, enabling efficient identification of sparse, behavior-specific subnetworks and directly connecting circuit discovery with model correction and compression. Table 1 summarizes how representative methods select behavior-relevant components and the form of their resulting circuits or pruned models.

Table 1: Comparison of behavior-relevant component selection. All methods aggregate their selection signal over a set of examples; they differ in the signal, selected unit, and resulting representation.

Method	Selection signal	Selected unit	Result
ACDC/EAP/RelP Conmy et al. (2023); Syed et al. (2023); Jafari et al. (2025)	Patching-based effects	Graph nodes or edges	Sparse computation graph
Wanda Sun et al. (2023)	$\lvert W\rvert \times \lVert X\rVert_2$	Individual weights	Sparse model
Ours	Component attribution	Neurons, heads, or weights	Behavior-specific pruned model

Model correction Model correction aims to suppress a specified behavior while retaining capabilities unrelated to the intervention. Existing approaches differ in where and how they modify the model. Alignment methods update model parameters through fine-tuning Ouyang et al. (2022). Representation-level methods instead remove or steer directions in hidden states Ravfogel et al. (2020); Turner et al. (2023); activation steering applies such modifications during inference. In vision models, attribution has also been incorporated into training objectives Ross et al. (2017), while other methods intervene on latent representations to reduce biased behavior Schramowski et al. (2020); Anders et al. (2022); Pahde et al. (2023); Dreyer et al. (2024). Parameter-editing methods target a different correction setting. Knowledge-neuron methods localize and modify neurons associated with factual predictions Dai et al. (2021), whereas ROME Meng et al. (2022a) and MEMIT Meng et al. (2022b) update selected weights to replace factual associations. We instead study dataset-defined generation behaviors and use contrastive attribution to rank existing neurons or individual weights. Attribution is compared between behavior-specific references and text representing general language

modeling, after which selected components are set to zero. The intervention therefore requires neither parameter optimization nor inference-time activation modification.

3 METHODS

We introduce a framework for attribution-guided pruning that connects component-level scoring with circuit discovery and model correction. The framework accepts any score defined over the candidate components. We evaluate it using LRP and gradient saliency, and compare Wanda where applicable. The pruning and contrastive-correction procedures do not otherwise depend on which score is used.

3.1 ATTRIBUTION-BASED PRUNING

Building on the framework introduced by Yeom et al. (2021); Hatefi et al. (2024), let $\Psi = \{\psi_1, \ldots, \psi_p\}$ denote a set of p components, defined as structural units like neurons from MLPs, attention heads, or other trainable parameters, that constitute a Deep Neural Network (DNN), and let $\mathcal{X}_{\text{ref}} = \{x_1, x_2, \ldots, x_{n_{\text{ref}}}\}$ represent a set of reference samples. For each component $\psi_k \in \Psi$ and reference sample $x_i \in \mathcal{X}_{\text{ref}}$, we define $R_{\psi_k}(x_i)$ as the relevance (or importance) score obtained from an attribution method (*e.g.,* LRP or gradient saliency). By aggregating these scores across all reference samples and applying the normalization described in Eq. (1), we obtain $\mathcal{R} = \{\bar{R}_{\psi_1}, \bar{R}_{\psi_2}, \ldots, \bar{R}_{\psi_p}\}$, the set of normalized attribution scores for all components.

$$\bar{R}_{\psi_k} = \frac{1}{n_{\text{ref}}} \sum_{i=1}^{n_{\text{ref}}} R_{\psi_k}(x_i)\,. \tag{1}$$

We consider structured, global unstructured, layer-wise unstructured, and row-wise unstructured pruning (Appendix C). In each case, we rank components by their attribution scores. Specifically, we assign indices c by sorting components in ascending order of relevance, up to the q-th place:

$$\{c\}_q = \text{argsort}(\mathcal{R})_{1,2,\ldots,q}\,. \tag{2}$$

Defining $\mathbf{1}$ to represent an indicator function with condition $i \in \{c\}_q$, the q least relevant components can be pruned by masking as:

$$\forall \psi_i \in \Psi : \psi_i \mapsto (1 - \mathbf{1}_{i \in \{c\}_q})\psi_i\,. \tag{3}$$

3.2 ATTRIBUTION SIGNALS

Let $f(x)$ denote the model's logits for a tokenized reference sequence x, and T its final input position. We attribute $s(x) = \max_v f(x)_{T,v}$, the logit of the predicted next token after x. We use the final position because its prediction conditions on the full input sequence and provides one model-internal target without an external behavior score. For generated examples, x includes the prompt and continuation. One-step gradient saliency Simonyan et al. (2014) assigns the absolute derivative $R^{\text{grad}}_{\psi_k}(x) = |\partial s(x)/\partial \psi_k|$ to component ψ_k. LRP Bach et al. (2015); Montavon et al. (2019) instead redistributes relevance from the same selected logit through the network using layer-wise rules. Both signals are averaged over the reference set using Eq. (1). The LRP rules and parameter-level decomposition are detailed in Appendix B.

3.3 CIRCUIT DISCOVERY

We define a circuit as a subnetwork comprising a subset of model components $\mathcal{C} \subseteq \Psi$, where Ψ denotes the candidate weights, neurons, or attention heads. We extract $\mathcal{C}$ by progressively pruning components with low attribution averaged over a behavior-specific reference set $\mathcal{X}_{\text{ref}}$. The final circuit is the sparsest retained subnetwork that preserves the target metric within a specified tolerance. Circuit discovery is therefore dataset-conditioned. Once extracted, $\mathcal{C}$ is represented by a fixed parameter mask and can be evaluated without access to the reference set.

Activation-space methods (*e.g.,* Conmy et al. (2023); Syed et al. (2023); Ferrando & Voita (2024); Jafari et al. (2025)) instead define circuits as nodes or edges in an activation graph. These methods can also aggregate over datasets and ablate selected components. Our narrower distinction is that the relevance scores and the extracted mask are defined directly over learned parameter groups and scalar weights, rather than being derived from an activation-graph score.

3.4 Model correction

Attribution on the undesired samples alone cannot show whether a component is specific to that behavior or is also needed for general text generation. We therefore compare attribution on the undesired samples with attribution on general text before pruning. Let $\mathcal{X}_{\text{ref}}^{\text{General}}$ and $\mathcal{X}_{\text{ref}}^{\text{Undesired}}$ denote the sets of reference samples that capture the model's general behavior (*e.g.,* datasets such as Wikipedia and C4) and a specific undesired behavior (*e.g.,* toxicity), respectively. Applying the framework described in Sec. 3.1 to each set yields $\bar{R}^{\text{General}}$ and $\bar{R}^{\text{Undesired}}$. We define the differential score as

$$\bar{R}_{\psi_k}^{\text{diff}} = \bar{R}_{\psi_k}^{\text{General}} - \bar{R}_{\psi_k}^{\text{Undesired}} \tag{4}$$

Intuitively, this contrasts components that support the model's general functionality with those that contribute to the undesired behavior. The resulting scores form the differential attribution set $\mathcal{R}^{\text{diff}} = \{\bar{R}_{\psi_1}^{\text{diff}}, \bar{R}_{\psi_2}^{\text{diff}}, \ldots, \bar{R}_{\psi_p}^{\text{diff}}\}$, which is subsequently used for pruning. Following Eq. (2), we prune the lowest differential scores first. A negative score specifically means that average attribution is greater on the undesired samples than on general text.

4 Experiments

Our experiments study two questions: whether attribution-guided pruning recovers sparse circuits that preserve a target behavior, and whether contrastive attribution can suppress an undesired behavior while retaining measured model utility. We evaluate circuit discovery on OPT-125M Zhang et al. (2022), TinyLlama-1.1B Zhang et al. (2024), and Qwen2-0.5B Yang et al. (2024). Model correction uses Qwen2-1.5B and additionally evaluates Llama-3-8B AI@Meta (2024), Gemma-3-4B Gemma Team (2025), and Qwen3-4B Yang et al. (2025), including Base and Instruct variants when available. We use compact checkpoints for IOI because each circuit is evaluated at many pruning levels and across seeds; correction is tested separately on checkpoints up to 8B. Unless stated otherwise, all results are aggregated over five or six data splits, as specified for each experiment.

4.1 Discovering task-specific and sparse circuits

We evaluate circuit discovery on Indirect Object Identification (IOI) Conmy et al. (2023), a standard task with a well-defined behavioral metric. We use least-relevant-first (LeRF) pruning to test whether the retained components are sufficient to preserve IOI accuracy at high sparsity, and most-relevant-first (MoRF) pruning to test the collective necessity of the highest-ranked components under the same intervention. At the shared neuron granularity, we compare our IOI rankings with RelP Jafari et al. (2025) to ask whether the two methods select the same top-ranked neurons.

Experimental settings. For each model, we generate six sets of 128 IOI sequences using different data-loader seeds and evaluate every extracted circuit on 512 sequences. We compare LRP, gradient saliency, and Wanda Sun et al. (2023); aggregated neuron activations are included only in the OPT neuron-level analysis. We evaluate structured pruning of neurons and unstructured pruning of individual weight elements; for OPT, we additionally evaluate attention-head pruning. The main text reports the OPT results, while the corresponding TinyLlama and Qwen2 experiments are reported in Appendix F.4.

The main difference from activation-space circuit discovery is the type of component that receives a score. ACDC and EAP-style methods score nodes or edges in an activation graph. They can also rank components such as neurons and ablate selected ones. RelP adds LRP coefficients to the attribution-patching signal. In contrast, we directly score learned parameter groups and individual weights. All of these methods use data to discover the circuit. After discovery, our method returns

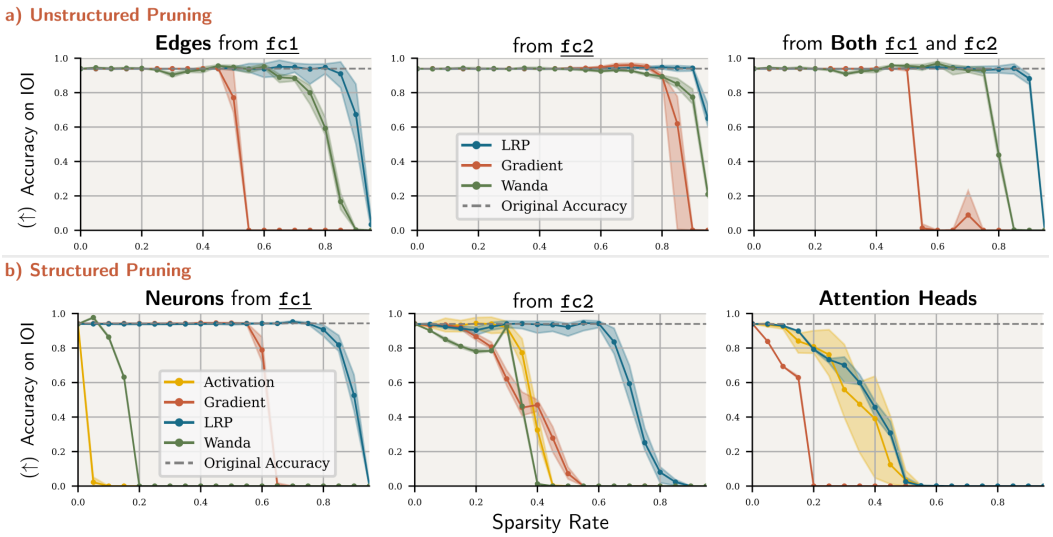

Figure 2: IOI circuits extracted from OPT-125M by *(a)* unstructured weight pruning and *(b)* structured neuron/head pruning. Curves show mean ± standard deviation across six IOI sets.

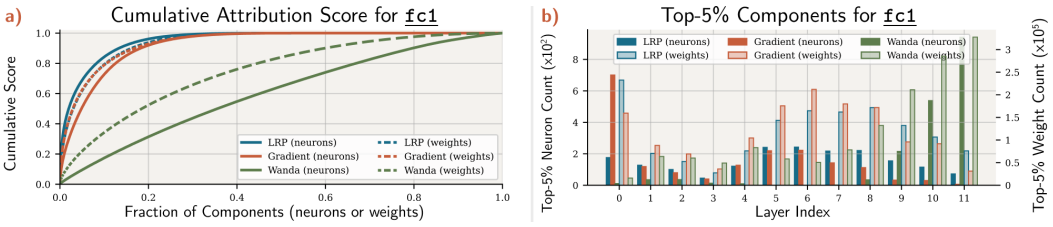

Figure 3: IOI localization in OPT-125M `fc1` layers. *(a)* Cumulative attribution concentration for neurons (solid) and weights (dashed). *(b)* Layer-wise counts among the global top-5% components; filled and outlined bars denote neurons and weights, respectively.

a fixed parameter mask that can be evaluated without the reference set or recomputing attribution scores. The methods can be compared at neuron granularity because they may select the same neurons. RelP's formulation does not define scores for individual learned weights, so we restrict the direct comparison to neurons.

We use performance-sparsity curves as a feature-perturbation test Samek et al. (2017). In the LeRF direction shown here, a high-quality score removes many components before IOI accuracy degrades. This measures the sufficiency of the retained subnetwork under the specified pruning intervention; it is not, by itself, a complete causal description of the model. We complement this test with MoRF pruning (Appendix F.5), which removes the highest-scoring components first and tests their necessity under the same masking intervention. At 5% MoRF pruning, the mean IOI accuracy of LRP and gradients is at most .009 and .015, respectively, in every tested Qwen2 and TinyLlama setting. Wanda retains substantial accuracy in three of six structured settings, but also collapses in the remaining structured and all unstructured settings.

Fig. 2 shows that LRP preserves IOI accuracy after removing more components than the alternative attribution signals. Wanda is competitive for unstructured pruning when its native row-wise rule is used, whereas LRP and gradients perform better with global ranking across layers (complete curves in Appendices F.2 and F.3). Across TinyLlama and Qwen2, LRP again produces the sparsest structured circuits and remains competitive with Wanda for weight pruning (Appendix F.4).

The localization analysis in Fig. 3 provides a complementary view of the score distributions. LRP is most concentrated at neuron level, while its weight-level curve nearly overlaps with gradient saliency; both are more concentrated than Wanda. Top-ranked LRP and gradient components span multiple layers, with gradient neurons peaking additionally in the first layer, whereas Wanda concentrates both

granularities in the final layers. The `fc2` analysis in Appendix F.1 shows that these patterns depend on the projection and component granularity.

We run RelP on the same six seeded IOI sets. At 5% retention, the methods share 18.40% to 45.13% of their selected neurons, compared with 5% expected from random sets; three of four layer types show more than $5.6\times$ layer-matched enrichment (Appendix F.6). Thus, they select some of the same IOI neurons, but this agreement does not establish causal validity for either ranking. Because RelP's formulation does not define scores for individual learned weights, scalar-weight overlap is not defined.

Discussion. Together, the LeRF and MoRF experiments provide intervention-specific evidence that the retained circuits are sufficient for IOI and that the top-ranked sets contain the necessary components for the behavior. Across the three models, LRP extracts the sparsest structured IOI circuits and remains competitive with Wanda for weight pruning. The score patterns depend on projection and granularity: LRP is most concentrated in `fc1`, gradient saliency has a similar weight-level profile, and Wanda emphasizes later layers. LRP's top-ranked neurons also overlap with those found by RelP beyond random and layer-matched expectations, showing that the two formulations localize related neurons despite assigning scores to different objects. These results do not provide a complete causal description of information flow; activation-patching methods study that complementary question, while our formulation directly produces neuron- and weight-level parameter masks.

4.2 Targeted model correction

We next test whether the same parameter attribution can be used to suppress a target behavior. We study toxic continuation and repetitive generation, which represent two different generation behaviors. Our objective is *behavior-specific correction*, not unconditional detoxification. For toxicity, we evaluate held-out prompts for which the unpruned model already generates the target toxic behavior. We therefore measure whether pruning suppresses this specific behavior. We do not estimate toxicity over arbitrary model use. The complete model-correction protocol is reported in Appendix G.

Models, data, and held-out evaluation. For toxicity, each checkpoint greedily generates 30-token continuations for all 99,442 RealToxicityPrompts records Gehman et al. (2020). After prompt deduplication, we retain model-specific examples whose generated continuation satisfies toxicity $\geq .9$, insult $\geq .9$, and obscene $\geq .9$ under `unitary/unbiased-toxic-roberta` Hanu & Unitary Team (2020). Across checkpoints, this filter retains 0.062-0.879% of unique prefixes under our decoding and classifier settings (Tab. 6). We partition these examples into five folds. For every fold, four partitions compute target attribution and the remaining partition is held out for behavioral evaluation. For repetition, we use 1,000 author-curated prompts drafted with assistance from ChatGPT and organized into ten distinct prompt families. We retain prompts whose unpruned unique-word ratio (UWR) is at most .5 and hold out two complete prompt families per fold. Thus, no repetition prompt family appears in both attribution and evaluation data within a fold. Exact sample counts and filtering details are in Appendix G.1.

C4's diverse web text serves as a proxy for broad language modeling; it is not selected to elicit either target behavior. We compute its scores once per checkpoint and method on 128 sequences of 512 tokens. For each fold, we average the target scores over its attribution split and contrast the two averages (Eq. (4)). At every pruning point, we regenerate continuations on attribution and held-out prompts, rescore the target behavior, and evaluate WikiText2 perplexity, HellaSwag, and ARC-Challenge. We compare LRP, gradients, and Wanda on OPT, TinyLlama, and Qwen2; the 4B and 8B models use LRP and gradients. Full configurations are in Appendix G.2.

Operating-point protocol. Stronger pruning may suppress more of the target behavior, but it can also reduce the model's general performance. We therefore report an *operating point*, which is one method, layer type, pruning granularity, and pruning count selected under a predefined utility constraint. We aggregate identical settings over all five folds and use two tiers: *strict* requires WikiText2 perplexity to remain within $1.10\times$ its baseline and both zero-shot accuracies to retain at least 95%; *moderate* uses $1.20\times$ and 90%. Within each method, we select the eligible point with the best mean attribution-split behavior score, breaking ties by fewer removed components. We then choose the best method by the same criterion. Each fold's evaluation prompts are excluded from that

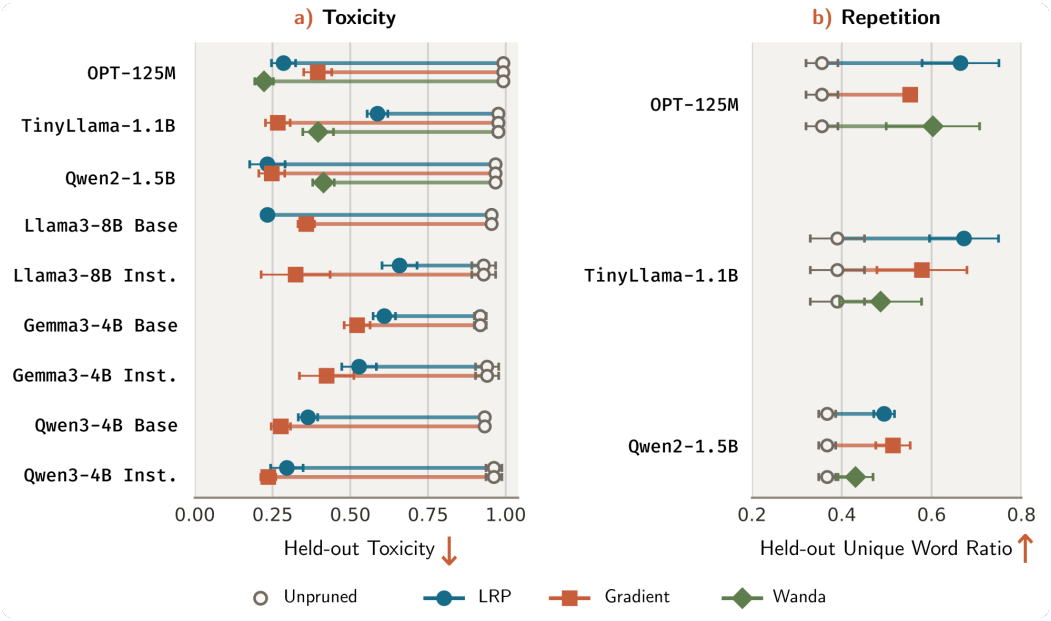

Figure 4: Best-found moderate operating point for each method and model. Each point uses the method's independently selected layer type, pruning granularity, and pruning count, and therefore summarizes the correction achieved by the complete procedure rather than comparing attribution methods under a matched pruning setting. Lower toxicity and higher UWR are better; error bars show standard deviation over five folds.

fold's attribution, and their held-out scores do not enter selection. Because one setting is selected across folds, these scores are not an independent final test. The final HellaSwag and ARC-Challenge evaluations are likewise run only after selection and cannot influence the selected configuration.

Fig. 4 reports the best valid moderate point found for each method, while Tab. 2 summarizes the best point per model and behavior. Because the methods can select different layers, granularities, and pruning counts, this figure describes how much correction the complete procedure can achieve rather than providing a matched comparison of attribution signals. The matched-setting comparison in Fig. 22 fixes these choices and directly compares LRP with gradients. The corresponding best-found strict operating points are reported in Fig. 15.

At the moderate points in Tab. 2, held-out toxicity decreases by 43.2-77.6% across all nine checkpoints, while UWR increases by 39.8-86.6% across the three repetition models. All 12 points satisfy the moderate constraints on the complete HellaSwag (10,042 examples) and ARC-Challenge (1,172 examples) evaluations. LRP is selected for two toxicity and two repetition models, gradients for six toxicity and one repetition model, and Wanda for OPT toxicity. These results show that contrastive attribution can suppress both target behaviors, but no attribution method is best for every model. Although `up_proj` recurs among moderate toxicity optima, selected layers, granularities, and counts vary across models and utility tiers, so the sweeps do not support a universal pruning setting (Tab. 12).

Discussion. The matched-setting comparison and operating-point selection answer different questions. For the same model, layer, granularity, and pruning count, LRP gives a larger toxicity reduction in 112 of 114 strict comparisons and 178 of 185 moderate comparisons (Appendix G.6). Gradients can still give the best eligible operating point by using a different layer or count. The complete sweeps show the behavior-utility trade-offs around the selected points (Appendix G.5).

5 Conclusion

Attribution-guided pruning connects circuit discovery and model correction through interventions on model parameters. In Indirect Object Identification (IOI), LRP yields the sparsest structured

Table 2: Final moderate operating points. Target values are held-out toxicity (T; lower is better) or UWR (R; higher is better). H/A are percentages of the unpruned full HellaSwag/ARC-Challenge `acc_norm`. S/U denote structured/unstructured pruning. The *Removed* column gives the count and percentage of eligible candidates across all selected layer-type modules: output neurons for S and scalar weights for U.

Task	Model	Method; mask	Removed (#; %)	Held-out target	PPL×	H	A
T	OPT-125M	Wanda; U-`fc2`	1M; 3.53	$.992 \rightarrow .223\ (-77.6\%)$	.969	100.3	100.6
T	TinyLlama-1.1B	Grad.; U-`up`	2M; .788	$.976 \rightarrow .267\ (-72.7\%)$	1.157	97.1	94.6
T	Qwen2-1.5B	LRP; U-`up`	200K; .0519	$.967 \rightarrow .233\ (-75.9\%)$	1.174	99.2	101.1
T	Llama-3-8B Base	LRP; S-`up`	2K; .436	$.955 \rightarrow .234\ (-75.5\%)$	1.174	98.0	93.3
T	Llama-3-8B Inst.	Grad.; S-`gate`	5K; 1.09	$.929 \rightarrow .324\ (-65.1\%)$	1.089	99.1	96.6
T	Gemma-3-4B Base	Grad.; U-`up`	50K; .00561	$.918 \rightarrow .522\ (-43.2\%)$	1.048	101.1	100.1
T	Gemma-3-4B Inst.	Grad.; S-`up`	300; .0862	$.940 \rightarrow .424\ (-54.9\%)$	1.184	98.3	91.9
T	Qwen3-4B Base	Grad.; S-`up`	5K; 1.43	$.932 \rightarrow .277\ (-70.3\%)$	1.141	96.9	103.8
T	Qwen3-4B Inst.	Grad.; U-`up`	200K; .0223	$.962 \rightarrow .236\ (-75.4\%)$	1.177	93.7	92.7
R	OPT-125M	LRP; S-`fc1`	50; .136	$.356 \rightarrow .664\ (+86.6\%)$	1.097	98.5	95.0
R	TinyLlama-1.1B	LRP; U-`up`	100K; .0394	$.390 \rightarrow .672\ (+72.3\%)$	1.197	99.2	96.2
R	Qwen2-1.5B	Grad.; U-`gate`	50K; .0130	$.368 \rightarrow .514\ (+39.8\%)$	1.063	97.2	93.9

circuits across the tested checkpoints, while retaining and removing ranked components probe their sufficiency and collective necessity. For model correction, pruning based on contrastive attribution reduces elicited toxicity and repetition on held-out prompts while preserving measured utility. Both LRP and gradient saliency work in this setting. The sweeps also show that the useful layer, pruning granularity, and count depend on the model and utility constraint. Attribution provides the component ranking, and the sweep selects among tested pruning settings under the chosen utility constraints.

Limitations. Our circuit-discovery experiments focus on IOI and assess only the sufficiency of retained circuits and the collective necessity of top-ranked components under the tested masks. They do not provide a complete causal description of the model. Model correction also depends on reference sets constructed to show a specific behavior. In particular, our toxicity results do not measure or reduce toxicity over arbitrary prompts. Some toxicity folds for the Instruct models are small, and the classifier and thresholds determine the exact behavior that we study. The best layer type, granularity, and pruning count also depend on the model and behavior. However, the predefined utility constraints prevent us from selecting a setting using held-out correction scores. Finally, setting parameters to zero does not reduce runtime without suitable sparse kernels, and our full downstream evaluation covers only two benchmarks. Future work should evaluate broader behaviors, larger held-out sets, and additional downstream tasks.

Broader impact. Parameter-level attribution can make model interventions easier to inspect and can suppress a specific undesired behavior without retraining. The same approach could also be misused to increase harmful behavior, remove useful knowledge, or suppress outputs that are appropriate in their context. Applications of this framework should therefore report how the reference set was constructed, which classifier was used, and whether the intervention can be reversed.

REFERENCES

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 135–168. PMLR, 21–27 Jul 2024.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and Lior Wolf. Xai for transformers: Better explanations through conservative propagation. In *International conference on machine learning*, pp. 435–451. PMLR, 2022.

Christopher J Anders, Leander Weber, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Finding and removing clever hans: Using explanation methods to debug and improve deep models. *Information Fusion*, 77:261–295, 2022.

Leila Arras, Bruno Puri, Patrick Kahardipraja, Sebastian Lapuschkin, and Wojciech Samek. A close look at decomposition-based xai-methods for transformer language models. *arXiv preprint arXiv:2502.15886*, 2025.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Daniel Becking, Maximilian Dreyer, Wojciech Samek, Karsten Müller, and Sebastian Lapuschkin. ECQ^{x}: Explainability-Driven Quantization for Low-Bit and Sparse DNNs. In *xxAI - Beyond Explainable AI, Lecture Notes in Computer Science (LNAI Vol. 13200), Springer International Publishing*, pp. 271–296, 2022. doi: 10.1007/978-3-031-04083-2_14.

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

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Maximilian Dreyer, Frederik Pahde, Christopher J Anders, Wojciech Samek, and Sebastian Lapuschkin. From hope to safety: Unlearning biases of deep models via gradient penalization in latent space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 21046–21054, 2024.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *arXiv preprint arXiv:2406.11944*, 2024.

Javier Ferrando and Elena Voita. Information flow routes: Automatically interpreting language models at scale. *arXiv preprint arXiv:2403.00824*, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Laura Hanu and Unitary Team. Detoxify: Toxic comment classification with pytorch lightning and transformers. https://github.com/unitaryai/detoxify, 2020.

Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Reduan Achtibat, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. Pruning by explaining revisited: Optimizing attribution methods to prune cnns and transformers. In *European Conference on Computer Vision*, pp. 152–169. Springer, 2024.

Farnoush Rezaei Jafari, Oliver Eberle, Ashkan Khakzar, and Neel Nanda. Relp: Faithful and efficient circuit discovery in language models via relevance patching. *arXiv preprint arXiv:2508.21258*, 2025.

Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*, 2023.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. *Layer-Wise Relevance Propagation: An Overview*, pp. 193–209. Springer International Publishing, Cham, 2019.

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

Frederik Pahde, Maximilian Dreyer, Wojciech Samek, and Sebastian Lapuschkin. Reveal to revise: An explainable ai life cycle for iterative bias correction of deep models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 596–606. Springer, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

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

Farnoush Rezaei Jafari, Grégoire Montavon, Klaus-Robert Müller, and Oliver Eberle. Mambalrp: Explaining selective state space sequence models. *Advances in neural information processing systems*, 37:118540–118570, 2024.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2017.

Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486, 2020.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pp. 3145–3153. PMLR, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations Workshop*, 2014.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

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

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

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

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

An Yang, Baosong Yang, Binyuan Hui, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A TRANSFORMER MODELS

A.1 MLP PROJECTIONS IN THE EVALUATED MODELS

Among the checkpoints studied here, OPT-125M is the only one with an ungated MLP. Its `fc1` layer projects to the intermediate dimension, and `fc2` projects back to the hidden dimension, with an activation between them. TinyLlama, Llama-3, Qwen2, Qwen3, and Gemma-3 instead use gated MLPs with `gate_proj`, `up_proj`, and `down_proj`. The activated gate output is multiplied by the up-projection output before the down projection. We evaluate these architecture-specific layer types separately; OPT's `fc1` does not correspond to either gated input projection alone. Attention projections are separate from these MLP layers.

B METHODS

This section defines the LRP, gradient-saliency, and Wanda scores used in our framework.

B.1 LRP

Layer-wise Relevance Propagation Bach et al. (2015); Montavon et al. (2019) treats a neural network with L layers as a Directed Acyclic Graph (DAG). For an input x, the network computes

$$f(x) = f^L \circ \cdots \circ f^l \circ f^{l-1} \circ \cdots \circ f^1(x)\,. \tag{5}$$

We initialize the output relevance with the selected logit $s(x)$ from Sec. 3.2 and redistribute it toward the model inputs. At layer l, let a_i be the activation of input neuron i, w_{ij} the weight connecting it to output neuron j, and $z_{ij} = a_i w_{ij}$ its contribution to the pre-activation z_j. The basic LRP rule redistributes the relevance R_j^l in proportion to these contributions:

$$R_{i\leftarrow j}^{(l-1,l)} = \frac{z_{ij}}{z_j} R_j^l\,. \tag{6}$$

Summing the incoming relevance over the upper-layer neurons gives the relevance of neuron i. For the basic rule, this redistribution conserves total relevance between adjacent layers:

$$\sum_i R_i^{l-1} = \sum_{i,j} R_{i\leftarrow j}^{(l-1,l)} = \sum_j R_j^l\,. \tag{7}$$

B.1.1 FROM NEURON LEVEL TO PARAMETER-LEVEL ATTRIBUTION

The redistribution in Eq. (6) produces R_j^l, the relevance of neuron j in layer l. For neuron-level pruning, the component scores in $\mathcal{R} = \{\bar{R}_{\psi_1}, \ldots, \bar{R}_{\psi_p}\}$ are obtained directly from these neuron relevances and averaged over the reference set using Eq. (1). For unstructured pruning, relevance must instead be assigned to individual weight elements. Following Becking et al. (2022) and as shown in Fig. 5, we extend the decomposition to produce one relevance score for each weight element.

LRP is typically implemented as a modified gradient method, where the gradient is scaled by an input term. As described in Becking et al. (2022) and detailed in Eq. (8), LRP offers the flexibility to define this input term as either the activation a_i or the weight parameter w_{ij}. The remaining component then serves as the modified gradient. For our pruning approach, we adopt the latter formulation, treating the weight w_{ij} as the input. This allows us to directly compute a relevance score for each weight, $R_{w_{ij}} = R_{i\leftarrow j}$, effectively measuring the importance of individual parameters at any layer l.

$$R_{i\leftarrow j} = \underbrace{\underbrace{a_i w_{ij}}_{z_{ij}} \frac{R_j}{z_j}}_{\text{explicit}} = a_i \underbrace{\underbrace{w_{ij}}_{\frac{\partial z_j}{\partial a_i}} \frac{R_j}{z_j}}_{\text{mod. grad.}} = w_{ij} \underbrace{\underbrace{a_i}_{\frac{\partial z_j}{\partial w_{ij}}} \frac{R_j}{z_j}}_{\text{mod. grad.}}\,. \tag{8}$$

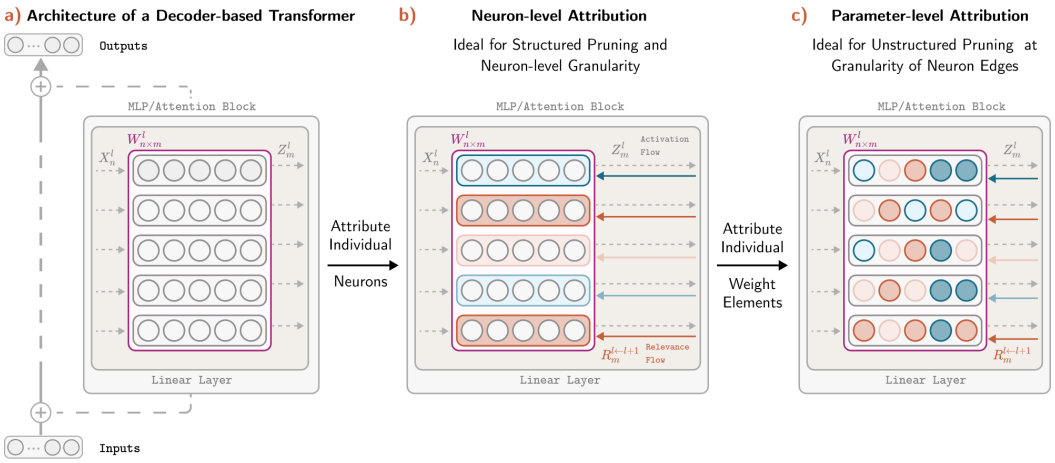

Figure 5: a) Linear layers with weight matrices W form the MLPs and attention projections of a decoder-only Transformer. b) The redistribution in Eq. (6) assigns relevance to individual neurons in linear layers, enabling structured pruning. c) For unstructured pruning, Becking et al. (2022) extends this decomposition to individual weight elements.

B.1.2 LRP RULES

Several LRP variants exist, including LRP-ϵ, LRP-$\alpha\beta$, LRP-z^+, and LRP-γ Bach et al. (2015); Montavon et al. (2019). They differ in how relevance is redistributed and stabilized. For the model-correction experiments, we use the LRP-ϵ rule with $\epsilon = 0$ in linear layers. Our implicit implementation computes this rule through modified gradients without explicitly dividing by z_j in Eq. (6), following the implementation principle discussed by Rezaei Jafari et al. (2024); Arras et al. (2025). We do not use the nonzero-ϵ stabilized variant in those experiments.

For Transformer attention, we use AttnLRP Achtibat et al. (2024) to propagate relevance through attention operations and obtain head-level scores.

B.2 GRADIENT SALIENCY

We extend one-step gradient saliency Simonyan et al. (2014) to internal components. For the same scalar output $s(x)$ used to initialize LRP, the score of component ψ_k is

$$R^{\text{grad}}_{\psi_k}(x) = \left| \frac{\partial s(x)}{\partial \psi_k} \right|. \tag{9}$$

Here, ψ_k is a neuron output or an individual learned weight. We average the scores over the reference set using Eq. (1). This is the absolute one-step derivative: it does not multiply the derivative by the component value, integrate gradients along a path, or use clean-corrupted activation patching.

B.3 WANDA

Unlike LRP, which requires both forward and backward passes to compute relevance scores, Wanda Sun et al. (2023) achieves efficient attribution using only a forward pass. It combines weight magnitudes and activations to derive attribution scores for a given weight matrix W at layer l with input activations X, computing R^l_W as:

$$R^l_W = |W| \cdot ||X||_2 \tag{10}$$

R^l_W has the same dimensions as W. Each individual element of R^l_W corresponds to a relevance score for the associated weight parameter w_{ij} in W.

Due to Wanda's design, relevance scores cannot be assigned at the granularity of individual attention heads, limiting its ability to capture fine-grained contributions compared to LRP. This limitation

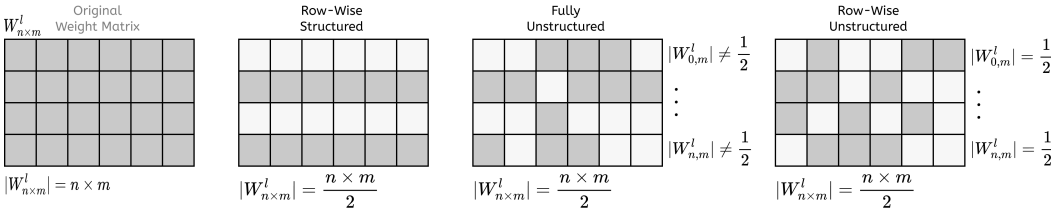

Figure 6: Applying an individual pruning rate to a linear layer with the weight matrix W, yet not limited to, can be achieved through either structured or unstructured approaches, potentially at various granularities targeting the entire matrix or its individual rows. For the purpose of compression, this paper adopts row-wise Unstructured pruning. In this figure, the zero-norm indicates the number of non-zero elements in W depicted with pink color

stems from Wanda's implementation, which is based on assessing weight values and activations directly, rather than isolating the contributions of specific components. As a result, Wanda faces challenges in attributing components that involve multiple weights, restricting its effectiveness in tasks such as discovering circuits among the attention heads.

C Pruning approaches

Several approaches can be used to apply pruning. The primary decision lies in choosing the granularity level, indicating whether to prune entire neurons or individual weight elements, and later the scale of comparison, which determines how the pruning rate is applied. For compressing LLMs, we follow Sun et al. (2023) and apply a uniform pruning rate to rows of weight matrices across all linear layers using a row-wise unstructured approach. This method is illustrated in Fig. 6, which also compares alternative pruning strategies.

In contrast to compression, we have followed these approaches for circuit discovery:

- Globally Structured: We compute an importance score for each neuron (i.e., each row in the weight matrix) of the linear layers, then rank neurons across the entire model. This allows for comparisons between neurons across different layers, such as comparing neuron i in layer l with neuron j in layer $l+1$.
- Row-Wise Unstructured: Following Sun et al. (2023) and our experiments in Appendix D, we apply uniform sparsity rates to each row of the weight matrix in the linear layer. Weight elements are compared within each row, as illustrated in Fig. 6.
- Globally Unstructured: This approach compares individual weight elements across layers, allowing direct comparisons between w_{ij} in one layer and w_{kl} in another. To reduce the computational cost of global sorting (which has $n\log(n)$ complexity), we use partitioning. However, this approximation may cause slight deviations from the desired pruning rate.

D Compression with unstructured pruning

Another application of our framework, is model compression which aims to reduce model size without hurting model performance on a general task. For model compression, *unstructured pruning* is the most widely used approach due to its finer granularity and strong potential to achieve high sparsity with minimal impact on performance. Compared to pruning individual components (*e.g.,* neurons or attention heads), it allows selective removal of individual weights. As detailed in Appendix C, unstructured pruning can be applied in various ways (*i.e.,* row-wise, layer-wise, or global).

Experimental settings We follow the evaluation protocol of Sun et al. (2023), applying *row-wise unstructured* pruning with uniform sparsity across the rows of weight matrices within linear layers. Thereby, attribution scores are ranked by magnitude per row, rather than across layers or the full model, as prior work Sun et al. (2023) found global or per-layer ranking to yield inferior performance.

To benchmark our method, we compare against the established post-training Wanda baseline Sun et al. (2023), which, like LRP, uses reference samples to assign importance scores to parameters without relying on external metrics or thresholds (see Appendix B.3). All experiments are conducted without fine-tuning. We evaluate three models from the Llama family: TinyLlama Zhang et al. (2024), Llama-2-7B Touvron et al. (2023), and Llama-3-8B AI@Meta (2024). Performance is assessed using two standard metrics: (1) perplexity on WikiText2 Merity et al. (2016), reflecting uncertainty in language modeling, and (2) zero-shot accuracy on a broad suite of tasks from Gao et al. (2024), capturing task-specific capabilities. Following Sun et al. (2023), we perform attribution using reference samples from the C4 dataset Raffel et al. (2020) to capture general model behavior. Specifically, we generate three sets of 128 samples (sequence length 2048), each from a different random seed to ensure robustness.

In Tab. 3, we apply a 50% sparsity rate. Higher sparsity rates (*e.g.,* 60%) typically degrade model performance strongly (as shown at Fig. 7 in the Appendix). Weight magnitude, a computationally cheap compression method, is not effective in pruning, as larger weights do not necessarily indicate greater contributions to decision-making. For example, a neuron with large weights may remain inactive. Both LRP and Wanda perform well in unstructured pruning and model compression, with Wanda showing a slight advantage. Our analysis in Appendix E details the key methodological differences between the two: Wanda efficiently attributes importance with fewer reference samples, while LRP excels at identifying sparser, task-relevant subgraphs. Notably, LRP becomes more effective when a larger corpus is available for attribution, which enables surpassing Wanda in performance (also see Fig. 7 in the Appendix).

Table 3: Perplexity (PPL) on WikiText2 and mean zero-shot accuracy (ACC) of TinyLlama, Llama2-7B, and Llama3-8B under 50% sparsity via row-wise unstructured pruning. Errors represent the standard error of the mean. Full performance details for each task are in the Appendix at Tab. 4.

Method	Models					
	TinyLlama		Llama2-7B		Llama3-8B	
	(↓) PPL.	(↑) ACC.	(↓) PPL.	(↑) ACC.	(↓) PPL.	(↑) ACC.
Original Scores	7.98	48.74	5.48	59.67	6.13	63.91
Magnitude	24.42	41.38	16.13	51.26	206.82	38.77
Wanda (Sun et al. (2023))	**11.52**±0.01	**45.46**±0.09	**6.94**±0.01	**55.92**±0.05	9.86±0.02	**57.07**±0.11
LRP	11.85±0.01	44.71±0.07	7.14±0.03	54.72±0.13	**9.82**±0.05	55.18±0.20

D.1 Zero-shot accuracies

Complete details on the zero-shot accuracy tasks are available in Tab. 4, with a summarized version in Tab. 3. While perplexity measures model uncertainty, evaluating zero-shot accuracy across various tasks provides insight into the reasoning capabilities of the compressed model.

Table 4: Zero-shot accuracy of TinyLlama, Llama2-7B, and Llama3-8B models compressed at a 50% pruning rate across tasks from Gao et al. (2024). The pruning rate is applied uniformly to rows of weight matrices in the linear layers using the row-wise unstructured approach described in Appendix C.

		Tasks							
		BQ	RTE	HS	WG	ARC-e	ARC-c	OBQA	Mean
TinyLlama	Original ACC.	*61.07*	*57.03*	*46.55*	*60.22*	*61.53*	*29.60*	*25.20*	*48.74*
	Magnitude	54.40	55.59	36.53	54.77	47.18	22.61	18.60	41.38
	Wanda(Sun et al. (2023))	**63.63**	**59.65**	39.32	**56.82**	51.57	25.10	**22.10**	**45.46**
	LRP	62.77	58.12	38.77	55.95	52.11	24.34	20.93	44.71
Llama2-7B	Original ACC.	*77.73*	*62.45*	*57.17*	*69.29*	*76.55*	*43.08*	*31.40*	*59.67*
	Magnitude	63.02	57.40	49.05	63.53	64.14	34.72	27.00	51.26
	Wanda(Sun et al. (2023))	**75.85**	53.42	**52.64**	**67.67**	**71.96**	**39.07**	**30.80**	**55.92**
	LRP	75.33	**55.11**	50.38	66.24	71.00	36.66	28.33	54.72
Llama3-8B	Original ACC.	*81.22*	*67.87*	*60.07*	*73.55*	*80.09*	*50.17*	*34.40*	*63.91*
	Magnitude	42.66	53.06	29.87	52.32	46.46	25.00	22.00	38.77
	Wanda(Sun et al. (2023))	**78.13**	**59.08**	**51.14**	**70.56**	**71.04**	**40.32**	**29.20**	**57.07**
	LRP	73.87	56.55	49.84	68.66	71.45	38.62	27.26	55.18

E LRP VS WANDA: CORE DIFFERENCES

In this section, we investigate the core differences between LRP and Wanda using the TinyLlama model. Attribution scores were computed with a single random seed, following the experimental setup of the unstructured pruning experiments in Appendix D. We conclude by summarizing the key observations from these comparisons.

Both Wanda and LRP rely on a set of reference samples, denoted as $\mathcal{X}_{\text{ref}}$, for attribution. Comparison of pruning performance with varying sizes of $\mathcal{X}_{\text{ref}}$ sheds light on the behavior of each method. As shown in Fig. 7, Wanda Sun et al. (2023) requires fewer reference samples to balance sparsity and performance (measured by perplexity on WikiText2). In contrast, LRP shows performance instability with small $\mathcal{X}_{\text{ref}}$ sizes but improves progressively as the sample size increases. With a large set of 8192 samples, LRP achieves a perplexity of 11.35, outperforming other methods, as reported in Tab. 3.

To gain deeper insights, we compared the distribution of attribution scores from LRP and Wanda under two scenarios: **1**) attribution using 128 reference samples, each with a sequence length of 2048 tokens, and **2**) attribution scores based on 3 individual samples, each of 2048 tokens. As shown in Fig. 8, Wanda's attribution score histograms remain largely consistent, even when only a single sample is used. This stability is consistent with our earlier observations in Fig. 7, highlighting Wanda's ability to maintain reliable attribution across different sample sizes.

Next, we investigate the sparsity of high-magnitude attribution scores in the TinyLlama model. Using 128 reference samples consistent with the settings in Appendix D, we collected attribution scores across all linear layers. These scores were min-max normalized to a range of $s \in [0, 1]$, and we counted those exceeding a given threshold. As shown in Fig. 9, LRP tends to concentrate importance on a smaller subset of weights, while Wanda distributes importance more broadly, generally favoring weights with large magnitudes and activations.

In our experiments (Appendix D, Sec. 4.1, and Fig. 7), Wanda reaches stable compression performance with relatively few reference samples. Its normalized weight scores are also less concentrated than the LRP scores in Fig. 9. LRP improves as the reference set grows and produces sparser IOI circuits in our circuit-discovery experiments. These observations describe the evaluated settings: Wanda is competitive for compression with a small reference set, whereas LRP provides the strongest IOI circuit-discovery results. They do not establish that either attribution score is best for model correction in general.

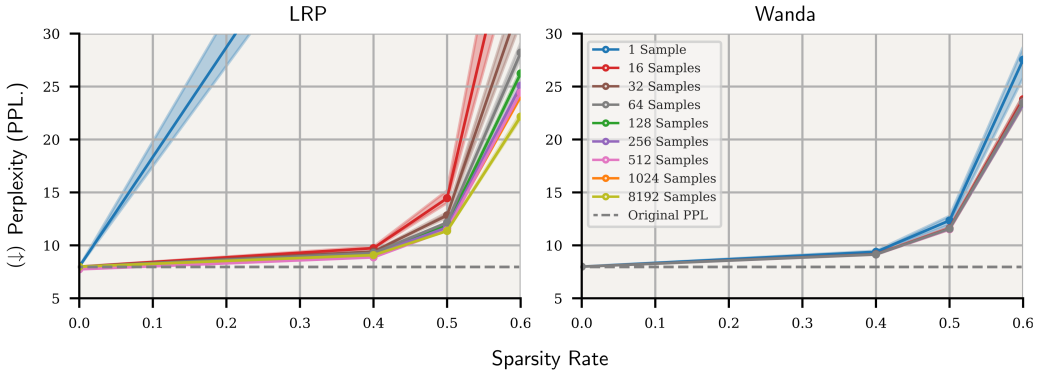

Figure 7: We performed row-wise unstructured pruning on TinyLlama using Wanda and LRP, testing with varying sizes of the reference sample set $\mathcal{X}_{\text{ref}}$. Three different $\mathcal{X}_{\text{ref}}$ sets were generated with variable random seeds. Due to GPU memory limitations in Wanda's official implementation, we were unable to use reference sets larger than 1024 samples. The results also show that applying sparsity rates above 50% significantly degrade model performance. The shaded regions in the figure represent the standard deviations across the different seeds, providing an indication of the variability in the results.

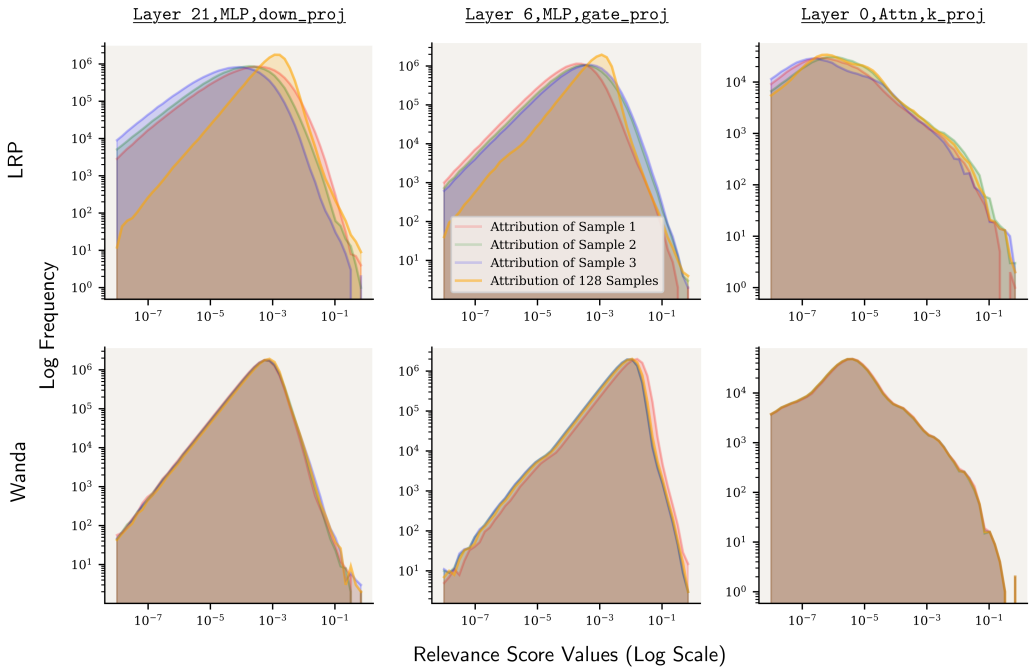

Figure 8: Attribution scores from Wanda and LRP were compared across three representative layers of the TinyLlama model: the layer with the highest average importance (`Layer 21, MLP, down_proj`), the median layer (`Layer 6, MLP, gate_proj`), and the layer with the lowest average importance (`Layer 0, Attn, k_proj`). The layers were ranked according to the importance scores computed by LRP. The histograms reveal that Wanda produces consistently stable attribution score distributions across these layers, while LRP exhibits higher variability. Similar trends were observed in other layers, though due to space constraints, we omit those visualizations.

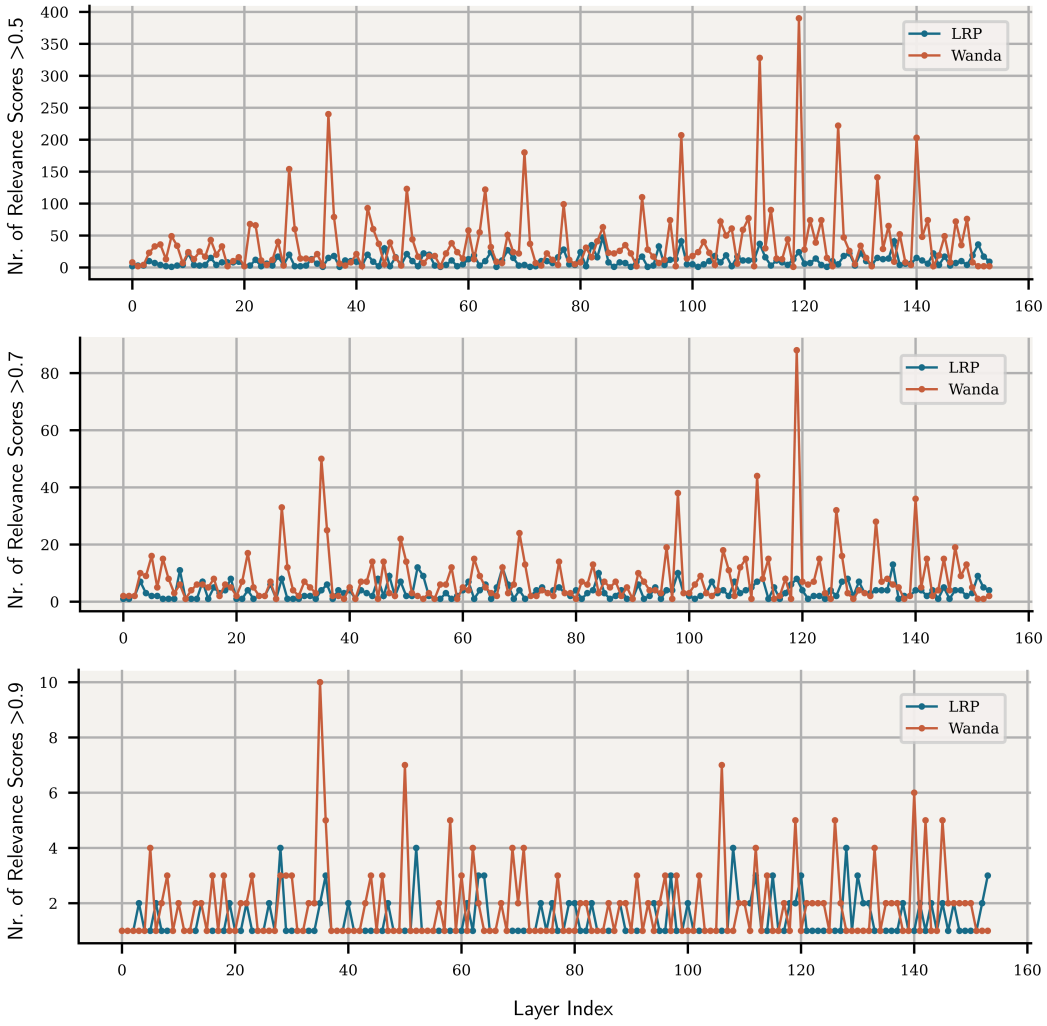

Figure 9: Number of attribution (relevance) scores exceeding a fixed threshold after min-max normalization of Wanda and LRP scores across all 154 linear layers of the TinyLlama model. This analysis reveals that LRP assigns importance to a sparser subset of components, whereas Wanda distributes importance more broadly.

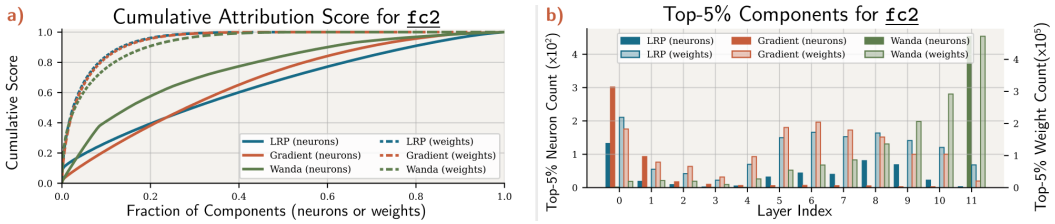

Figure 10: Localization analysis for IOI in the `fc2` layers of OPT-125M. *a)* Cumulative attribution concentration for neurons (solid) and weights (dashed). *b)* Layer-wise counts among the global top-5% components; filled and outlined bars denote neurons and weights, respectively.

F Circuit Discovery

In this section, we present the extracted IOI circuits of the OPT model across a broad set of sparsity rates, at other granularity levels based on the description in Appendix C. The experimental settings are similar to the configurations from Sec. 4.1.

F.1 Localization Analysis for FC2

We additionally analyze attribution concentration and layer-wise localization in the `fc2` layers, shown in Fig. 10. The result differs from `fc1`. At weight level, LRP and gradient saliency have nearly identical cumulative curves, and all three methods concentrate most attribution within a small fraction of weights. At neuron level, Wanda is more concentrated, while LRP and gradient saliency distribute attribution over more components. The layer-wise counts nevertheless follow a pattern similar to `fc1`: Wanda selects predominantly from the final layers, whereas LRP and gradient saliency select across a wider range of layers, with gradient neurons concentrated particularly strongly in the first layer. Thus, localization is specific to the projection and component granularity; `fc1` is the setting in which LRP shows its clearest concentration advantage.

F.2 Circuit Discovery via Structured Pruning

Circuits in this scenario are derived from neurons identified via globally structured pruning (Appendix C). As shown in Fig. 11, LRP-extracted circuits within the MLP layers are notably sparser and deliver superior performance compared to alternative methods. In this context, activation information was added as an additional baseline to align with ACDC Conmy et al. (2023). Interestingly, all methods perform poorly when targeting attention heads, underlining the crucial role of all heads working together in the IOI task within the OPT model.

F.3 Circuit Discovery via Unstructured Pruning

In this scenario, circuits are discovered from the weight elements attributed through globally unstructured and row-wise unstructured pruning, as outlined in Appendix C. Wanda achieves the best performance when IOI circuits are extracted using a uniform sparsity rate applied to rows of weight elements within each linear layer, as shown in Fig. 12 . This suggests that Wanda is particularly effective when pruning is applied uniformly at the row level. On the other hand, LRP and gradient-based methods yield superior results when weight elements are compared across different layers, as illustrated in Fig. 12, indicating that these methods benefit from global pruning strategies. The optimal configurations for Wanda, LRP, and gradient-based circuit discovery are summarized in panel *a* of Fig. 2.

F.4 Circuit Discovery on TinyLlama and Qwen2

We repeat the IOI experiment on TinyLlama-1.1B and Qwen2-0.5B. Attribution uses 128 IOI examples per seed and circuit evaluation uses 512 examples, matching the OPT protocol. In their gated MLP blocks, we separately target `down_proj`, `gate_proj`, and `up_proj`. Fig. 13 shows that LRP retains IOI accuracy at higher structured sparsity than gradient saliency and Wanda. For

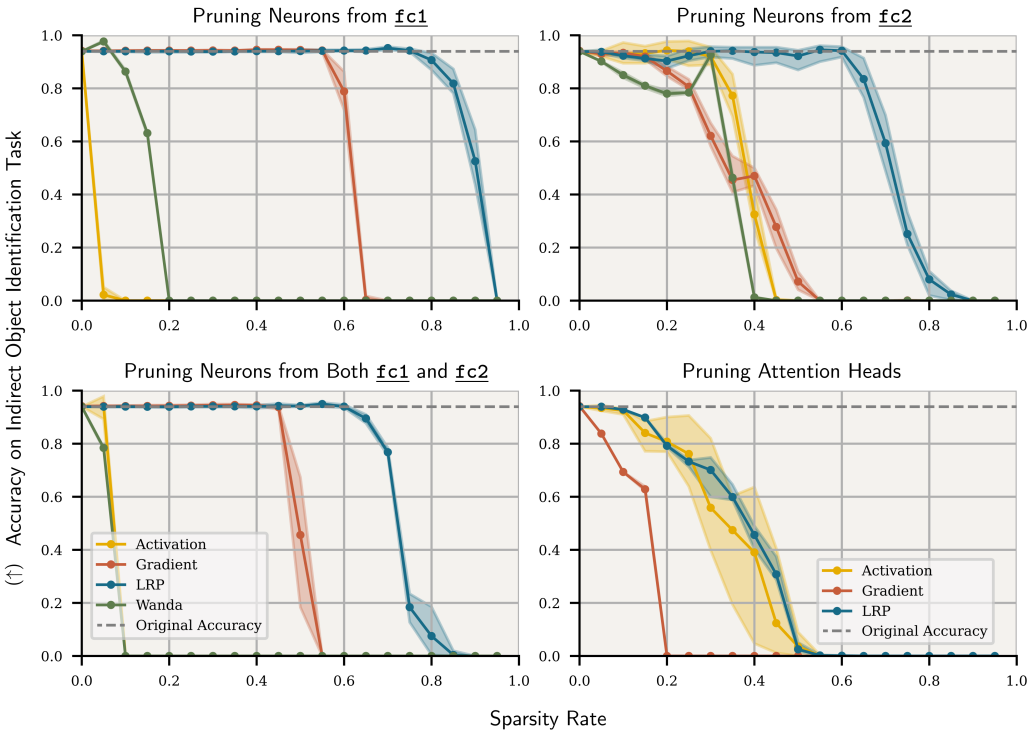

Figure 11: Based on the IOI circuits extracted from the OPT model using structured pruning, LRP identifies sparser and more effective circuits across neurons in MLPs and attention heads compared to Wanda and gradient. Notably, due to the absence of explicit weight parameters for individual attention heads in standard Transformer architectures, Wanda cannot be applied for circuit discovery within these heads (see Appendix B.3). The shaded regions in the results represent the mean of standard deviations, reflecting the variability in circuit discovery outcomes.

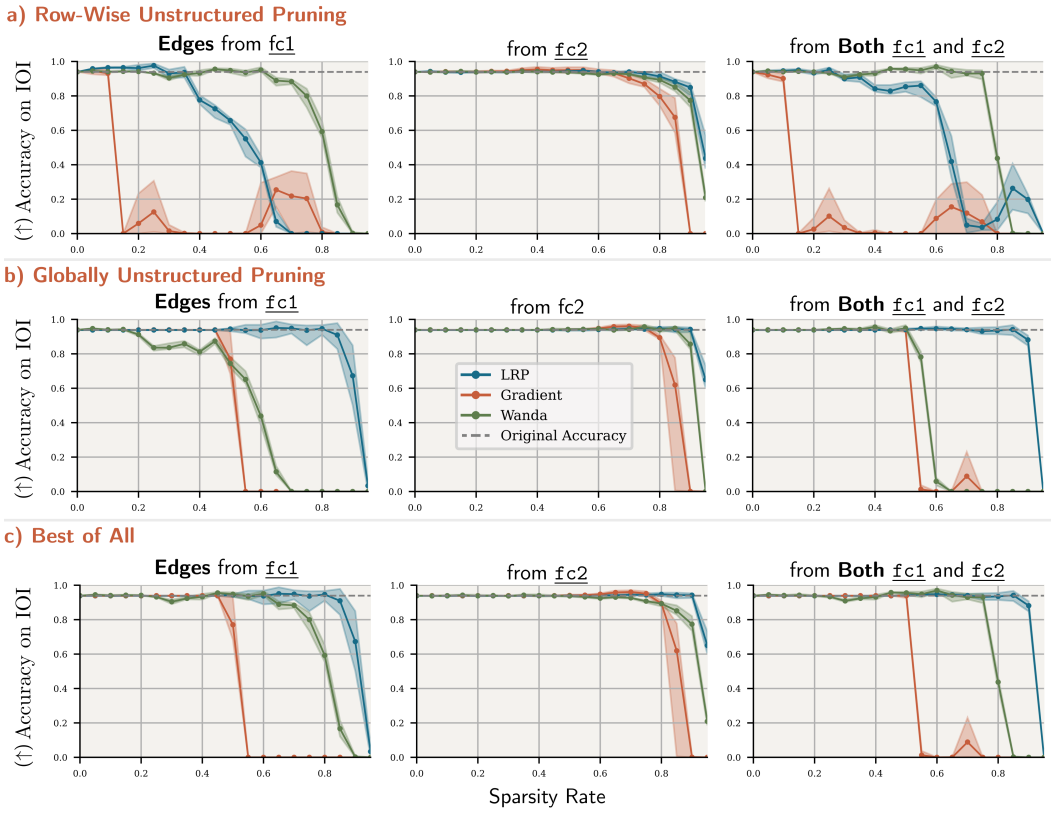

Figure 12: An overview of IOI circuits discovered from the OPT model using different layer types and unstructured pruning approaches (described in Appendix C) is shown in the figure. Panels a and b correspond to the row-wise and globally unstructured pruning approaches, respectively. Panel c represents the best configuration for each attribution method, where the row-wise approach is optimal for Wanda and the global technique is preferred for gradient and LRP. The shaded area in the figure indicates the mean of standard deviations.

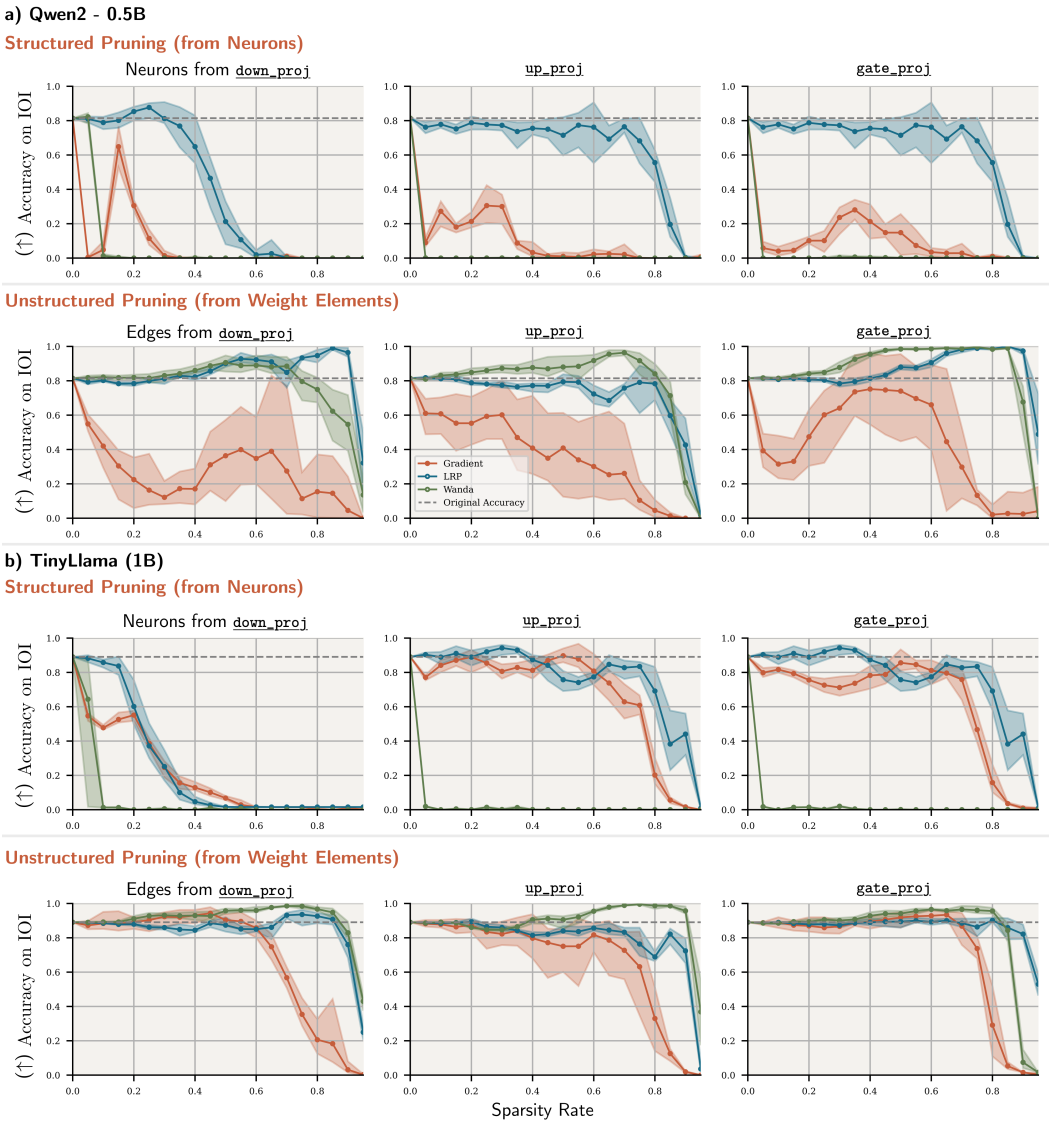

Figure 13: IOI circuit discovery on Qwen2-0.5B and TinyLlama-1.1B under structured (top) and unstructured (bottom) pruning. Shading shows variation across seeds.

unstructured pruning, LRP and row-wise Wanda are competitive, while gradients degrade earlier and show higher variance. These results extend the circuit-discovery pattern across three architectures; they do not imply that the same neuron coordinates are shared across models.

F.5 Most-relevant-first circuit ablation

The LeRF experiments retain the most highly attributed components and test whether the resulting subnetwork is sufficient for IOI. We additionally reverse the intervention and prune the most relevant components first in 5% steps. This MoRF experiment tests whether the components ranked highest by each method are necessary for the behavior under direct parameter masking.

At the first 5% intervention, LRP reduces mean IOI accuracy to the range $[0, .009]$ and gradients to $[0, .015]$ across all structured and unstructured settings in Fig. 14. Wanda is more dependent on the setting. It retains .900 and .666 accuracy for the Qwen2 `gate_proj` and `up_proj` neurons, and .951 for the TinyLlama `up_proj` neurons. It collapses in the other three structured settings and in every unstructured setting. Thus, MoRF provides complementary necessity evidence for the



Figure 14: IOI most-relevant-first ablations on Qwen2-0.5B and TinyLlama-1.1B. The upper and lower blocks show structured neuron and unstructured weight pruning, respectively. Curves report mean ± sample standard deviation over six attribution seeds, except TinyLlama structured `gate_proj` LRP ($n = 5$ because one run ended before the first intervention). Later points from interrupted runs are shown only while every contributing seed is available. Wanda uses its native row-wise rule for unstructured pruning.

top-ranked LRP and gradient components. It distinguishes them from Wanda in some structured MLP projections, but not uniformly across pruning settings. Like LeRF, this conclusion is specific to the applied masking intervention and does not by itself establish a complete causal account of IOI.

F.6 Neuron-level agreement with RelP

RelP Jafari et al. (2025) combines LRP coefficients with the activation difference used by attribution patching. It scores neurons inside an activation-patching framework. In contrast, our method scores the outputs of parameterized linear layers and can further decompose these scores over individual weights. We therefore compare the methods only at the shared neuron granularity. RelP's formulation does not define scores for individual learned weights, so a weight-level comparison is not defined.

For OPT-125M and Qwen2-0.5B, we recompute both methods on each of the same six IOI sets containing 128 examples. Rankings are global within one model and location. For Qwen's intermediate MLP location, we average the matching `gate_proj` and `up_proj` neuron scores to avoid counting one intermediate coordinate twice. At retention fraction r, both methods select exactly $k = \mathrm{round}(rN)$ neurons. For a matched seed, let A and B denote our and RelP's top-k sets. We define overlap as $|A \cap B|/k$, the fraction of one selected set shared by the other. Two uniformly random top-k sets have expected overlap r.

The layer-controlled enrichment separates agreement on exact neuron coordinates from agreement about which layers are important. Let a_l and b_l be the numbers selected by the two methods from a layer containing N_l neurons. If both methods preserve these layer counts but select neuron indices randomly within each layer, the expected intersection is $\sum_l a_l b_l / N_l$. We divide the observed intersection by this expectation. A value of $1\times$ therefore indicates no within-layer agreement beyond the shared layer allocation, whereas a value above $1\times$ indicates agreement on neuron indices within those layers. Finally, seed stability is computed separately for each method as $|A_s \cap A_t|/k$, averaged over all 15 pairs of the six data-loader seeds. It measures whether each method recovers similar top-k sets when the IOI attribution samples change.

Table 5: Our-RelP top-neuron agreement at 5% retention. Overlap and layer-controlled enrichment are averaged over six matched seeds; overlap is shown as mean ± sample standard deviation. Seed stability is the mean within-method overlap over the 15 seed pairs, reported as ours/RelP.

Model/location	k	Overlap	Layer-ctrl. ×	Seed stability (ours/RelP)
Qwen2 `fc1`	5,837	$32.65 \pm 0.35\%$	5.78	$92.98/90.52\%$
Qwen2 `fc2`	1,075	$32.65 \pm 0.65\%$	5.65	$93.44/90.95\%$
OPT `fc1`	1,843	$45.13 \pm 0.59\%$	7.34	$94.89/90.63\%$
OPT `fc2`	461	$18.40 \pm 1.51\%$	1.58	$86.73/80.56\%$

For both Qwen locations and OPT `fc1`, the methods select many of the same neuron indices even after controlling for their layer distributions. OPT `fc2` has above-random overlap, but its layer-controlled enrichment is only $1.58\times$. Thus, much of its raw agreement is explained by selecting neurons from the same layers rather than the same indices within those layers. The high seed-stability values show that both methods recover similar top-5% sets across IOI data-loader seeds, so the cross-method agreement is not caused by one favorable seed. The methods use the same IOI samples and both use LRP-related signals. We therefore interpret this result as agreement in neuron localization, not as an independent causal validation.

G TARGETED MODEL-CORRECTION PROTOCOL

This section describes the complete data-construction, pruning sweep, selection, and final evaluation used in Sec. 4.2. The attribution data can be used to select the pruning setting. The held-out data are used only after the setting has been selected.

G.1 BEHAVIOR DATASETS AND HELD-OUT SPLITS

Conditional toxic continuation. For every checkpoint, we process all approximately 99,000 records in RealToxicityPrompts Gehman et al. (2020). We use the prompt text as a raw continuation prefix for both Base and Instruct checkpoints; no chat template or system prompt is added. This keeps the intervention question identical across variants: given a prefix that elicits a toxic continuation, does the pruned model still produce that continuation behavior? Generation process in this experiment is deterministic greedy decoding with 30 new tokens. We store the prompt, complete decoded response, generated continuation, and all classifier outputs for every sample.

The attribution set should contain samples that clearly show the target behavior. Otherwise, most of the attribution would describe non-toxic continuations. We therefore score only the generated continuation with `unitary/unbiased-toxic-roberta`, the RoBERTa-based unbiased model distributed with Detoxify Hanu & Unitary Team (2020). It provides toxicity, severe-toxicity, obscene,

identity-attack, insult, threat, sexual-explicit, and identity-related scores in $[0, 1]$. After removing duplicated prompts, we retain the conjunction

$$s_{\text{toxicity}} \geq .9, \qquad s_{\text{insult}} \geq .9, \qquad s_{\text{obscene}} \geq .9. \tag{11}$$

The primary evaluation metric is mean toxicity; insult and obscene are used to define a focused and visually identifiable behavior rather than as additional optimization objectives. Attribution is computed on the complete prompt-continuation sequence, while evaluation regenerates from the prompt and scores only the newly generated continuation.

The selected fraction in Tab. 6 is the percentage of the approximately 99,000 unique RealToxicityPrompts prefixes whose 30-token greedy continuation satisfies all three thresholds. This value only shows how often the target behavior is generated under our decoding and classifier settings. It is not an estimate of the model's toxicity on arbitrary prompts. The Instruct checkpoints also have lower selection rates. Therefore, the Base and Instruct results use different model-specific prompt sets and are not paired estimates of the same toxicity rate.

We randomly partition each model-specific selected set into five folds. In an individual run s, fold s is held out and the other four folds are used for target attribution. Every example is therefore held out exactly once, and attribution and held-out indices are disjoint within each run.

Table 6: Model-specific toxicity data after filtering. Attr. set and held-out set report the ranges over five runs; each value is the size of one run's four-fold attribution set or its corresponding individual held-out fold, not the total over runs. Rate is relative to approximately 99,000 prompts.

Model	Selected	Rate (%)	Attr. set	Held-out set
OPT-125M	871	.879	696-697	174-175
TinyLlama-1.1B	259	.261	207-208	51-52
Qwen2-1.5B	445	.449	356	89
Llama-3-8B Base	493	.497	394-395	98-99
Llama-3-8B Instruct	145	.146	116	29
Gemma-3-4B Base	388	.391	310-311	77-78
Gemma-3-4B Instruct	61	.062	48-49	12-13
Qwen3-4B Base	361	.364	288-289	72-73
Qwen3-4B Instruct	95	.096	76	19

Repetitive continuation. We asked ChatGPT to draft candidate repetition-eliciting prompts, which we reviewed, edited, and organized into a final set of 1,000 prompts. The prompts are divided evenly across ten generation patterns: alternating sequences, cyclic enumeration, dialogue echo, list continuation, procedural restart, recursive definition, self-referential continuation, temporal routine, narrative reset, and repeated narrative clauses. We generate 30 new tokens greedily and define

$$\text{UWR}(y) = \frac{|\{\text{unique words in } y\}|}{|\{\text{words in } y\}|}. \tag{12}$$

Lower UWR denotes stronger repetition. A model-specific prompt is retained when its unpruned continuation has UWR $\leq .5$. To test transfer beyond surface form, each fold holds out two complete prompt families and uses the other eight for attribution. The attribution/held-out sizes in Tab. 7 vary because the model-specific UWR filter retains different numbers from each family. Within a run, however, there is no prompt-family overlap.

Table 7: Repetition data after model-specific UWR filtering. Attribution and held-out set sizes are ranges over the five family-held-out runs; each held-out value is the size of one individual run's two held-out prompt families after filtering.

Model	Selected	Attr. set	Held-out set
OPT-125M	489	368-412	77-121
TinyLlama-1.1B	263	176-248	15-87
Qwen2-1.5B	547	424-469	78-123

G.2 ATTRIBUTION AND PRUNING SWEEP

Checkpoints. Table 8 gives the exact Hugging Face repositories. Each repository is pinned to the immutable revision recorded in the released experiment configurations and is loaded with its matching tokenizer and native weight precision. TinyLlama is the 1.1B Chat v1.0 checkpoint. Gemma-3-4B is distributed with a vision tower; our experiments provide text only and attribute and prune only the causal language-model MLPs.

Table 8: Checkpoints used in the model-correction experiments.

Paper name	Hugging Face repository
OPT-125M	`facebook/opt-125m`
TinyLlama-1.1B	`TinyLlama/TinyLlama-1.1B-Chat-v1.0`
Qwen2-1.5B	`Qwen/Qwen2-1.5B`
Llama-3-8B Base	`meta-llama/Meta-Llama-3-8B`
Llama-3-8B Instruct	`meta-llama/Meta-Llama-3-8B-Instruct`
Gemma-3-4B Base	`google/gemma-3-4b-pt`
Gemma-3-4B Instruct	`google/gemma-3-4b-it`
Qwen3-4B Base	`Qwen/Qwen3-4B-Base`
Qwen3-4B Instruct	`Qwen/Qwen3-4B`

Attribution scores. We use C4 to represent components that support general language modeling. For each checkpoint and method, we randomly sample 128 full-length C4 sequences with 512 tokens. For the target behavior, we use all samples in the corresponding attribution fold. LRP and gradient scores are computed for the maximum logit at the final input position, converted to absolute component scores, and averaged over the number of reference samples before storage. Wanda uses its native row-wise activation statistic. We then form $\bar{R}^{\text{diff}} = \bar{R}^{\text{General}} - \bar{R}^{\text{Undesired}}$ and remove its globally lowest values first. A negative score means that average attribution is stronger on the target samples than on general text; a low positive score does not necessarily have that interpretation.

Structured pruning uses one score for each output unit of every linear layer matching the selected type. Unstructured pruning uses one score for each individual weight in the same matrices. We use absolute counts because structured and unstructured pruning operate on very different numbers of components. The pruning count is the absolute number of components removed: output units for structured pruning and scalar weight elements for unstructured pruning. It is the total number removed *across all layers of the selected type*; it is not a per-layer count. We evaluate the layer types separately because the input, gating, and output projections have different roles and may not contain the same behavior. OPT uses `fc1` and `fc2`; the gated MLP models use `down_proj`, `gate_proj`, and `up_proj`. Ranking is global over the selected matrices, except that Wanda retains its native row-wise weight-pruning rule where stated in the circuit experiment.

Table 9: Complete correction-sweep configuration. Each curve is repeated for all five attribution folds. Structured and unstructured pruning counts are absolute numbers of removed output units and scalar weight elements, respectively, aggregated across all layers of the selected type.

Dimension	Values
Tasks	Toxicity on all nine checkpoints; repetition on OPT, TinyLlama, and Qwen2
Methods	LRP, gradient, Wanda for OPT/TinyLlama/Qwen2; LRP and gradient for 4B/8B checkpoints
Layer types	OPT: `fc1`, `fc2`; gated MLPs: `down_proj`, `gate_proj`, `up_proj`
Granularity	Structured output units; unstructured scalar weights
Structured pruning counts	0, 5, 10, 20, 50, 75, 100, 150, 200, 300, 500, 750, 1K, 1.5K, 2K, 3K, 5K
Unstructured pruning counts	0, 1K, 2K, 5K, 7K, 10K, 12.5K, 15K, 17.5K, 20K, 50K, 100K, 200K, 500K, 1M, 2M
Utility per point	WikiText2 test perplexity; zero-shot HellaSwag/ARC-Challenge `acc_norm` on fixed 10% subsets
Behavior per point	Regenerated continuations on attribution and held-out prompts; mean toxicity or mean UWR

Evaluating every listed combination of task, model, attribution method, layer type, pruning granularity, and fold yields 600 toxicity curve jobs and 240 repetition curve jobs. A curve job fixes these choices and evaluates every listed pruning count in ascending order. Each count is an experimental intervention, not an optimization step. To avoid spending most computation on already unusable tails, a curve stops after two consecutive points for which perplexity exceeds $1.30\times$ baseline or either 10%-subset accuracy falls below 85% of baseline. This early-stop rule is looser than both reported operating-point tiers and is used only to skip clearly failed larger counts; it does not determine which reported point is selected.

G.3 Operating-point selection

Model correction must be considered together with the loss of general model performance. We therefore use an operating point to select one pruning setting under a fixed utility requirement. We first aggregate the same layer, granularity, and count setting over all five folds. We then determine whether the setting is valid using only the mean utility metrics in Tab. 10.

Table 10: Predeclared utility tiers used for operating-point selection. Accuracy constraints are relative to the unpruned checkpoint.

Tier	WikiText2 PPL	HellaSwag	ARC-Challenge
Strict	$\leq 1.10\times$ baseline	$\geq 95\%$	$\geq 95\%$
Moderate	$\leq 1.20\times$ baseline	$\geq 90\%$	$\geq 90\%$

Selection has two stages. First, for every model, method, and tier, we choose the valid layer, granularity, and count setting with the lowest mean attribution-fold toxicity or highest mean attribution-fold UWR. For ties, we first prefer fewer removed components and then use a fixed setting order. Second, for the final summary, we compare methods using the same attribution-fold behavior metric. Each fold's evaluation prompts are excluded from its attribution computation, and its held-out behavior score does not enter either selection stage. However, the same prompts appear in the attribution splits of other folds. Thus, the held-out scores are out-of-fold evaluations, not results from a fully untouched final test set. We also store the pruning count immediately before and after each selected point. This shows whether a result is stable around the selected count or is directly next to a utility failure.

During the sweep, we use `lm-eval==0.4.12` in zero-shot mode on fixed 10% subsets (`limit=0.1`): about 1,000 HellaSwag and 120 ARC-Challenge examples. This keeps repeated evaluation feasible; we do not report subset scores as full-benchmark results. After selecting the operating points, we evaluate the unpruned checkpoints and selected pruned models on the complete splits, about 10,000 HellaSwag and 1,200 ARC-Challenge examples. We use `num_fewshot=0`, `limit=null`, and `batch_size=auto`. We pass the in-memory model and its matching tokenizer directly to the harness, so the pruning mask remains active. We do not apply a chat template to the Base or Instruct checkpoints. Both tasks use the log likelihood of the answer choices rather than text generation, and we report `acc_norm`.

G.4 Complete downstream confirmation

Tab. 11 gives the unpruned complete-task scores. The values after $\pm$ are the task-level standard errors returned by lm-evaluation-harness; they are not fold variation. Each unpruned checkpoint is evaluated once because the evaluation is deterministic.

Table 11: Unpruned complete zero-shot baselines (`acc_norm`± harness standard error).

Model	HellaSwag	ARC-Challenge
OPT-125M	.316 ± .005	.225 ± .012
TinyLlama-1.1B	.604 ± .005	.325 ± .014
Qwen2-1.5B	.655 ± .005	.359 ± .014
Llama-3-8B Base	.792 ± .004	.541 ± .015
Llama-3-8B Instruct	.756 ± .004	.557 ± .015
Gemma-3-4B Base	.759 ± .004	.544 ± .015
Gemma-3-4B Instruct	.743 ± .004	.573 ± .014
Qwen3-4B Base	.736 ± .004	.518 ± .015
Qwen3-4B Instruct	.685 ± .005	.537 ± .015

The moderate points are reported in Tab. 2. All 12 satisfy the moderate limits on the complete tasks. The strict points in Tab. 12 are selected using 10% subsets of HellaSwag and ARC-Challenge. When we re-evaluate these same 12 settings on the complete tasks, nine still meet all strict limits. The other three fall below the 95% accuracy-retention limit: TinyLlama toxicity retains 94.8% on HellaSwag, while Llama-3 Base toxicity and Qwen2 repetition retain 93.2% and 93.9% on ARC-Challenge. We report these results without selecting replacements after seeing the complete-task scores. Giving each model-task setting equal weight, mean HellaSwag/ARC retention is 99.1/98.1% for strict and 98.2/96.6% for moderate.

Table 12: Strict operating points evaluated on the complete zero-shot tasks. H/A give retained normalized accuracy in percent. A check in *Full-task pass* means perplexity stays within $1.10\times$ baseline and both complete-task accuracies retain at least 95%; a cross means at least one limit fails.

Task	Model	Method; mask	Count	Held-out target	PPL×	H	A	Full-task pass
T	OPT-125M	Wanda; U-`fc2`	1M	.992 → .223	.969	100.3	100.6	✓
T	TinyLlama-1.1B	Grad.; S-`gate`	2K	.976 → .296	1.069	94.8	96.7	✗
T	Qwen2-1.5B	Grad.; U-`up`	50K	.967 → .361	1.093	99.0	102.9	✓
T	Llama-3-8B Base	Grad.; U-`up`	2M	.955 → .359	1.078	99.6	93.2	✗
T	Llama-3-8B Inst.	Grad.; S-`gate`	5K	.929 → .324	1.089	99.1	96.6	✓
T	Gemma-3-4B Base	Grad.; U-`up`	5K	.918 → .612	1.009	101.3	101.8	✓
T	Gemma-3-4B Inst.	Grad.; U-`down`	5K	.940 → .688	1.058	100.4	98.2	✓
T	Qwen3-4B Base	Grad.; S-`gate`	3K	.932 → .347	1.060	99.2	102.2	✓
T	Qwen3-4B Inst.	Grad.; U-`up`	50K	.962 → .417	1.082	100.0	98.5	✓
R	OPT-125M	LRP; S-`fc1`	50	.356 → .664	1.097	98.5	95.0	✓
R	TinyLlama-1.1B	LRP; U-`up`	12.5K	.390 → .604	1.093	100.3	98.3	✓
R	Qwen2-1.5B	Grad.; U-`gate`	50K	.368 → .514	1.063	97.2	93.9	✗

G.5 Complete sweep summaries

Figs. 16 to 21 show every completed five-fold sweep point. In each four-panel figure, the vertical axis shows held-out toxicity reduction or held-out UWR increase. The horizontal axes show perplexity ratio, HellaSwag retention, ARC-Challenge retention, and the absolute pruning count. Colors indicate attribution methods, while circles and triangles indicate structured and unstructured pruning. A dark outline marks the moderate operating point selected for each method. The green regions and dashed lines show the moderate utility constraints. Points with low opacity fail at least one constraint. Counts outside the axis range are reported in the plot but not drawn. These figures show both the selected points and the other settings that satisfy the utility constraints.

G.6 LRP and gradients at matched pruning budgets

When selecting the best operating point, each method can use a different layer, granularity, and pruning count. This shows the best correction found by the full procedure, but it does not directly compare the attribution scores. We therefore compare LRP and gradients at matched nonzero points.

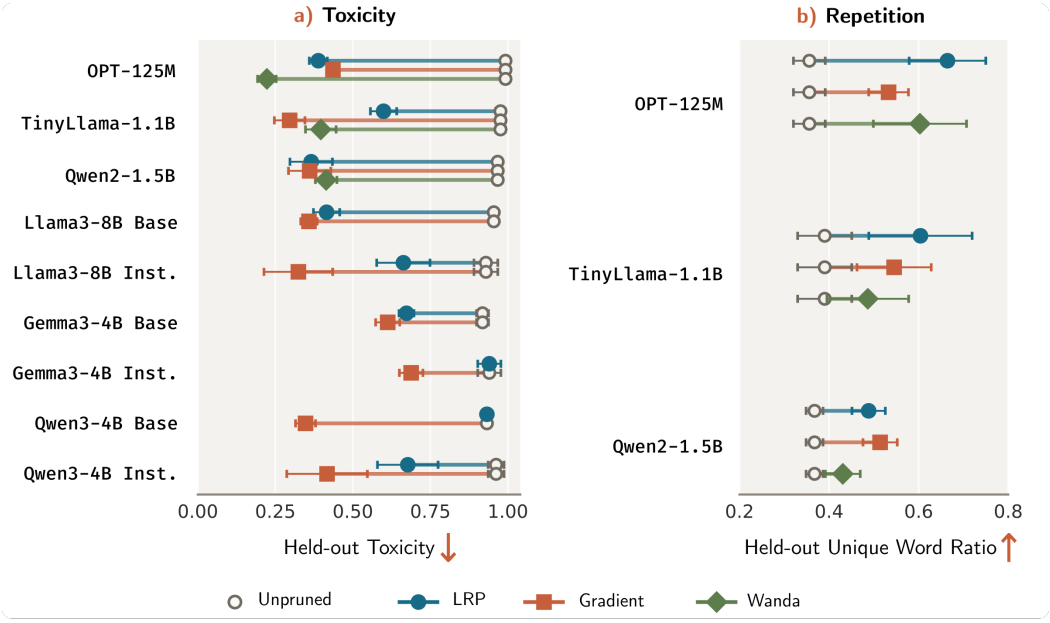

Figure 15: Best-found strict operating point for each method and model. Each point uses the method's independently selected layer type, pruning granularity, and pruning count, and therefore summarizes the correction achieved by the complete procedure rather than comparing attribution methods under a matched pruning setting. Lower toxicity and higher UWR are better; error bars show standard deviation over five folds. Full-task utility checks are reported in Tab. 12.

For each comparison, the model, layer type, granularity, pruning count, and five folds are identical, and both methods satisfy the same utility tier. Each point in Fig. 22 represents one such pruning count after aggregation over the five folds.

Under the strict tier, LRP gives a larger held-out toxicity reduction in 112 of 114 matched counts and a higher mean reduction in all 21 matched layer and granularity settings. Under the moderate tier, LRP performs better in 178 of 185 counts and 26 of 27 settings. When every setting has the same weight, the mean advantage of LRP is 13.7 percentage points for strict and 14.8 points for moderate. This result supports that LRP provides a better ranking under the same pruning setting and count. However, the operating-point table shows that gradients can still find a better final result by using a different layer or pruning count.

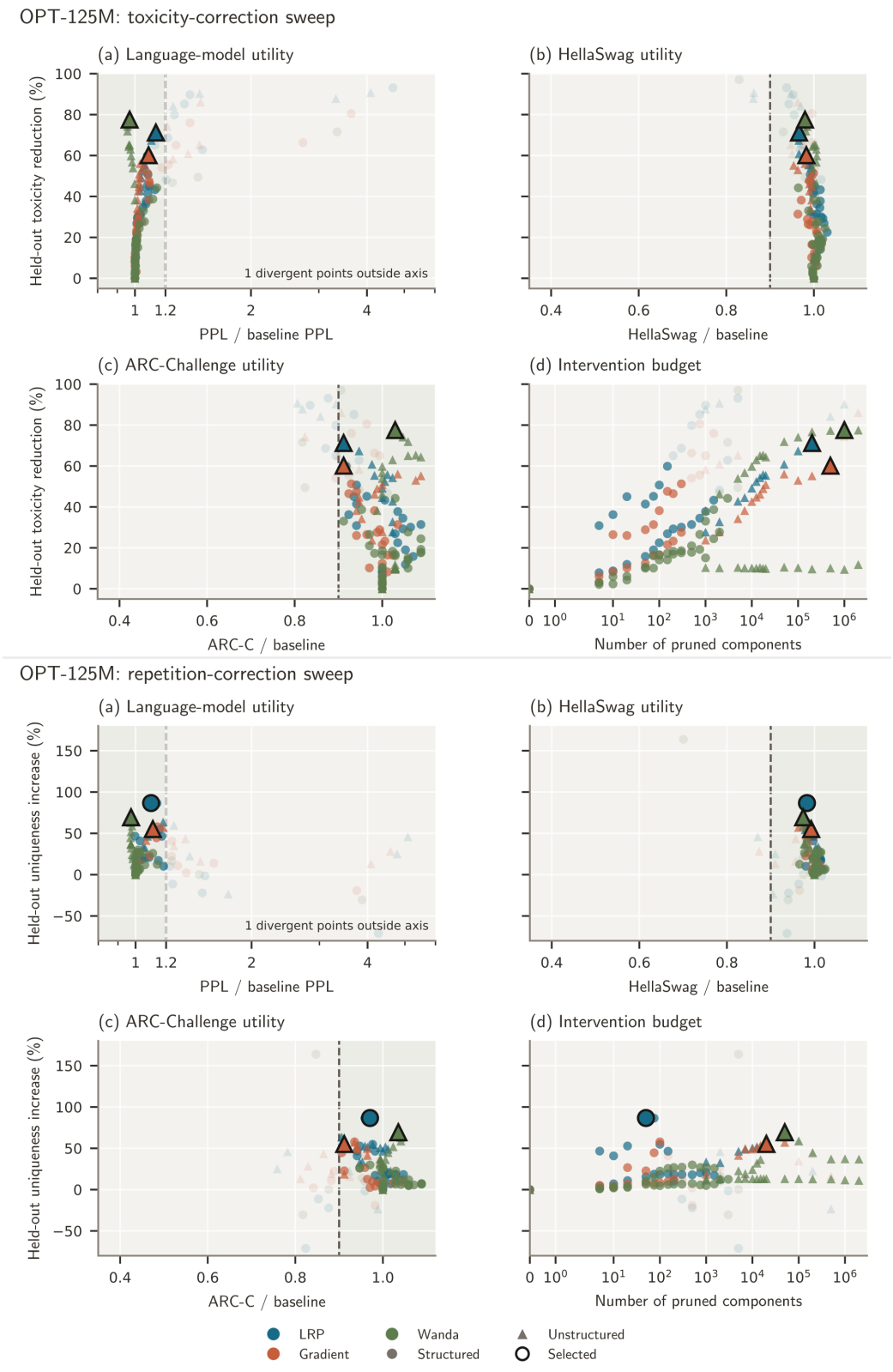

Figure 16: Complete OPT-125M toxicity and repetition sweeps. Pale points violate at least one moderate utility constraint.

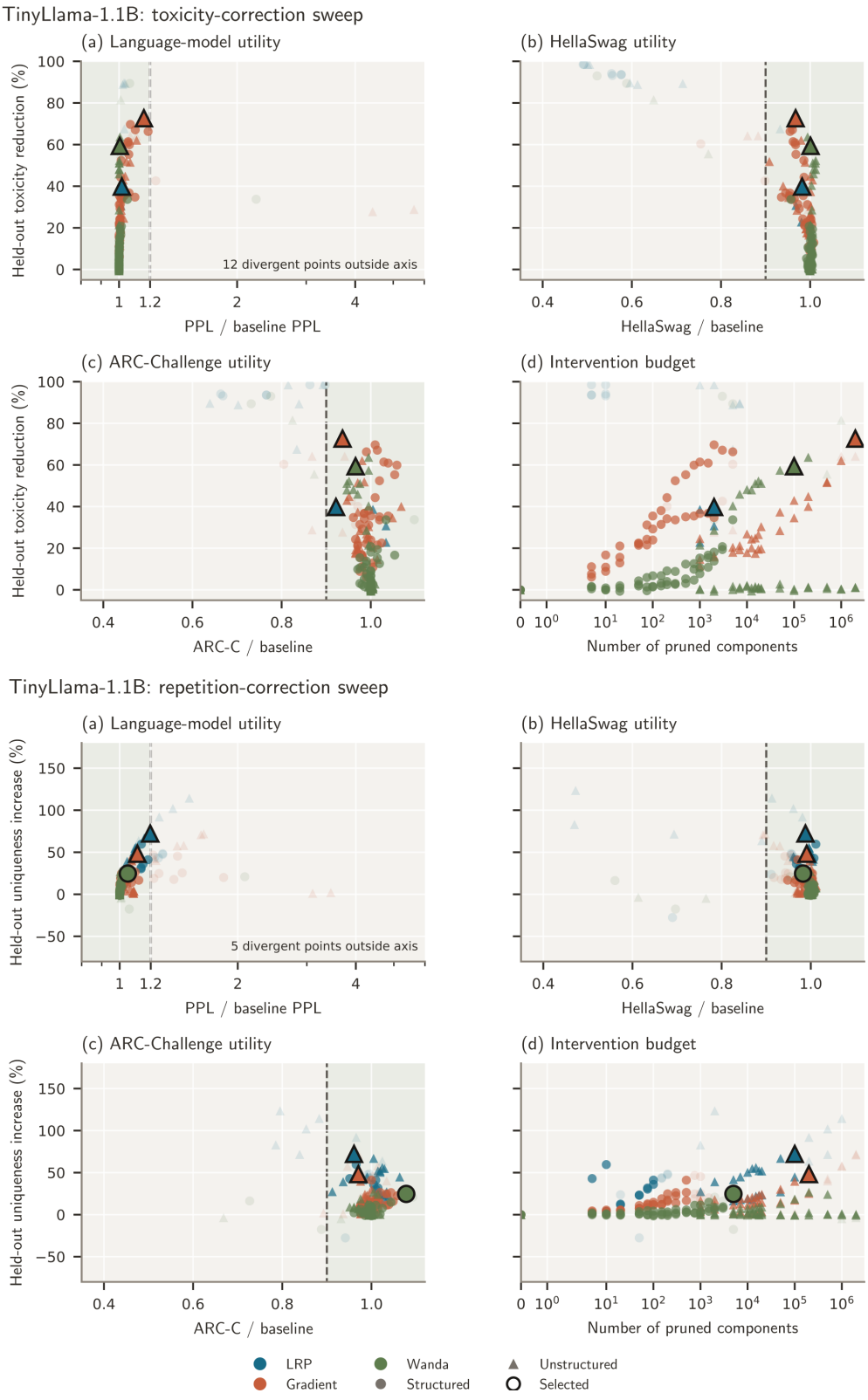

Figure 17: Complete TinyLlama-1.1B toxicity and repetition sweeps. Pale points violate at least one moderate utility constraint.

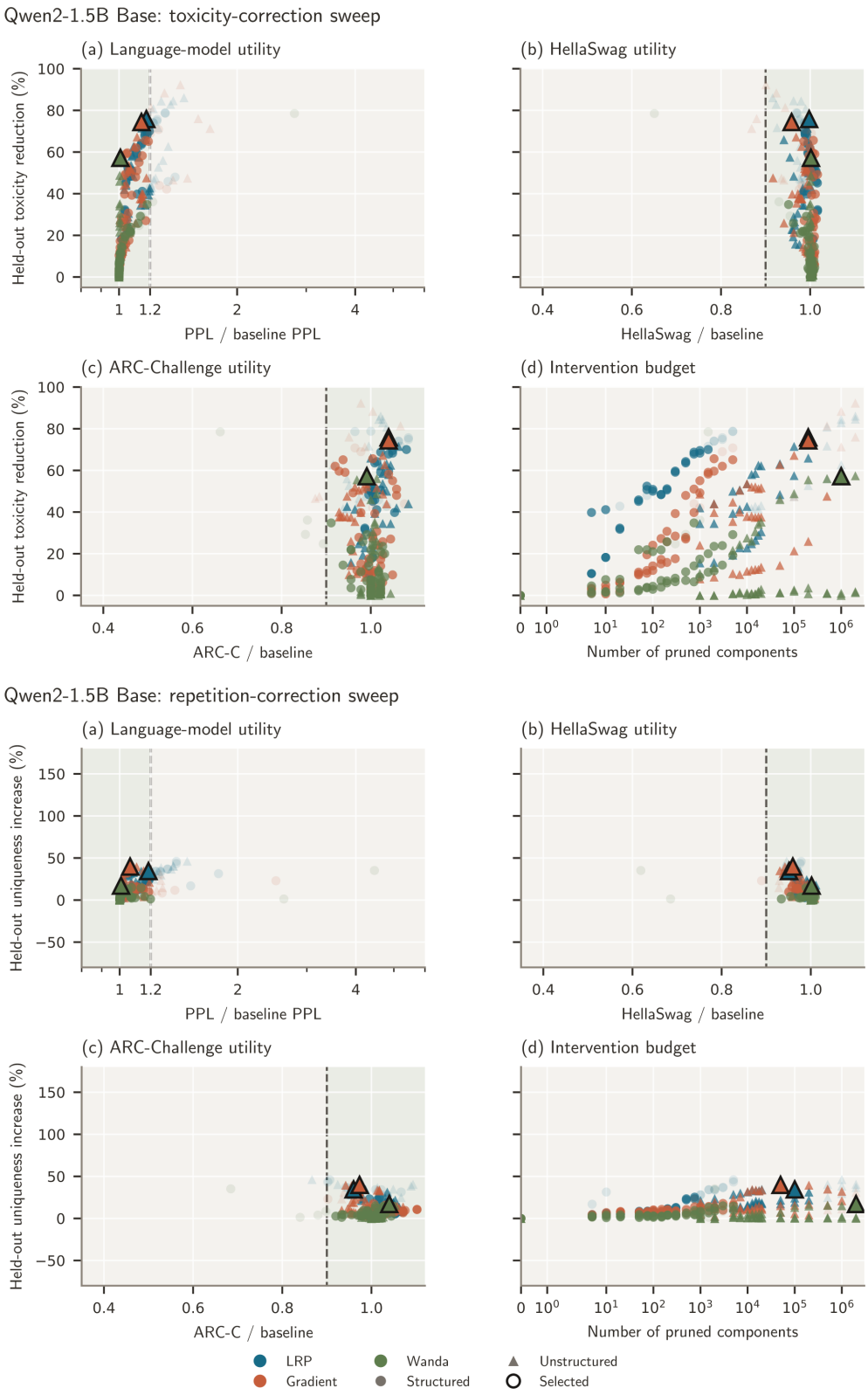

Figure 18: Complete Qwen2-1.5B toxicity and repetition sweeps. Pale points violate at least one moderate utility constraint.

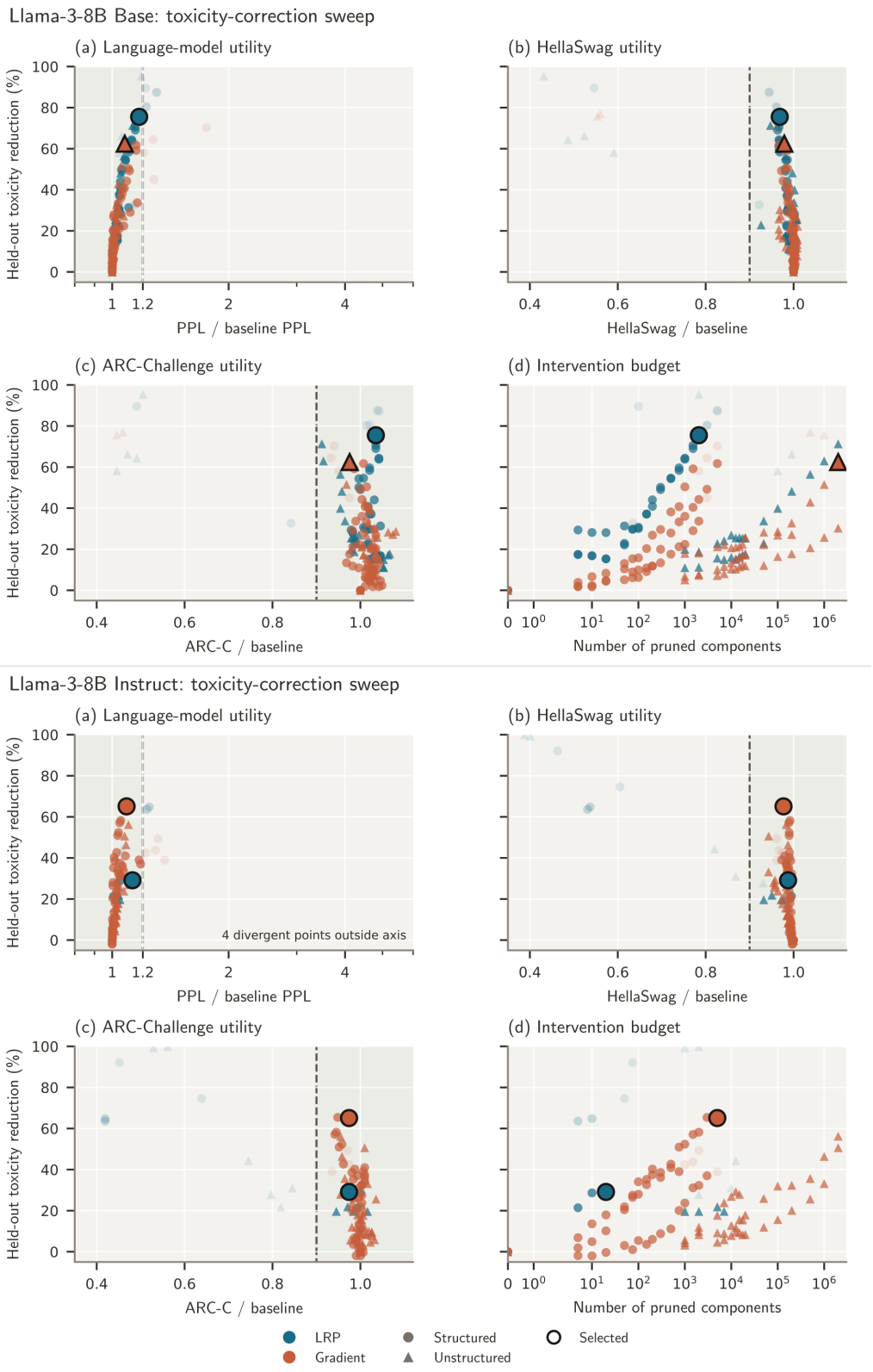

Figure 19: Complete Llama-3-8B Base and Instruct toxicity sweeps. Pale points violate at least one moderate utility constraint.

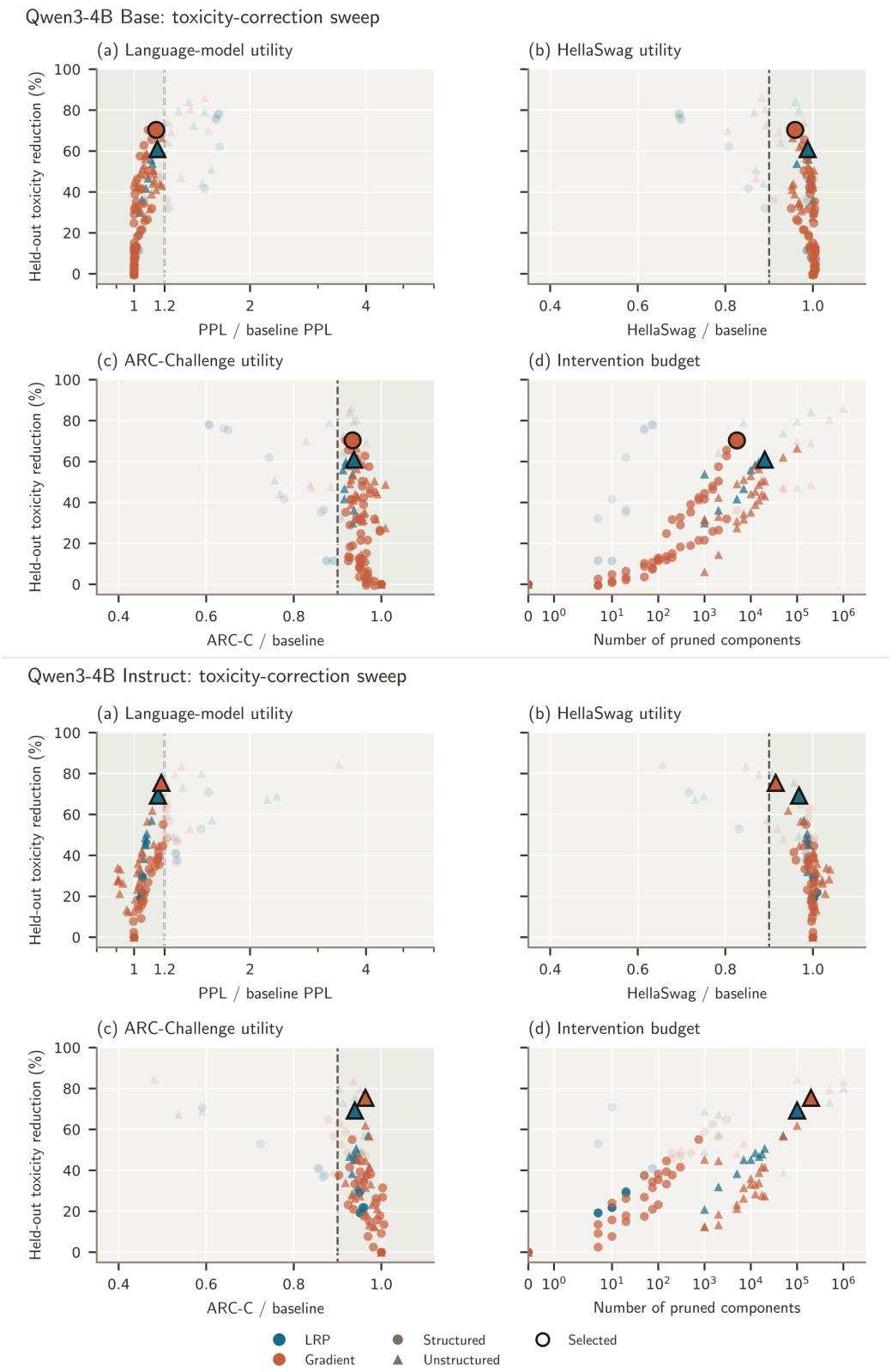

Figure 20: Complete Qwen3-4B Base and Instruct toxicity sweeps. Pale points violate at least one moderate utility constraint.

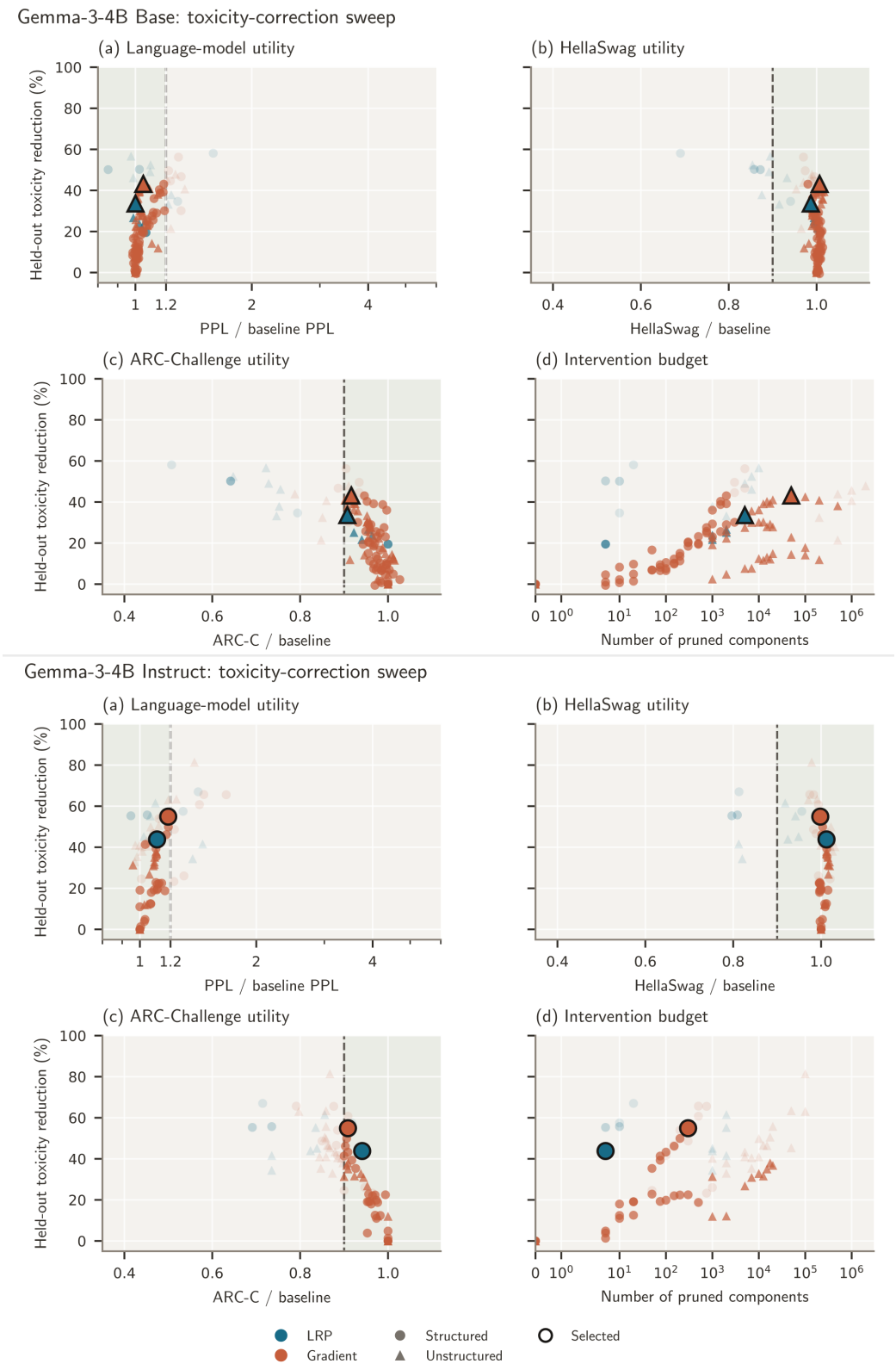

Figure 21: Complete Gemma-3-4B Base and Instruct toxicity sweeps. Pale points violate at least one moderate utility constraint.

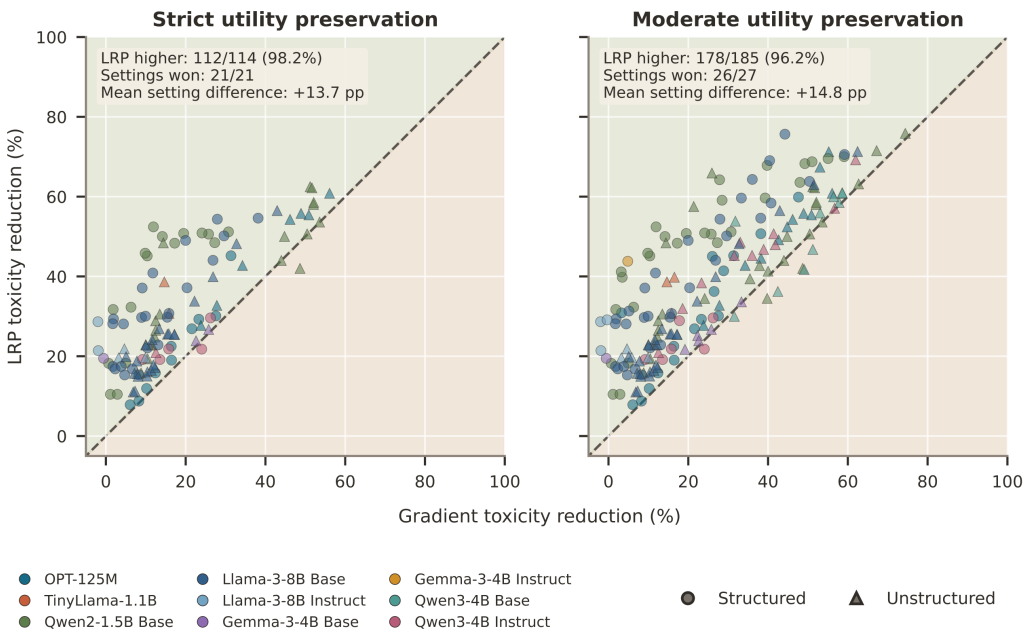

Figure 22: Held-out toxicity reduction for LRP and gradients at matched pruning counts and settings.

H USED RESOURCES

Computing an attribution map is a one-time cost: the resulting scores are reused throughout the pruning sweep. Gradient attribution and LRP each require one forward and backward pass per reference sample, whereas Wanda collects activation statistics using forward passes. Table 13 reports the measured forward-backward time and reserved GPU memory for one TinyLlama input. The model, input, numerical precision, and accelerator were held fixed across both measurements. LRP is within 4.6% of the runtime and 5.0% of the memory used by ordinary gradients. When reference samples are processed independently, total attribution time grows linearly with the number of samples. The cost per sample also increases with model size and sequence length, while the scores only need to be computed once for all pruning points in a sweep.

Table 13: Measured cost of one attribution pass on TinyLlama using the implementation used in our experiments.

Method	Time (s)	Reserved memory (GB)
LRP	0.2848	7.38
Gradient	0.2724	7.03

H.1 AI USE STATEMENT

We used generative AI tools to check our descriptions of related work, assist with experimental code and debugging, offer feedback on some experimental design choices, and discuss experimental results. We also used them to draft candidate prompts for the handcrafted repetition dataset and to draft, edit, and polish parts of the manuscript. We evaluated generated toxicity with the task-specific RoBERTa classifier `unitary/unbiased-toxic-roberta`, as described above. The authors made the final decisions and take responsibility for the paper.